第十五屆旺宏科學獎

成果報告書

參賽編號：SA15-069

作品名稱：百動不如一靜——

固定策略玩家對奈許均衡的影響

姓名：林宛萱

關鍵字：沙灘賣冰模型、奈許均衡、保證獲利

壹、 研究動機

在上數學課時，老師曾跟我們提起有關於經濟學上的沙灘賣冰理論。沙灘賣冰問題是一個空間競爭模型，探討沙攤上各攤販會為了獲得最大利益而不斷改變自己的設攤位置，當攤販們不再變動時便達到奈許均衡。這個理論可以解釋眾多人類行為，從企業對產品價帶的選定，到選舉時候選人政治光譜的設定，皆可用沙灘賣冰理論來解釋其行為模式。

我們在研究此模型時，發現某些位置的攤販可獲得較大利益，因此我們嘗試加入『不會隨著其他競爭者改變策略』的玩家，如同在沙灘賣冰模型中加入固定位置的攤販，並觀察其是否可以形成均衡及其獲利，進而分析固定攤販對奈許均衡存在性的影響，及其與其他攤販的獲利關係。

貳、 研究目的

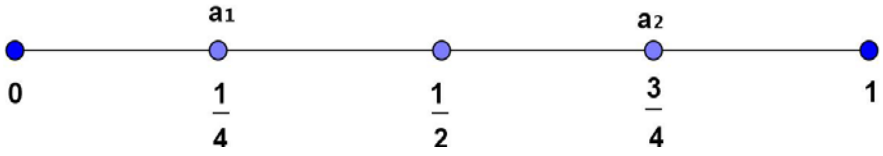
一、探討固定攤販對奈許均衡的影響
二、固定攤販的保證獲利
三、探討增加固定攤販數量時的均衡情形
四、探討遊客數量分布不均的均衡情形
五、以程式模擬模型固定攤販與流動攤販的均衡情形

參、 研究設備與器材

紙、筆、電腦、Geogebra、Microsoft Visual Studio C++、Excel

肆、 研究過程與方法

在沙灘賣冰模型中，我們假設遊客是平均分布在長度為 1 的沙灘上，且沙灘上的攤販不論品質或價格都是相同的，因此遊客只會依據距離選擇去較近的攤販消費，而沙灘上的流動攤販均是短視近利的，他們會為了擁有最大客源以獲得最大利益而不斷移動位置，但是若他無法因為移動而獲得更多利益時，他便會停在該位置，並不會為了影響其他人獲利而移動。當所有攤販都靜止不再移動時，我們便稱其達到奈許均衡。



奈許均衡(Nash equilibrium)是賽局理論中一種解的概念。以兩家攤販(a_1, a_2)的沙灘賣冰模型為例，理想的狀態是兩人分別設攤於$a_1 = \frac{1}{4}$與$a_2 = \frac{3}{4}$，這樣所有遊客為了買冰所走的距離都不會超過$\frac{1}{4}$，兩人也各分得$\frac{1}{2}$均等的利潤。

但事實上，這兩家攤販不會停在這個位置，因為只要往對方移動一點，就可以得到更多一點的客源。於是兩個攤販會不停往中間移動，直到兩人都停在$\frac{1}{2}$。此時不論哪一個人離開$\frac{1}{2}$，所得到的利益都會變小，因此兩個攤販便會固定在$\frac{1}{2}$的位置，不再移動。這就是奈許提出來的『均衡』概念。

定義(奈許均衡)：

假設一個賽局總共有 n 個參與者，每個參與者個策略空間分別為 $S_1, S_2, \cdots, S_n$，報酬函數為 $f_1, f_2, \cdots, f_n$，其中 $f_k : S_1 \times S_2 \times \cdots \times S_n \to \mathbb{R}$。當一組 $(a_1^*, a_2^*, \cdots, a_n^*) \in S_1 \times S_2 \times \cdots \times S_n$，滿足

$f_k(a_1^*, a_2^*, \cdots, a_k^*, \cdots, a_n^*) \geq f_k(a_1^*, a_2^*, \cdots, a_k, \cdots, a_n^*) \ \forall a_k \in S_k, k = 1, 2, \cdots, n$，

則稱此策略組合 $(a_1^*, a_2^*, \cdots, a_n^*)$ 為這個賽局的一組奈許均衡。

這個理論可以解釋眾多人類行為，主要的應用在於企業對產品的定位及選舉時候選人對政策的設定，我們皆可以將沙灘視為商品價帶及政治光譜，而攤販則是商場上的企業及參選人的政策取向，以理論來為其行為模式解釋。

均衡時若出現兩家攤販位於同一個點的情況，我們稱之為共點，假設此時兩家攤販是緊鄰在隔壁，且兩家攤販平分其所獲得的利益範圍。根據文獻中的研究，我們可以得知：兩家攤販均位於 $\frac{1}{2}$ 時會產生均衡；三家攤販不會有均衡產生；四家以及五家攤販則會產生唯一均衡。六家以上攤販均衡型態並不唯一。

而文獻中提出沙灘賣冰模型中均衡形成的充分必要條件為：

定理 1(Eaton - Lipsey**)：沙灘賣冰模型中均衡形成的充分必要條件為：**

1. **任一攤販的獲利不大於任一攤販的兩倍。**
2. **距端點最近之攤販必為共點攤販。**

同時，根據文獻及計算例子所獲得的結果，我們歸納出了以下三個引理：:

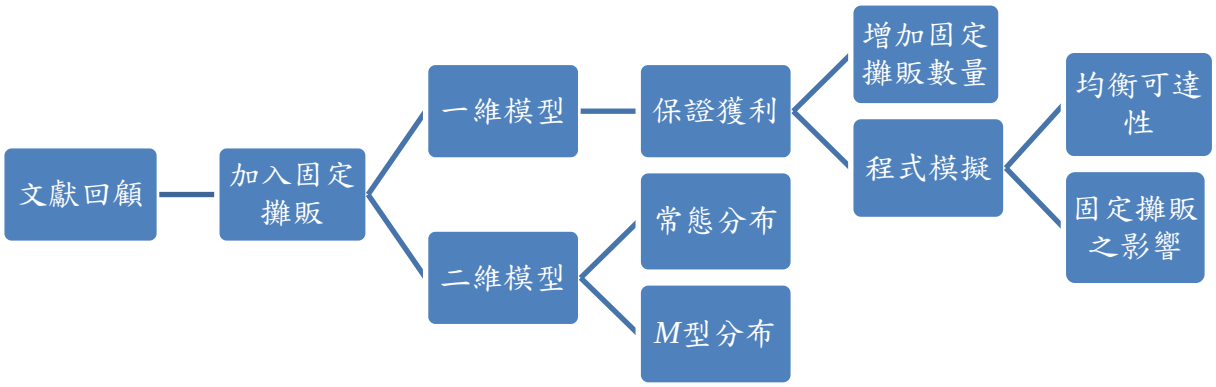
引理 1：達成均衡時，在任何一個位置上，最多只會有兩個攤販共點，並且共點攤販左右兩側分得的區域相等。

引理 2：在均衡時，最靠近兩端點的攤販必定兩兩共點。

引理 3：均衡時，每一家共點攤販的利益皆相同。因此，若均衡時最左側的共點攤販位於 a，則與他們相鄰的攤販會位於 $3a$，而最右側的共點攤販位於 $1-a$，並且與他們相鄰的攤販會位於 $1-3a$。

我們發現在均衡的例子中，會有某些位置的攤販分得比其他人還要多的獲利。以五家攤販為例，位於 $\frac{1}{2}$ 的攤販可以獲得 $\frac{1}{3}$ 的利益，而其餘四家攤販僅能各分得 $\frac{1}{6}$。因此我們猜測：若在模型中加入一家一旦選擇了位置即不會再做任何移動的固定攤販，而當他選擇位於可以擁有最大獲利的位置時，是否就真的可以獲得較多的利益？而這樣的固定攤販會對原本均衡的結果產生什麼影響？因此我們做了以下研究，探討固定攤販的位置會對均衡產生何種影響，並探求其獲益和流動攤販相比，是否較為有利。

研究架構

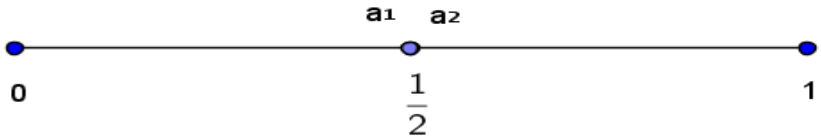


一、 固定攤販對奈許均衡的影響

在以下討論中，我們假設在沙灘賣冰模型中，有 1 個攤販在 b 的位置固定不動，及 n 家會隨著利益改位置的流動攤販。同時，當流動攤販達成均衡時，假設其分別位於 $a_1, a_2, \cdots, a_n$，並且 $0 \le a_1 \le a_2 \le \cdots \le a_n \le 1$；而他們所獲得的利益則分別為 f_b, f_1, …, f_n，並且根據對稱性，我們只需要討論 $0 \le b \le \frac{1}{2}$ 的情形。而為了方便起見，在討論各種均衡形態時，我們以 b 表示固定攤販，以 1 表示單獨的流動攤販、2 表示共點的流動攤販，而 b+1 則表示固定攤販與一家流動攤販共點的情形。例如：$(b,1,1,2)$ 表示在 4 家流動攤販達成均衡時，他們及固定攤販分別位於 $b < a_1 < a_2 < a_3 = a_4$，而 $(2,b+1,1,2)$ 則表示 6 家流動攤販達成均衡時，$a_1 = a_2 < b = a_3 < a_4 < a_5 = a_6$ 的情形。我們先依據前面的引理以及定理歸納出均衡時可能的型態，再計算出固定攤販可能的位置範圍及其可以分得的利益。

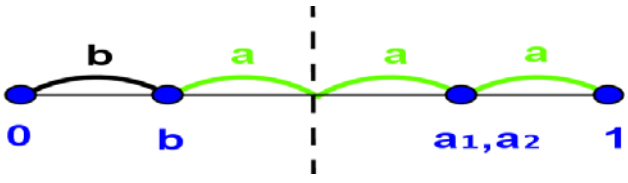
(一) 1 個固定攤販+1 家流動攤販

當 $b \neq \frac{1}{2}$ 時，流動攤販會位於固定攤販利益範圍較長的一側並向他靠近，但不會共點，故此時沒有奈許均衡。因此，均衡態只有一種 $b = a_1 = \frac{1}{2}$，如<圖 1-1.1>。



<圖 1-1.1>

(二) 1 個固定攤販+2 家流動攤販



<圖 1-2.1>

根據引理一～三，形成均衡時，只有 $b < a_1 = a_2$，或 $a_1 = a_2 < b$ 兩種可能，根據對稱性，不妨假設 $b \le \frac{1}{2} \Rightarrow b < a_1 = a_2$，如<圖 2-2.1>所示。若 $a_1 = a_2 = 1-a$，則 $b = 1-3a$，此時 $f_b(b,1-a,1-a) = b+a = 1-2a$，$f_1 = f_2(b,1-a,1-a) = a$。

根據定理 1，我們可以得到以下關係式：

$$0 \le b = 1-3a \le a \Leftrightarrow \frac{1}{4} \le a \le \frac{1}{3} \text{ 且 } 0 \le b \le \frac{1}{4}\ ,$$

加上對稱性，故均衡存在的充分必要條件是

$$0 \le b \le \frac{1}{4} \text{ 或 } \frac{3}{4} \le b \le 1\ 。$$

結論：$n=2$ 時，均衡產生在

$$(b, a_1, a_2) = \begin{cases} \left(b, \frac{b+2}{3}, \frac{b+2}{3}\right), & \text{當 } 0 \le b \le \frac{1}{4} \\ \left(b, \frac{b}{3}, \frac{b}{3}\right), & \text{當 } \frac{3}{4} \le b \le 1 \end{cases}$$

並且，三家攤販所分得的利益分別為

$$f_b\left(b, \frac{2+b}{3}, \frac{2+b}{3}\right) = \frac{1-2b}{3}\ ,\ f_1 = f_2\left(b, \frac{2+b}{3}, \frac{2+b}{3}\right) = \frac{1-b}{3}\ ,\ \text{當 } 0 \le b \le \frac{1}{4}\ ,$$

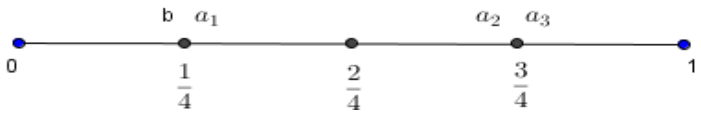
或 $f_b\left(b, \frac{b}{3}, \frac{b}{3}\right) = \frac{3-2b}{3}$ ， $f_1 = f_2\left(b, \frac{b}{3}, \frac{b}{3}\right) = \frac{b}{3}$ ，當 $\frac{3}{4} \le b \le 1$ 。

由此可知，當 $x = \frac{1}{4}$ 或 $\frac{3}{4}$ 時，b 可獲得最大利益，此時均衡態為 $(b, a_1, a_2) = \left(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}\right)$ 或 $\left(\frac{3}{4}, \frac{1}{4}, \frac{1}{4}\right)$ ，並且 $f_b = \frac{1}{2}$, $f_1 = f_2 = \frac{1}{4}$ 。而當 $x = 0$ 或 1 時，b 有最小利益，此時 $f_b = f_1 = f_2 = \frac{1}{3}$ 。

(三) 1 個固定攤販+3 家流動攤販

根據對稱性，我們同樣假設 $b \le \frac{1}{2} \Rightarrow b \le a_1 < a_2 = a_3$ ，則前面的引理和定理告訴我們，1 個固定攤販+3 家流動攤販只有如<圖 2-3-1.1>或<圖 2-3-2.1>二種情形。

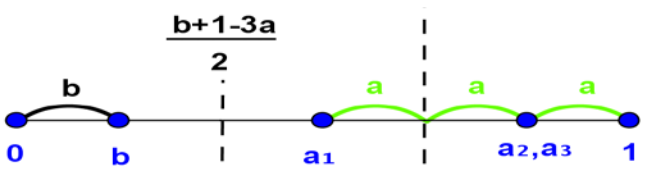
1. $(b+1, 2)$ ：



<圖 1-3-1.1>

若均衡時兩兩共點，根據前面的討論我們可以得知均衡點為 $\frac{1}{4}$ 、$\frac{3}{4}$ ，故 $(b, a_1, a_2, a_3) = (\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, \frac{3}{4})$ ，此時 $f_b = f_1 = f_2 = f_3 = \frac{1}{4}$ 。

2. $(b, 1, 2)$ ：



<圖 1-3-2.1>

根據引理 3，我們可以假設均衡時 $a_1=1-3a, a_2=a_3=1-a$，此時

$$f_b=\frac{b+1-3a}{2},\ f_1=\frac{1-b-a}{2},\ f_2=f_3=a\ 。$$

由於 $b\ge 0, b+3a<1$，且根據定理 1，我們可以得到：

$$\begin{cases} b\le a \\ \dfrac{b+1-3a}{2}\le 2b \Leftrightarrow a+b\ge\dfrac{1}{3} \\ a\le\dfrac{1-a-b}{2}\Leftrightarrow 3a+b\le 1 \end{cases}$$

因此 $b+3b\le b+3a<1$ 及 $a+a\ge a+b\ge\frac{1}{3}$，故 $0<b<\frac{1}{4}$ 且 $\frac{1}{6}\le a<\frac{1}{3}$。

結論：當 $n=3$ 且 $b\le\frac{1}{2}$ 時，均衡產生在 $(\frac{1}{4},\frac{1}{4},\frac{3}{4},\frac{3}{4})$ 或 $(b,a_1,a_2,a_3)=(b,1-3a,1-a,1-a)$

其中 $0\le b<\frac{1}{4},\frac{1}{6}\le a<\frac{1}{3}$ 且 $b\le a$。

並且，四家攤販所分得的利益分別為 $f_b=f_1=f_2=f_3=\frac{1}{4}$，當 $b=\frac{1}{4}$ 或 $\frac{3}{4}$

或 $f_b=\frac{b+1-3a}{2},\ f_1=\frac{1-b-a}{2},\ f_3=f_4=a$，當 $0\le b<\frac{1}{4},\frac{1}{6}\le a<\frac{1}{3}$ 且 $b\le a$。

由此可知，當 $b=\frac{1}{4}$ 時，b 有確定利益 $\frac{1}{4}$。而當 $a=b=\frac{1}{6}$，則 b 有最大利益 $\frac{1}{3}$，此時均衡為 $(b,a_1,a_2,a_3)=(\frac{1}{6},\frac{1}{2},\frac{5}{6},\frac{5}{6})$，並且 $f_b=f_1=\frac{1}{3},\ f_2=f_3=\frac{1}{6}$。

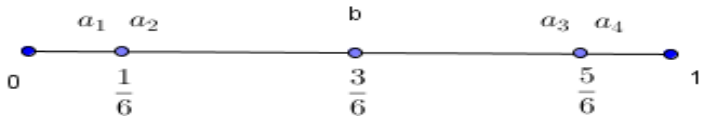
然而當 $b=\frac{1}{6}$ 時，對任意的 $\frac{1}{6}\le a<\frac{1}{3}$，$(b,a_1,a_2,a_3)=(\frac{1}{6},1-3a,a,a)$ 都會是奈許均衡，因此當固定攤販位於 $\frac{1}{6}$ 時，其最大獲利為 $\frac{1}{3}$，最小獲利則會大於 $\frac{1}{6}$。

(四) 1 個固定攤販+4 家流動攤販

如同先前的討論，$n=4$ 時我們同樣假設 $0\le b\le\frac{1}{2}$，則奈許均衡有 $(2,b,2)$、$(b,2,2)$、$(b+1,1,2)$ 或 $(b,1,1,2)$ 四種可能：

1. $(2,b,2)$：

根據先前討論，在此情況下只有一個唯一的奈許均衡如<圖 1-4-1.1>：

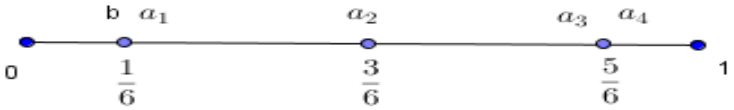


<圖 1-4-1.1>

此時 $(a_1,a_2,b,a_3,a_4)=(\frac{1}{6},\frac{1}{6},\frac{1}{2},\frac{5}{6},\frac{5}{6})$，並且 $f_b=\frac{1}{3},\ f_1=f_2=f_3=f_4=\frac{1}{6}$。

2. $(b+1,1,2)$：

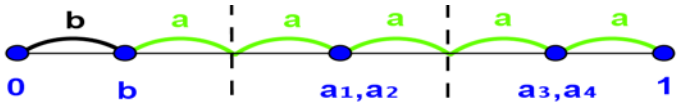
假設均衡時$b=a_1, a_3=a_4=1-a$，則$a_2=3b=1-3a$由於$a\le b$且$b\le a$，所以$b=a=\frac{1}{6}$，故在這種情況下只有一個唯一的奈許均衡如<圖 1-4-2.1>：



<圖 1-4-2.1>

此時$(b,a_1,a_2,a_3,a_4)=(\frac{1}{6},\frac{1}{6},\frac{1}{2},\frac{5}{6},\frac{5}{6})$，並且$f_b=f_1=f_3=f_4=\frac{1}{6}$, $f_2=\frac{1}{3}$。

3. $(b,2,2)$：



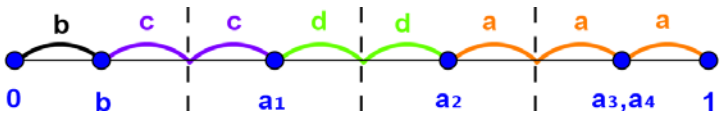
<圖 1-4-3.1>

如<圖 1-4-3.1>，假設均衡時$a_1=a_2=1-3a, a_3=a_4=1-a$，則$b=1-5a$。因為$0\le b\le a$，則$b+5b\le b+5a\le 1\Rightarrow b\le\frac{1}{6}$且$\frac{1}{6}\le a<\frac{1}{5}$。此時均衡為

$$(b,a_1,a_2,a_3,a_4)=(1-5a,1-3a,1-3a,1-a,1-a)$$，

並且$f_b=1-4a$, $f_1=f_2=f_3=f_4=a$。

4. $(b,1,1,2)$：



<圖 1-4-4.1>

如<圖 1-4-4.1>，假設均衡時$a_1=b+2c, a_2=1-3a, a_3=a_4=1-a$，並令$d=\frac{1-b-2c-3a}{2}$，則此時$f_b=b+c$, $f_1=c+d$, $f_2=d+a$, $f_3=f_4=a$。

根據定理 1，可得知$b\le a, c\le a, d\le a$且$c+d\ge a$,。因為

$$d=\frac{1-b-2c-3a}{2}\le a\Rightarrow 5a+b+2c\ge 1$$，

故$5a+a+2a\ge 5a+b+2c\ge 1$，因此$a\ge\frac{1}{8}$。又

$$c+d=c+\frac{1-b-2c-3a}{2}\ge a\Rightarrow 5a+b\le 1$$，

故$5b+b\le 5a+b\le 1$，因此$b\le\frac{1}{6}$。

由 2.～4. 我們可以得知，若固定攤販位於$b=\frac{1}{6}$時，其最大利益為$\frac{1}{3}$，最小利益則是$\frac{1}{6}$。然而，若固定攤販位於$b=\frac{1}{2}$，則可得到確定獲利$\frac{1}{3}$，又根據定理 1，$f_b\le f_k$, $\forall k=1,2,3,4$，

而 $f_b+f_1+f_2+f_3+f_4=1$，故 $f_b\le\frac{1}{3}$，此時固定攤販的利益 $\frac{1}{3}$ 為最大值。

結論：當 $n=4$ 且 $b\le\frac{1}{2}$ 時，奈許均衡存在的充分必要條件是

$$b=\frac{1}{2}\text{ 或 }0\le b\le\frac{1}{6}\text{。}$$

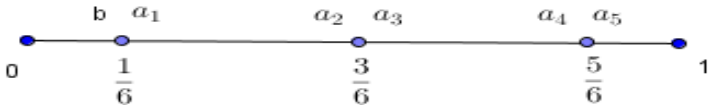
且當 $b=\frac{1}{2}$ 時，固定攤販可以確定得到其獲利的最大值 $\frac{1}{3}$。

(五) 1 個固定攤販+5 家流動攤販

隨著攤販數的增加，均衡型態也跟著複雜了起來，為了簡化，我們同樣假設固定攤販位於 $b\le\frac{1}{2}$，則均衡時可能的型態有以下六種：

1. $(b+1,2,2)$：

根據文獻，此種型態(2,2,2)的均衡解必定如<圖 1-5-1.1>。

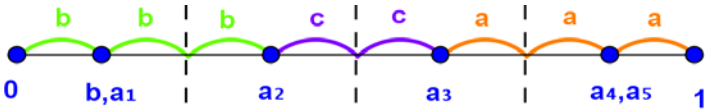


<圖 1-5-1.1>

故 (b,a_1,a_2,a_3,a_4,a_5) or $(a_1,a_2,b,a_3,a_4,a_5)=(\frac{1}{6},\frac{1}{6},\frac{1}{2},\frac{1}{2},\frac{5}{6},\frac{5}{6})$，

並且 $f_b=f_1=f_2=f_3=f_4=f_5=\frac{1}{6}$。

2. $(b+1,1,1,2)$：



<圖 1-5-2.1>

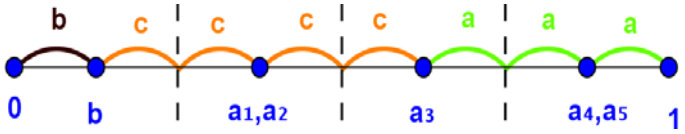
如<圖 1-5-2.1>所示，根據引理三，奈許均衡為

$$(b,a_1,a_2,a_3,a_4,a_5)=(b,b,3b,1-3b,1-b,1-b)\text{，}$$

其中 $0<b<\frac{1}{6}$，並且 $f_b=f_1=f_4=f_5=b,\ f_2=f_3=\frac{1-4b}{2}$。

令 $c=\frac{1-6b}{2}$ 由定理 1，可得 $c=\frac{1-6b}{2}\le b$，故 $b\ge\frac{1}{8}$。

3. $(b,2,1,2)$：



<圖 1-5-3.1>

如<圖 1-5-3.1>所示，設均衡時 $a_1=a_2=b+2c,\ a_3=b+4c,\ a_4=a_5=1-a$，

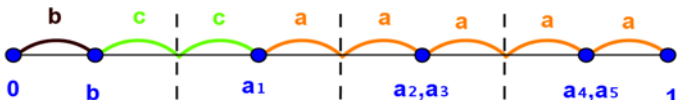
由引理三，可得 $a_3 = b + 4c = 1 - 3a$ ，因此 $c = \dfrac{1-b-3a}{4}$ 。

$$f_b = b + c,\ f_1 = f_2 = c,\ f_3 = c + a,\ f_4 = f_5 = a \text{ 。}$$

根據定理 1，可得 $b \le a, b \le c, c \le a$ 且 $a \le c$ ，
故 $a = c$ ，而且 $b = 1 - 7a$ 。因此

$$7a + a \ge 7a + b = 1 \Rightarrow a \ge \frac{1}{8} \Rightarrow b = 1 - 7a \le \frac{1}{8}$$

4. $(b, 1, 2, 2)$ ：



<圖 1-5-4.1>

如<圖 1-5-4.1>所示，設均衡時 $a_4 = a_5 = 1 - a, a_2 = a_3 = 1 - 3a, a_1 = 1 - 5a,\ a < \dfrac{1}{5}$ ，並令 $c = \dfrac{1-b-5a}{2}$ ，則

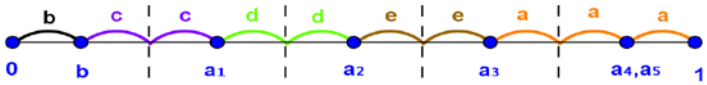
$$f_b = b + c,\ f_1 = c + a,\ f_2 = f_3 = f_4 = f_5 = a \text{ 。}$$

根據定理 1，可得 $b \le a$ 且 $c \le a$ ，因此

$$c = \frac{1-b-5a}{2} \le a \Rightarrow 7a + b \ge 1 \text{ ，}$$

則 $7a + a \ge 7a + b \ge 1$ ，故 $\dfrac{1}{8} \le a < \dfrac{1}{5}$ ，又 $b + 5b \le b + 5a < 1$ ，可以得到 $b < \dfrac{1}{6}$ 。

5. $(b, 1, 1, 1, 2)$ ：



<圖 1-5-5.1>

如<圖 1-5-5.1>，設均衡時

$a_1 = b + 2c, a_2 = b + 2c + 2d, a_3 = 1 - 3a, a_4 = a_5 = 1 - a$,並令 $e = \dfrac{1-b-3a-2c-2d}{2}$ ，則

$f_b = b + c,\ f_1 = c + d,\ f_2 = d + e,\ f_3 = e + a,\ f_4 = f_5 = a$ 。

根據定理 1 可得知：$\begin{cases} a \ge b, a \ge c, a \ge d, a \ge e, \\ c + d \ge a, c + d \ge b, c + d \ge e, \\ d + e \ge a, d + e \ge b, d + e \ge c. \end{cases}$

故

$$a \ge e = \frac{1-b-3a-2c-2d}{2} \Rightarrow 5a + b + 2c + 2d \ge 1$$

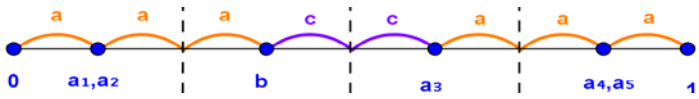
$$\therefore\ 5a + a + 2a + 2a \ge 5a + b + 2c + 2d \ge 1 \Rightarrow a \ge \frac{1}{10}$$

又

$$c+d=\frac{1-b-2d-2e-3a}{2}\geq a\Rightarrow 5a+b+2e\leq a$$

$$\therefore\ 5b+b\leq 5a+b<5a+b+2e\leq 1\Rightarrow b<\frac{1}{6}$$

6. $(2,b,1,2)$：



<圖 1-5-6.1>

如<圖 1-5-6.1>所示，根據引理三，設均衡時，

$$a_1=a_2=a, b=3a, a_3=1-3a, a_4=a_5=1-a,\ a<\frac{1}{6}\ ,$$

並令 $c=\frac{1-6a}{2}$ ，則

$$f_1=f_2=f_4=f_5=a,\ f_b=f_3=a+c\ 。$$

由定理 1，$a\geq c=\frac{1-6a}{2}\Rightarrow a\geq\frac{1}{8}$ ，又 b=3a，故 $\frac{3}{8}\leq b<\frac{1}{2}$ 。

結論：當 *n*=5 且 $b\leq\frac{1}{2}$ 時，奈許均衡存在的充分必要條件是 $0\leq b\leq\frac{1}{6}$ 或 $\frac{3}{8}\leq b\leq\frac{1}{2}$ 。

並且當 $b=\frac{1}{6}$ 或 $\frac{1}{2}$ 時，固定攤販可以得到確定獲利 $\frac{1}{6}$ ；

而當 $b=\frac{3}{8}$ 時，固定攤販可以確定得到其獲利的最大值 $\frac{1}{4}$ 。

觀察上面的結果，我們發現，固定攤販所在的位置不但可以決定奈許均衡是否存在——使沒有均衡的情況產生均衡，也可以破壞原有的均衡；還能決定均衡解的型態。並且，從 2 到 5 家流動攤販的例子中，固定攤販都可以藉由選取的位置得到保證獲利。因此我們希望能進一步得到固定攤販的保證獲利模式。

二、固定策略的保證獲利

在沙灘賣冰模型中，假設有 n 家流動攤販 $a_1, a_2,\cdots, a_n$ 以及 1 個固定攤販 b。

(1) 當 n=1 時，固定攤販必須位於 $\frac{1}{2}$ 的位置，才能產生均衡，並可獲利 $\frac{1}{2}$ 。

(2) 當 n=2 時，固定攤販若選取 $\frac{1}{4}$ 或 $\frac{3}{4}$ 的位置，就可獲得最大利益 $\frac{1}{2}$ 。

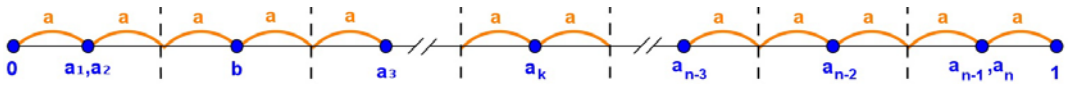
(3) 當 n=3 時，若固定攤販選取 $\frac{1}{4}$ 或 $\frac{3}{4}$ 的位置，可以得到確定獲利 $\frac{1}{4}$；若想要獲得最大利益 $\frac{1}{3}$ ，則必須選取 $\frac{1}{6}$ 或 $\frac{5}{6}$ 的位置，此時獲利會介於 $\frac{1}{6}$ 到 $\frac{1}{3}$ 之間。

(4) 當 n=4 時，固定攤販若選取 $\frac{1}{2}$ 的位置，就可獲得最大利益 $\frac{1}{3}$ 。

(5) 當 n=5 時，固定攤販若選取$\frac{3}{8}$或$\frac{5}{8}$的位置，則可獲得最大利益$\frac{1}{4}$。

(6) 當 $n\geq 6$ 時，透過觀察 n=4 和 n=5 的例子，發現固定攤販的最大保證獲利都在位於最兩側共點攤販的隔壁，也就是(2, 1, …1,2)的型態。若假設$0\leq b\leq\frac{1}{2}$且$a_1=a_2=a, b=3a$，則最大保證獲利發生在 a 有最小值的時候。因此我們猜測：固定攤販應選擇$\frac{3}{2(n-1)}$的位置。經由計算和證明，我們得到下列定理：

定理 2：當$n\geq 4$時，若固定攤販位於$\frac{3}{2(n-1)}$，則流動攤販存在唯一的奈許均衡，如<圖 2.1>所示，並且固定攤販可獲得確定利益$\frac{1}{n-1}$。



<圖 2.1>

證明：如<圖 2.1>所示，顯然

$$(a_1, a_2, b, a_3, \cdots, a_{n-1}, a_n)=(\frac{1}{2n-2}, \frac{1}{2n-2}, \frac{3}{2n-2}, \frac{5}{2n-2}, \cdots, \frac{2n-3}{2n-2}, \frac{2n-3}{2n-2})$$

會是一組奈許均衡。接下來我們要證明這會是唯一的均衡。

當$b=\frac{3}{2(n-1)}$，則均衡解可能為$a_1=a_2<b$，或$b\leq a_1$。

(1) 若$b\leq a_1$，設$a_{n-1}=a_n=1-a$，則根據引理 3，$a_{n-2}=1-3a$，

又定理 1 告訴我們：

$$a\geq b \text{ 且 } f_k\geq a,\ \ \forall 1\leq k\leq n-2,$$

則

$$f_b+\sum_{k=1}^{n} f_k\geq b+nb=\frac{3(n+1)}{2(n-1)}>\frac{3}{2},$$

顯然與$f_b+\sum_{k=1}^{n} f_k=1$矛盾。故不存在此種型態的奈許均衡。

(2) 而當$a_1=a_2<b$時，因為$b=\frac{3}{2(n-1)}$，則根據引理 3，

$$a_1=a_2=\frac{1}{2(n-1)} \text{ 且 } a_{n-1}=a_n=1-\frac{1}{2(n-1)},$$

並且

$$f_1=f_2=f_{n-1}=f_n=\frac{1}{2(n-1)}。$$

由於

$$f_b+\sum_{k=1}^{n} f_k=1\Rightarrow f_b=\begin{cases}1-\dfrac{4}{2(n-1)}-\sum_{k=3}^{n-2} f_k, & \text{當} n\geq 5,\\ 1-\dfrac{4}{6}=\dfrac{1}{3}, & \text{當} n=4.\end{cases}$$

又根據定理 1，

$$f_b\leq 2f_1\leq\frac{1}{n-1}\ \text{且}\ f_k\leq 2f_1=\frac{1}{n-1},\ \forall 3\leq k\leq n-2$$

$$\Rightarrow f_b=1-\frac{4}{2(n-1)}-\sum_{k=3}^{n-2} f_k\geq 1-\frac{2}{n-1}-\frac{(n-4)}{(n-1)}=\frac{1}{n-1}$$

故

$$f_b=\frac{1}{n-1}\text{且}\ f_k=\frac{1}{n-1},\ \forall 3\leq k\leq n-2\text{，當 } n\geq 5\text{。}$$

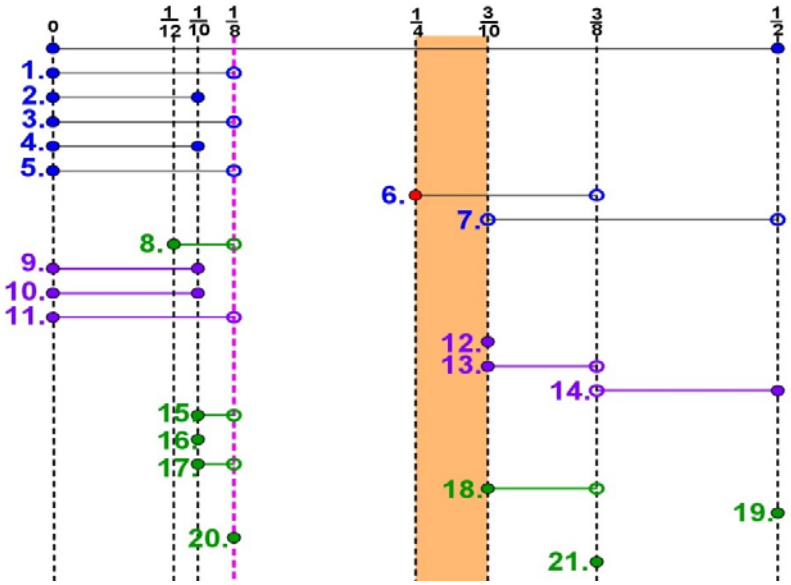
因此我們可以推論：在均衡時，在 b 左側至多只有一組共點攤販。並且

$$(a_1,a_2,b,a_3,\cdots,a_{n-1},a_n)=(\frac{1}{2n-2},\frac{1}{2n-2},\frac{3}{2n-2},\frac{5}{2n-2},\cdots,\frac{2n-3}{2n-2},\frac{2n-3}{2n-2})$$

為唯一的奈許均衡。■

為了瞭解當固定攤販位於$\frac{3}{2(n-1)}$時，所得到的保證獲利是否為最大值，我們進一步計算分析了 1 個固定攤販+ 7 家流動攤販的例子。

假設均衡時固定攤販位於 b，其餘 7 家流動攤販分別位於$a_1\leq a_2\leq\cdots\leq a_6\leq a_7$。首先，我們將所有可能形成奈許均衡的 21 種流動攤販與固定攤販的排列方式列出，求出在各排列方式中可以使均衡存在的 b 之範圍，並且畫成如下<圖 3.2>來挑選出有可能讓固定攤販擁有高於平均值利益的位置。下圖的線段便是固定攤販在各排列方式中可以使均衡存在的位置 b 之範圍。當 n =7 時，$b=\frac{3}{2\times 6}=\frac{1}{4}$時為我們提出的保證獲利策略，以紅點表示。為了方便起見，我們以(*b*, 1, 1, 1, 1, 1, 2)表示$b<a_1<a_2<a_3<a_4<a_5<a_6=a_7$，(*b*+1)則表示 b 與流動攤販共點。而根據對稱性，我們下面只討論$0\leq b\leq\frac{1}{2}$的情形。



<圖 2.2>

1. (b, 1, 1, 1, 1, 1, 2)，範圍$b<\frac{1}{8}$。

2. (b, 2, 1, 1, 1, 2)，範圍$b\le\frac{1}{10}$。

3. (b, 1, 2, 1, 1, 2)，範圍$b<\frac{1}{8}$。

4. (b, 1, 1, 2, 1, 2)，範圍$b\le\frac{1}{10}$。

5. (b, 1, 1, 1, 2, 2)，範圍$b<\frac{1}{8}$。

6. (2, b, 1, 1, 1, 2)，範圍$\frac{1}{4}\le b<\frac{3}{8}$。

7. (2, 1, b, 1, 1, 2)，範圍$\frac{3}{10}<b<\frac{1}{2}$。

8. (b+1, 1, 1, 1, 1, 2)，範圍$\frac{1}{12}\le b<\frac{1}{8}$。

9. (b, 2, 2, 1, 2)，範圍$b\le\frac{1}{10}$。

10.(b, 2, 1, 2, 2)，範圍$b\le\frac{1}{10}$。

11.(b, 1, 2, 2, 2)，範圍$b<\frac{1}{8}$。

12.(2, b, 2, 1, 2)，範圍$b=\frac{3}{10}$。

13.(2, b, 1, 2, 2)，範圍$\frac{3}{10}\le b<\frac{3}{8}$。

14.(2, 1, b, 2, 2)，範圍$\frac{3}{8}<b\le\frac{1}{2}$。

15.(b+1, 2, 1, 1, 2)，範圍$\frac{1}{10}\le b<\frac{1}{8}$。

16.(b+1, 1, 2, 1, 2)，範圍$b=\frac{1}{10}$。

17.(b+1, 1, 1, 2, 2)，範圍$\frac{1}{10}\le b<\frac{1}{8}$。

18.(2, b+1, 1, 1, 2)，範圍$\frac{3}{10}\le b<\frac{3}{8}$。

19.(2, 1, b+1, 1, 2)，範圍$b=\frac{1}{8}$。

20.(b+1, 2, 2, 2)，範圍$b=\frac{1}{2}$。

21.(2, b+1, 2, 2)，範圍$b=\frac{3}{8}$。

如上圖可見，只有粉色線及橘色區域的均衡解唯一，因此我們可知道擁有唯一均衡解的 b 的位置僅有(2, b+1, 2 ,2)的$b=\frac{3}{8}$及(2, b, 1, 1, 1, 2)的$\frac{1}{4}\le b<\frac{3}{10}$，但由先前的結果得知共點攤販的利益必不大於獨立攤販的利益，所以當 b 共點時(上圖綠色線段及點)所獲得的利益必定不會高於平均值，因此 b 位於粉色線時不會有高於平均值的利益。而橘色線段則是在我們在先前的報告提出的保證獲利，而在此種流動攤販與固定攤販的排列方式中，最大獲利發生在 a 有最小值的時候，因此最大獲利發生在$b=\frac{1}{4}$，且 $f_b=\frac{1}{n-1}=\frac{1}{7-1}=\frac{1}{6}$。由上述的結果得知，在唯一解的狀況下，最大獲利與我們的先前假設相符。

接著我們觀察若 b 的位置可以形成一種以上的均衡，是否存在所有解的獲利都比我們提出的保證策略還要好的例子？我們先將與綠色線段重疊的部分及先前所探討的唯一解狀況屏除後，發現剩下(2, 1, b, 1, 1, 2)及(2, 1, b, 2, 2)共同存在的$\frac{3}{8}<b<\frac{1}{2}$。但在(2, 1, b, 1, 1, 2)的情形中，b 位於兩個獨立的流動攤販旁邊，因此非與固定攤販相鄰之流動攤販若向 b 靠近亦可形成均衡，使得 b 獲得的利益會小於平均值，因此此解便不保證可以獲得高於平均的利益。

更進一步觀察，我們可以發現可以形成均衡(2,b,1,1,1,2)與(2,1,b,1,1,2)之b的範圍有重疊，因此我們猜測：當$n\ge 6$時，形成均衡$(2,b,1,\cdots,1,2)$, $(2,1,b,1,\cdots,1,2)$, $(2,1,1,b,\cdots,1,2)$, …, $(2,1,1,\cdots,1,b,2)$之 b 的範圍會涵蓋所有可能產生更高獲利的位置。因此，我們藉著下列引理和定理三，證明出我們在定理二所提出的公式會是固定攤販所能得到的保證獲利之最大值。

引理 4：當 $n\geq 6$ 時，對任意 $\frac{3}{2n-4}\leq b\leq 1-\frac{3}{2n-4}$ ，

$a_1=a_2=\frac{1}{2n-4}, a_k=\frac{2k-3}{2n-4}\left(3\leq k\leq n-2\right), a_{n-1}=a_n=\frac{2n-5}{2n-4}$ **會是一組奈許均衡。**

(證明詳見附錄一)

引理 5：當 $n\geq 3$ 時，若 n 為奇數，則對任意 $0\leq b\leq \frac{1}{n+1}$ ，

$a_1=\frac{1}{n+1}, a_{2k}=a_{2k+1}=\frac{2k+1}{n+1}, \forall 1\leq k\leq \left[\frac{n}{2}\right]$ **會是一組奈許均衡；**

若 n 為偶數，則對任意 $0\leq b\leq \frac{1}{n+2}$ ，

$a_1=\frac{1}{n+2}, a_2=\frac{3}{n+2}, a_{2k-1}=a_{2k}=\frac{2k+1}{n+2}, \forall 2\leq k\leq \left[\frac{n}{2}\right]$ **會是一組奈許均衡。**

(證明詳見附錄一)

引理 6：當 $n\geq 5$ 時，對任意 $\frac{3}{2n-2}\leq b<\frac{3}{2n-4}$ ，

$a_1=a_2=\frac{b}{3}, a_k=1-\frac{(2n-2k-1)b}{3}\left(3\leq k\leq n-2\right), a_{n-1}=a_n=1-\frac{b}{3}$ **會是一組奈許均衡。**

(證明詳見附錄一

引理 7：(i) 在均衡時，設共點攤販所分得的利益為 a ，若兩組共點攤販間僅有奇數家獨立的流動攤販位於 $a_{m+1}<a_{m+2}<\cdots<a_{m+2k+1}$ ，則此 2k+1 家流動攤販的利益總和

$$\sum_{j=1}^{2k+1} f_{m+j}\geq (2k+2)a$$

(ii) 在均衡時，若固定攤販位於最左側 b 的位置，而他與最近的共點攤販之間若有偶數家獨立攤販位於 $a_1<a_2<\cdots<a_{2k}$ ，則此 2k 家流動攤販與固定攤販的利益總和

$$f_b+\sum_{j=1}^{2k} f_j\geq (2k+2)b$$

(證明詳見附錄一)

定理 3：當 $n\geq 4$ 時，若 n 為奇數，則當固定攤販位於 $\frac{1}{n+1}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡；

若 n 為偶數，則當固定攤販位於 $\frac{1}{n+2}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡。

證明：

1. 當 n 為奇數，依照均衡時的型態可分為 $b \le a_1$ 或 $a_1 = a_2 < b$ ，兩種情況：

   若 $b \le a_1$ ，設各共點攤販所分得的利益為 a ，由於獨立攤販的利益不小於共點攤販，並且 $a \ge b > \frac{1}{n+1}$ ，故攤販利益總和 $f_b + \sum_{i=1}^{n} f_i \ge b + na \ge (n+1)b > 1$ ，不合。

   若 $a_1 = a_2 < b$，設共點攤販的利益 $= f_1 = f_2 = f_{n-1} = f_n = a$，則 $a \le \frac{b}{3} < \frac{1}{2n-2}$，且 $0 \le f_b \le 2a$ ，同時獨立攤販的利益會介於 $[a, 2a]$ 之間，故所有攤販的利益總和

   $f_1 + f_2 + f_b + \sum_{i=3}^{n-2} f_i + f_{n-1} + f_n \le 4a + (n-4) \cdot 2a + 2a = (2n-2)a < 1$ ，也不合。

   故當 n 為奇數，且固定攤販位於 $\frac{1}{n+1} < b < \frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡。

2. 當 n 為偶數，我們同樣依照均衡時的型態分為 $b \le a_1$ 或 $a_1 = a_2 < b$ ，兩種情況：

   (1) $a_1 = a_2 < b$

   由於 $b < \frac{3}{2n-2}$ ，同 1. 的討論，我們可以知道無法形成 $a_1 = a_2 < b$ 型態的均衡。

   而當 $b \le a_1$ ，我們可以再分成 $b = a_1$ 即 $(b+1, \cdots, 2)$ ，和 $b < a_1 = a_2$ 即 $(b, 2, \cdots, 2)$ 與 $b < a_1 < a_2$ 即 $(b, 1, \cdots, 2)$ 三種。

   (2) $b = a_1$ ：

   此時共點攤販的利益 $a = b > \frac{1}{n+2}$ 。因為 n 是偶數，故均衡時必有奇數家獨立攤販，則必存在奇數家獨立攤販介於兩組共點攤販間，由引理 7 (i)可知，則此 $2k+1$ 家流動攤販的利益總和 $\sum_{j=1}^{2k+1} f_{m+j} \ge (2k+2)a$ ，而其餘 $n-(2k+1)$ 家流動攤販與固定攤販的利益均不小於共點攤販，故所有攤販的利益總和

   $$1 = f_b + \sum_{i=1}^{n} f_i \ge a + (2k+2)a + [n-(2k+1)]a = (n+2)a > 1$$ ，不合。

   (3) 若 $b < a_1 = a_2$ 即 $(b, 2, \cdots, 2)$ ：

   此時固定攤販的利益 $f_b = 2b$，而其餘 n 家流動攤販與固定攤販的利益均不小於共點攤販的利益 $a = f_1 = f_2 = b$ ，故所有攤販的利益總和

   $$1 = f_b + \sum_{i=1}^{n} f_i \ge 2b + nb = (n+2)b > 1$$ ，亦不合。

   (4) 若 $b < a_1 < a_2$ 即 $(b, 1, \cdots, 2)$ ：

   此時共點攤販的利益 $a \ge b > \frac{1}{n+2}$ 。因為 n 是偶數，故均衡時必有偶數家獨立攤販，則根據引理 7 (ii)，此 $2k$ 家流動攤販與固定攤販的利益總和 $f_b + \sum_{j=1}^{2k} f_j \ge (2k+2)b$ ，而其餘 n-$2k$ 家流動攤販的利益均不小於共點攤販，故所有攤販的利益總和

   $$1 = f_b + \sum_{i=1}^{n} f_i \ge (2k+2)b + (n-2k)a \ge (n+2)b > 1$$ ，也不合。

   若有奇數家獨立攤販位於固定攤販與其最近的共點攤販之間，則必有奇數家獨立攤販位

於兩組共點攤販之間，根據引理7 (i)，此$2k+1$家流動攤販的利益總和$\sum_{j=1}^{2k+1} f_{m+j} \geq (2k+2)a$，而其餘$n-(2k+1)$家流動攤販與固定攤販的利益均不小於共點攤販，故所有攤販的利益總和

$1 = f_b + \sum_{i=1}^{n} f_i \geq a + (2k+2)a + [n-(2k+1)]a = (n+2)a > 1$，同樣不合。

由上述討論 1、2、2-(1)～2-(4)可以得知：

當n為奇數且固定攤販位於$\frac{1}{n+1} < b < \frac{3}{2n-2}$，或$n$為偶數且固定攤販位於$\frac{1}{n+2} < b < \frac{3}{2n-2}$時，其餘$n$家流動攤販均無法形成奈許均衡。■

由引理四～七及定理三可知：當$0 \leq b \leq \frac{1}{2}$的範圍中，除了$b = \frac{3}{2n-2}$的範圍之外，其餘位置若可以使均衡存在，都會存在某組均衡，使得固定攤販的利益$f_b < \frac{1}{n-1}$。也就是說，當固定攤販位於$b = \frac{3}{2n-2}$，有最大保證獲利$f_b = \frac{1}{n-1}$。

定理 4：當 $n \geq 4$ 時，若固定攤販位於$\frac{3}{2(n-1)}$，則流動攤販存在唯一的奈許均衡，如<圖 3.1>所示，此時固定攤販可獲得確定利益$\frac{1}{n-1}$，並且其為最大保證獲利(*Max Minimum*)。

三、增加固定攤販數量對奈許均衡的影響

由定理四我們可以得知，當模型中有一家固定攤販時，若其採取保證獲利策略，便可獲得大於平均的利益，那麼其他流動攤販是否也會想要變成固定攤販以獲得更大利益呢？因此我們試著增加固定攤販的數量，假設固定攤販是依序進入沙灘，觀察其進入沙灘的順序是否會影響均衡的型態，使其獲得更大利益。

若固定攤販輪流站定，則根據定理四所提出的保證獲利策略，當模型中有n家流動攤販時，第一家固定攤販b_1必然會採取保證獲利策略，選擇站於$\frac{3}{2(n\text{-}1)}$以獲取較多的利益$\frac{1}{n-1}$。此時，必有一組共點攤販位於b_1的左側，也就是$\frac{1}{2(n\text{-}1)}$的位置，並且可以獲得$\frac{1}{2(n\text{-}1)}$的利益。由於其餘攤販的利益不大於共點攤販的兩倍，因此其餘的攤販利益最大值即為$\frac{1}{n-1}$，所以其餘的固定攤販不可能獲得大於$\frac{1}{n-1}$的利益。而固定攤販若要獲得$\frac{1}{n-1}$的利益，那麼就必須站在**《定理四》**所提出的均衡點上。

四、固定策略玩家對平面模型的影響

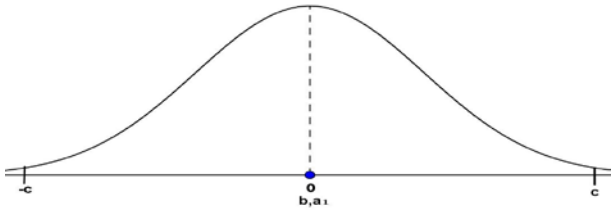
最後我們將模型推廣到平面，在我們的平面模型中，假設攤販只能在一條直線沙灘上移動，而遊客在沙灘上的分布是不均勻的，沙灘上每一個位置對應到的遊客數量皆不同。我們在接下來的研究中，分別討論了以常態分布及M型化分布作為沙灘上遊客的分布情況。而平面模型中遊客及攤販的移動方式與沙攤賣冰模型的假設相同。在遊客非均勻分布的模型中，我們假設沙灘為線段$[-c, c]$，而遊客分布的機率密度函數為$g(x)$，有一家固定攤販位於

$-c \le b \le 0$，其餘 n 家流動攤販達成均衡時位於 $a_1, a_2, \cdots, a_n$，其中 $-c \le a_1 \le a_2 \le \cdots \le a_n \le c$，而其所獲得的利益分別為 f_b 及 $f_1, \cdots, f_n$，並且 $f_b + f_1 + \cdots + f_n = \int_{-c}^{c} g(x)dx = 1$。

(一) 常態分布

1. 1 家固定攤販+1 家流動攤販

當 $b \neq 0$ 時，流動攤販會在固定攤販旁邊空間較長的一側並向他靠近，但不會共點，故此時沒有奈許均衡。因此，均衡產生在 $b = a_1 = 0$，如<圖 4-1.1>。



<圖 4-1-1.1>

2. 1 家固定攤販+2 家流動攤販

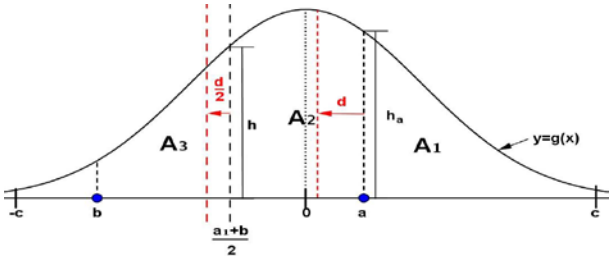
根據定理一，兩家流動攤販必共點，因此假設 $a_1 = a_2 = a, 0 \le a \le c$，

令面積 $A_1 = \int_a^c g(x)dx, A_2 = \int_{\frac{b+a}{2}}^{a} g(x)dx, A_3 = \int_{-c}^{\frac{b+a}{2}} g(x)dx$，高度 $h = g\left(\frac{b+a}{2}\right), h_a = g(a)$，如<圖 4-1-2.1>所示，則流動攤販共點的條件為 $A_1 = A_2$ 且 $h_a \ge h$。

證明： $h_a \ge h$

$$\because g\left(\frac{b+a}{2}\right) \le g(a) \quad \therefore g\left(\frac{b+a-d}{2}\right) < g\left(\frac{b+a}{2}\right) \le g(a) < g(a-d)$$

$\therefore a$ 不論往左或往右獲利皆會變小，因此可形成均衡■



<圖 4-1-2.1>

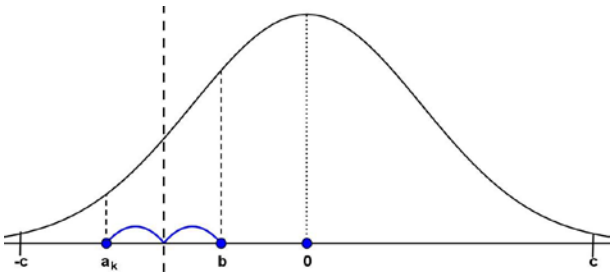
根據以上條件，當形成均衡時，固定攤販可獲得的最大利益發生在 h_a=h 時，並且其利益的最大值為 f_b=A_3=A_1=A_2 = $\frac{1}{3}$。

定理 5：平面模型中，假設流動攤販所在位置對應到的遊客密度為 h_1，而其與相鄰攤販利益平分線的遊客密度為 h_2，則流動攤販的共點條件為：$h_1 \ge h_2$ 且 A_1=A_2

3. 1 個固定攤販+n 個流動攤販：

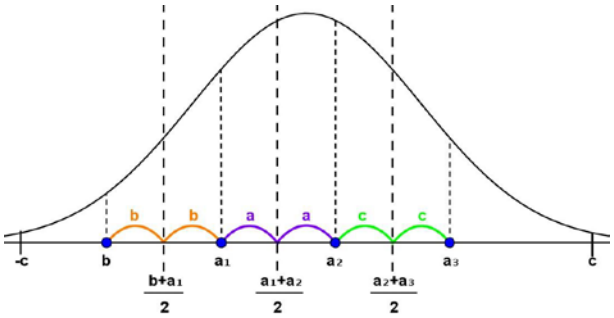
(1) b 左側攤販數：

若 b<0，根據定理五，則 b 左側不會有流動攤販。



<圖 4-1-3.1>

(2) b 右側攤販數：

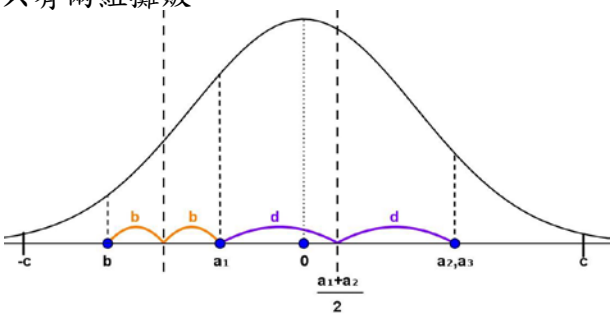


<圖 4-1-3.2>

假設 b 右側有不只一組共點攤販，如<圖 4-1-3.2>

若 $g\left(\frac{b+a_1}{2}\right)<g\left(\frac{a_1+a_2}{2}\right)\Rightarrow$ 則 a_k 會往右；若 $g\left(\frac{b+a_1}{2}\right)>g\left(\frac{a_1+a_2}{2}\right)\Rightarrow$ 則 a_k 會往左。

$\therefore g\left(\frac{b+a_1}{2}\right)=g\left(\frac{a_1+a_2}{2}\right)$，同理 $g\left(\frac{a_1+a_2}{2}\right)=g\left(\frac{a_2+a_3}{2}\right)$，但在此模型中，至多只有兩條線等高，因此最多只有兩組攤販。



<圖 4-1-3.3>

若 b 右側有兩組共點攤販，則根據定理一，端點攤販需為共點攤販，設 $b<a_1<a_2=a_3$

又 $g\left(\frac{b+a_1}{2}\right)=g\left(\frac{a_1+a_2}{2}\right)\therefore\frac{b+a_1}{2}<0<\frac{a_1+a_2}{2}\Rightarrow g\left(\frac{a_1+a_2}{2}\right)>g\left(a_2\right)$

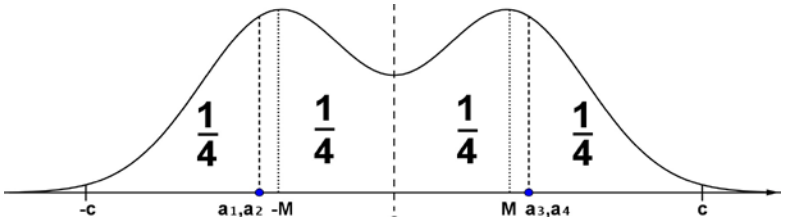
$\therefore$ a_2、a_3 會往左移。

因此當 b 右側有三家以上的攤販時，不可能形成均衡。

(二) M 型化分布

在 M 型化模型中，假設模型是由兩個具有相同標準差的常態分佈模型疊加而成，波峰分別位於 $-M$ 及 M，波谷位於 0。根據文獻，我們知道在沒有固定攤販的情形中，一家攤販可在沙灘上任一點形成均衡；兩家攤販會於沙灘中央形成均衡；三家攤販沒有均衡；四

家攤販會兩兩共點於a_1=a□a_3=a_4，其中$\int_{-c}^{a_1} g(x)dx = \int_{a_1}^{0} g(x)dx = \int_{0}^{a_3} g(x)dx = \int_{a_3}^{c} g(x)dx = \frac{1}{4}$，如<圖 4-2>所示；四家攤販以上不可能形成均衡。因此，我們同樣在模型中加入固定攤販以增加均衡的可能，並觀察其獲利與流動攤販獲利的關係。



<圖 4-2>

根據我們先前的研究，我們可以知道攤販形成均衡的必要條件除了：

1.任一攤販的利益不大於其他攤販的兩倍，

2.共點攤販的利益面積左右相等。

之外，還需要考慮沙灘上每一點對應到的遊客密度。因此，我們發現：

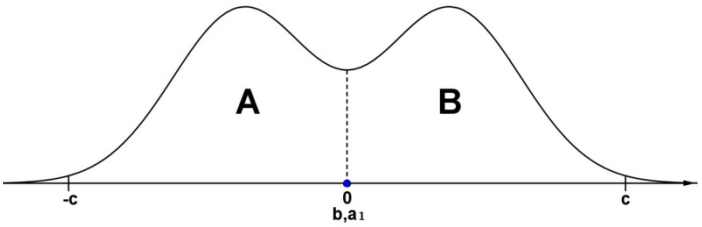
(1) 若 $g\left(\frac{a_{k-1}+a_k}{2}\right) < g\left(\frac{a_k+a_{k+1}}{2}\right)$

或 $g\left(\frac{a_{k-1}+a_k}{2}\right) = g\left(\frac{a_k+a_{k+1}}{2}\right)$ 且 $g'\left(\frac{a_{k-1}+a_k}{2}\right) < g'\left(\frac{a_k+a_{k+1}}{2}\right)$，則流動攤販 a_k 會右移。

(2) 若 $g\left(\frac{a_{k-1}+a_k}{2}\right) > g\left(\frac{a_k+a_{k+1}}{2}\right)$

或 $g\left(\frac{a_{k-1}+a_k}{2}\right) = g\left(\frac{a_k+a_{k+1}}{2}\right)$ 且 $g'\left(\frac{a_{k-1}+a_k}{2}\right) > g'\left(\frac{a_k+a_{k+1}}{2}\right)$，則流動攤販 a_k 會左移。

1. 1 家固定攤販+1 家流動攤販：

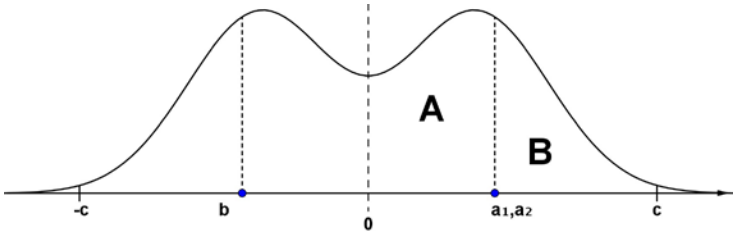


<圖 4-2-1>

均衡時，$b = a_1 = 0$，$f_b = f_{a_1} = \frac{1}{2}$。

2. 1 家固定攤販+2 家流動攤販

當只有兩家流動攤販時，顯然(*b*,2)會是唯一的均衡型態。



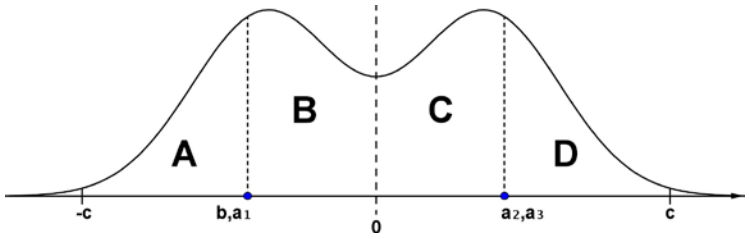
<圖 4-2-2>

若固定攤販位於 b，並滿足 $\int_{-c}^{b} g(x)dx = \int_{b}^{0} g(x)dx = \frac{1}{4}$ 且 $g(b) \geq g(0)$，則均衡時，流動必共點於 $a_1 = a_2 = -b$。此時，固定攤販可獲得最大利益 $\frac{1}{2}$。

3. 1 家固定攤販+3 家流動攤販

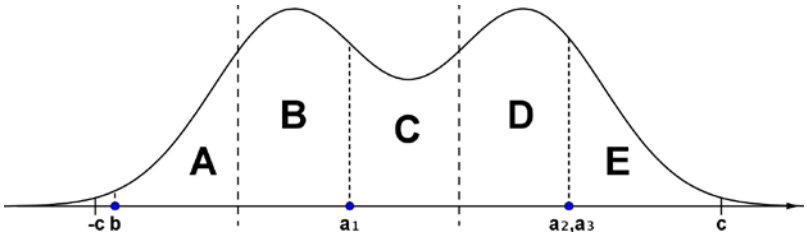
當只三家流動攤販時，有兩種可能的的均衡型態：

(1) (b+1,2)



<圖 4-2-3>

(2) 均衡時，$A = B = C = D = \frac{1}{4}$，$f_b = \frac{1}{4}$。($b$,1,2)



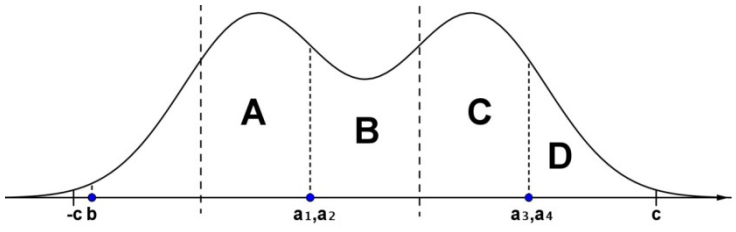
<圖 4-2-4>

假設 $\int_{\frac{b+a_1}{2}}^{-c} g(x)dx = A$、$\int_{a_1}^{\frac{b+a_1}{2}} g(x)dx = B$、$\int_{\frac{a_1+a_2}{2}}^{a_1} g(x)dx = C$、$\int_{a_2}^{\frac{a_1+a_2}{2}} g(x)dx = D$、$\int_{C}^{a_3} g(x)dx = E$，其中 $b < a_1 < a_2 = a_3$。由於均衡時，必須滿足 $g\left(\frac{b+a_1}{2}\right) = g\left(\frac{a_1+a_2}{2}\right)$、$g'\left(\frac{b+a_1}{2}\right) = g'\left(\frac{a_1+a_2}{2}\right)$ 且 $g\left(\frac{a_1+a_2}{2}\right) \leq g(a_2) = g(a_3)$，故 $A \leq D = E$，又任一攤販所獲得利益不小於共點攤販，因此 $D \leq B + C$。則得知 $A \leq D = E \leq B + C$，因此 $A \leq \frac{1}{4}$。

由以上討論，我們得知：3 家流動攤販時，固定攤販的利益最大值發生在(b+1,2)。

4. 1 家固定攤販+4 家流動攤販

(1) (b,2,2)

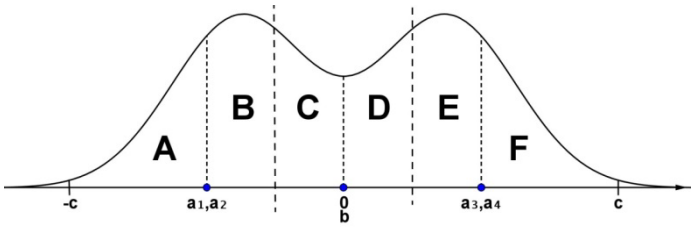


<圖 4-2-5>

假設 $\int_{a_1}^{\frac{b+a_1}{2}} g(x)dx = A$、$\int_{\frac{a_1+a_3}{2}}^{a_1} g(x)dx = B$、$\int_{a_3}^{\frac{a_1+a_3}{2}} g(x)dx = C$、$\int_{-C}^{a_3} g(x)dx = D$。由於

均衡時，$g\left(\frac{b+a_1}{2}\right)\le g(a_1)$、$g(a_1)\ge g\left(\frac{a_1+a_3}{2}\right)$、$g\left(\frac{a_1+a_3}{2}\right)\le g(a_3)$，且$A=B=C=D<\frac{1}{4}$，則必有一組共點攤販位於$a_3=a_4\in(M,c)$，並且$\frac{a_1+a_3}{2}\in(0,M)$，又$\left|\frac{a_1+a_3}{2}-a_1\right|=\left|a_3-\frac{a_1+a_3}{2}\right|$，顯然$B<C=D$，不符合均衡條件，因此此情況無法形成均衡。

(2) (2,*b*,2)



<圖 4-2-6>

若均衡時流動攤販倆倆共點於$a_1=a_2$、$a_3=a_4$，則必須滿足$g(a_1)\ge g\left(\frac{a_1+b}{2}\right)$、$g(a_3)\ge g\left(\frac{b+a_3}{2}\right)$以及$\int_{-c}^{a_1}g(x)dx=\int_{a_1}^{\frac{a_1+b}{2}}g(x)dx=\int_{\frac{b+a_3}{2}}^{a_3}g(x)dx=\int_{a_3}^{c}g(x)dx$，故$a_1=-a_3$，並且固定攤販位於 *b*=0。此時固定攤販的獲利 $f_b=1-4\int_{a_3}^{c}g(x)dx$。當 $g(0)=g(c)$ 時，固定攤販可以獲得最大利益 $f_b=\frac{1}{3}$

(3) (*b*+1,1,2)

由上述(2,*b*,2)的討論得知，在此情形中，獨立攤販必位於$a_2=0$，因此必須滿足$g\left(\frac{a_1}{2}\right)=g\left(\frac{a_3}{2}\right)$、$g'\left(\frac{a_1}{2}\right)=g'\left(\frac{a_3}{2}\right)$且$\int_{-c}^{a_1}g(x)dx=\int_{a_1}^{\frac{a_1}{2}}g(x)dx=\int_{\frac{a_3}{2}}^{a_3}g(x)dx=\int_{a_3}^{c}g(x)dx$。但根據對稱性，顯然$g'\left(\frac{a_1}{2}\right)\ne g'\left(\frac{a_3}{2}\right)$，因此不可能形成均衡。

(4) (*b*,1,1,2)

根據產生均衡之必要條件，流動攤販獨立時左右兩側利益分界線高度必相等，而在 *M* 型分布下，共會有四條線等高，但此時$g'\left(\frac{b+a_1}{2}\right)\ge g'\left(\frac{a_1+a_2}{2}\right)$且$g'\left(\frac{a_1+a_2}{2}\right)<g'\left(\frac{a_2+a_3}{2}\right)$，因此流動攤販無法形成均衡。

根據以上討論，在四家流動攤販的情形中，只有(2,*b*,2)可能產生均衡，且 $f_b\le\frac{1}{3}$。而從討論我們知道，在 *M* 型化模型中，由於只有四條線等高，因此均衡時最多只會有兩組流動攤販與一個固定攤販，且共點攤販不只能位於端點，因此四家以上的流動攤販皆無法形成均衡。

五、程式模擬

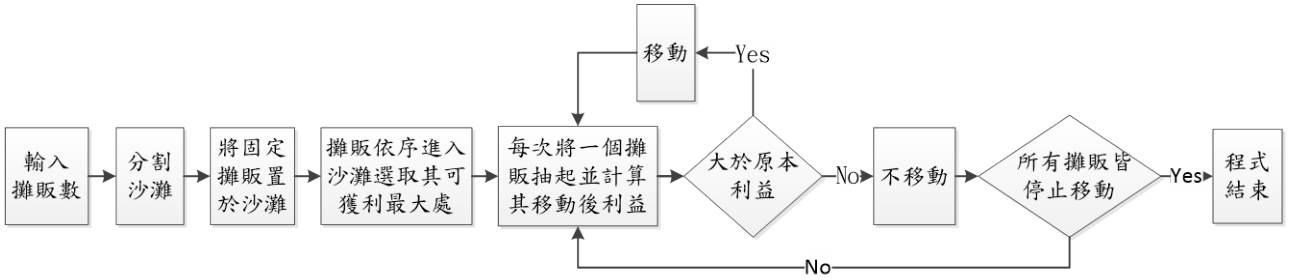
為了證明我們的模型在現實生活中的應用價值，我們以程式模擬模型，觀察有固定攤販時，是否可以達成公式所述之均衡態，並觀察不同攤販數下達成均衡的速度。我們也模擬了沒有固定攤販的模型作為對照組，比較兩個模型當攤販總數相同時，攤販達成均衡的速度，探討固定攤販對均衡的影響，觀察其是否助於達成均衡。

我們的程式已可以成功演算 n 家攤販的均衡情形，但由於當攤販數過多時，測試時間會過長，因此我們選用攤販總數 5~9 來模擬。

以下流程圖為程式之架構，完整程式碼詳見附錄二。

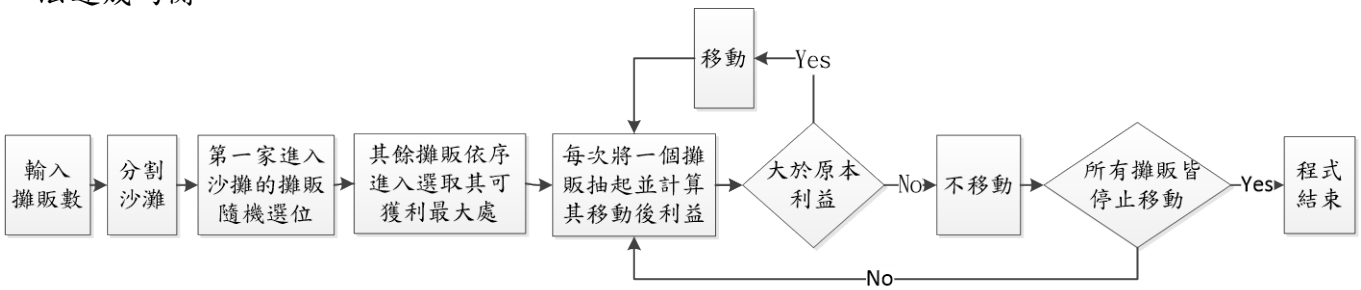
(一)有固定攤販之模型

在此模型中，有一家固定攤販，而我們測試了流動攤販數 n=4~9 時沙灘的均衡狀態，每一種攤販數接測試二十次，而在程式中，每家流動攤販都重新決定位置稱為一回，我們記錄了二十次中達成均衡所需的移動回數，找出最大值及最小值，並計算平均移動回數。在與無固定攤販之對照組比較時，是以沙灘上攤販的總數為基準，比較兩者達成均衡之速度。為了使固定攤販可以位於公式所提出之點上，我們依據公式將沙灘分割為約 100 格。



(二)無固定攤販之模型

在此模型中，沒有固定攤販，我們將沙灘分割為 100 格，並測試了流動攤販數 n=4~10 時沙灘的均衡狀態，每一種攤販數接測試十次，我們記錄了十次中達成均衡所需的移動回數，找出最大值及最小值，並計算平均移動回數，若當移動回數超過 700000 時，我們即視此次無法達成均衡。



伍、 研究結果

一、固定攤販對奈許均衡的影響

1. 達成均衡時，在任何一個位置上，最多只會有兩個攤販共點，並且共點攤販左右兩側分得的區域相等，而最靠近兩端點的攤販必定兩兩共點。

2. $n=2$ 時且 $b \le \frac{1}{2}$ 時，均衡產生在 $(b, a_1, a_2) = \left(b, \frac{b+2}{3}, \frac{b+2}{3}\right)$

並且，三家攤販所分得的利益分別為 $f_b = \frac{1-2b}{3}$ ， $f_1 = f_2 = \frac{1-b}{3}$ 。

3. 當 $n=3$ 且 $b\le\frac{1}{2}$ 時，均衡產生在 $(\frac{1}{4},\frac{1}{4},\frac{3}{4},\frac{3}{4})$ 或 $(b,a_1,a_2,a_3)=(b,1-3a,1-a,1-a)$，其中 $0\le b<\frac{1}{4},\frac{1}{6}\le a<\frac{1}{3}$ 且 $b\le a$。

並且，四家攤販所分得的利益分別為：當 $b=\frac{1}{4}\vee\frac{3}{4}$, $f_b=f_1=f_2=f_3=\frac{1}{4}$

或 $0\le b<\frac{1}{4},\frac{1}{6}\le a<\frac{1}{3}$ 且 $b\le a$ ， $f_b=\frac{b+1-3a}{2}$, $f_1=\frac{1-b-a}{2}$, $f_3=f_4=a$ 。

4. 當 $n=4$ 且 $b\le\frac{1}{2}$ 時，奈許均衡存在的充分必要條件是 $b=\frac{1}{2}$ 或 $0\le b\le\frac{1}{6}$ 。

5. 當 $n=5$ 且 $b\le\frac{1}{2}$ 時，奈許均衡存在的充分必要條件是 $0\le b\le\frac{1}{6}$ 或 $\frac{3}{8}\le b\le\frac{1}{2}$ 。

二、固定策略的保證獲利

1. 當 $n=1$ 時，固定攤販必須位於 $\frac{1}{2}$ 的位置，才能產生均衡，並可獲利 $\frac{1}{2}$ 。

2. 當 $n=2$ 時，固定攤販若選取 $\frac{1}{4}$ 或 $\frac{3}{4}$ 的位置，就可獲得最大利益 $\frac{1}{2}$ 。

3. 當 $n=3$ 時，若固定攤販選取 $\frac{1}{4}$ 或 $\frac{3}{4}$ 的位置，可以得到確定獲利 $\frac{1}{4}$ ；

若想要獲得最大利益 $\frac{1}{3}$ ，則必須選取 $\frac{1}{6}$ 或 $\frac{5}{6}$ 的位置，但不保證獲利為 $\frac{1}{3}$ 。

4.當 $n=4$ 時，若 n 為奇數，則當固定攤販位於 $\frac{1}{n+1}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡；若 n 為偶數，則當固定攤販位於 $\frac{1}{n+2}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡。

5.當 $n\ge4$ 時，若固定攤販位於 $\frac{3}{2(n-1)}$ ，則流動攤販存在唯一的奈許均衡，此時固定攤販可獲得確定利益 $\frac{1}{n-1}$ ，並且其為最大保證獲利。

三、增加固定攤販數量對奈許均衡的影響

當固定攤販不只一家時，我們假設他們會輪流站定，則第一家固定攤販必然會採取保證獲利策略。而第二家或第二家以上的固定攤販必須站定於**《定理四》**所提出的均衡點，才能在其他人達成均衡時，獲得最大的利益。

四、固定策略玩家對平面模型的影響

(一)常態分布

1.當 $n=1$ 時，固定攤販必須位於 0 的位置，才能產生均衡，並可獲利 $\frac{1}{2}$ 。

2.當 $n=2$ 時，固定攤販可獲得的最大利益為 $\frac{1}{3}$ 。

3.在常態分布且有一家固定攤販的模型中，達成均衡時，最多只會有兩家流動攤販。

4.平面模型中，假設流動攤販所在位置對應到的遊客人數為 h_1 ，而其與其他攤販利益平分線的遊客人數為 h_2 ，則流動攤販的共點條件為： $h_1\ge h_2$

(二)*M* 型化分佈

1.在 *M* 型化模型中，由於只有四條線等高，因此均衡時最多只會有兩組流動攤販與一個固定攤販，且共點攤販不只能位於端點，因此四家以上的流動攤販皆無法形成均衡。

2.當 $n=1$ 時，產生均衡的條件為 $b=a_1=0$ ，則 $f_b=f_{a_1}=\frac{1}{2}$ 。

3.當 $n=2$ 時，$(b,2)$ 為唯一均衡型態，而 $\int_{-c}^{b} g(x)dx=\int_{b}^{0} g(x)dx=\frac{1}{4}$ 且

$g(b)\geq g(0)$ ，固定攤販可獲得最大利益 $\frac{1}{2}$ 。

4.當 $n=3$ 時，固定攤販的利益最大值發生在 $(b+1,2)$ 的情況，且 $f_b=\frac{1}{4}$ 。

5.當 $n=4$ 時，唯 $(2,b,2)$ 可以有均衡產生。產生均衡時，固定攤販滿足

$\int_{-c}^{b} g(x)dx=\int_{b}^{0} g(x)dx=\frac{1}{4}$ 且 $g\left(\frac{a_1}{2}\right)=g\left(\frac{a_3}{2}\right)\geq g\left(\frac{a_1+b}{2}\right)$ ，此時 $b=0$ ， $f_b=\frac{1}{3}$ 。

五、程式模擬

(一)有固定攤販之模型

在<表 5-1>中，*n* 為流動攤販數，並記錄二十次實驗中，達成均衡所需移動回數。

攤販總數	5	6	7	8	9	10
n	4	5	6	7	8	9
第 1 次	18	946	3308	54567	7999	124029
第 2 次	18	2278	12451	881	19238	153990
第 3 次	18	3099	23224	18043	150665	65584
第 4 次	18	437	13677	44473	3749	14000
第 5 次	18	646	1123	15300	11744	94561
第 6 次	18	1397	18845	7914	9301	1504
第 7 次	18	3685	3622	23603	93498	15405
第 8 次	18	2414	1535	807	35960	200884
第 9 次	18	324	3560	24289	19364	95935
第 10 次	18	1226	5383	2191	10313	541580
第 11 次	18	1349	1305	74857	35885	75961
第 12 次	18	454	7473	116960	205131	50568
第 13 次	18	2380	18704	8617	71601	75823
第 14 次	18	1447	6520	13288	53903	8740
第 15 次	18	575	2764	1946	29222	10017
第 16 次	18	443	6580	36584	6209	18380
第 17 次	18	3206	10440	60740	4395	46398
第 18 次	18	873	11969	8150	128505	53221
第 19 次	18	524	10719	789	94668	105889
第 20 次	18	1989	28925	5365	51820	11575
Max	18	3685	28925	116960	205131	541580
Min	18	324	1123	807	3749	1504
Average	18	1484	9606	27182	52158	88202

<表 5-1>

1.透過程式的模擬，我們得到當固定攤販位於定理四所選定的位置時，其餘流動攤販均可達成均衡，且流動攤販達成的均衡態與固定攤販獲得的利益與定理四之預測相符。

2.從<表 5-1>中，可以由平均移動次數觀察到隨著流動攤販數量的增加，達成均衡的速度變慢、所需的移動回數增加。

3. 當流動攤販數量增加時，各攤販數達成均衡所需次數的最大、最小值差距極大是因為程式中，當兩個以上的位置利益相等時，攤販會隨機選取其中一個位置所造成的差異。

(二)無固定攤販之模型

在<表 5-2>中，*n* 為流動攤販數，並記錄十次實驗中，達成均衡所需移動回數。

均衡時攤販位置結果詳見附錄三。

攤販總數	4	5	6	7	8	9	10
第一次	16	4387	18	700217 無均衡,集中	15	711254 無均衡,集中	16
第二次	15	734104 無均衡,集中	535	521	48	222	733257 無均衡,集中
第三次	19	711115 無均衡,集中	765701 無均衡,集中	794025 無均衡,集中	751324 無均衡,集中	501	60
第四次	18	771674 無均衡,集中	451	527	5542	1769	707717 無均衡,集中
第五次	20	721622 無均衡,集中	636	743186 無均衡,集中	162	528	1790
第六次	18	750987 無均衡,集中	723659 無均衡,集中	751122 無均衡,集中	344	754150 無均衡,集中	728968 無均衡,集中
第七次	25	752297 無均衡,集中	784202 無均衡,集中	1223	759190 無均衡,集中	1344	186
第八次	21	736691 無均衡,集中	1354	977	18	1558	2045
第九次	15	770769 無均衡,集中	17	1015	2907	2621	190
第十次	20	722450 無均衡,集中	15	2560	18	727911 無均衡,分散	142
Max	25	4387	1354	2560	5542	2621	2045
Min	15	4387	15	521	15	222	16
Average	18.7	4387	432	1137	1131	1220	632

<表 5-2>

1.無固定攤販之模型雖然理論上存在均衡態，但透過程式模擬的時候不一定每次皆可形成均衡。

2.同一攤販數下，當達到均衡時，可產生不只一種均衡態，如理論所預測。

2.由於四家流動攤販只有唯一一種攤販兩兩聚集於兩側的均衡態，因此較易達成。

3.五家流動攤販的均衡態必須有一家流動攤販恰好選擇正中央的位置，因此較難達成均衡。

4.當無法達成均衡時，流動攤販大多聚集於沙灘中央重複同樣的移動過程，唯在九家流動攤販中，有一次實驗結果攤販會均勻分布於沙灘上，我們觀察其移動過程，發現不斷重複，因此推定此次無法達成均衡。

5.受限於實驗模擬次數，無固定攤販之模型產生均衡的速度與攤販數看不出相關性。

陸、結論

一、 在均為流動攤販的情況下，經由探討會發現當達成均衡時，最多只會有兩個攤販共點，並且共點攤販左右分得的區域相等。而最靠近兩端點的攤販必為兩兩共點。

二、 一維直線且沒有固定攤販時，在攤販數為 3 時沒有均衡。

三、 一維直線且沒有固定攤販時，若所有的攤販均勻分布於直線上，可以達到最大公共利益，但卻不是均衡的形式之一，因為端點攤販須為共點攤販。而因此在均衡形式中，使公共利益最大化的形式為只有端點攤販共點，其餘攤販獨立；使公共利益最小化的形式則是當所有攤販皆為共點攤販。

四、 n 家攤販中有一個攤販"固定不動"時，均衡是否存在，不一定與 n-1 個或 n 個流動攤販是否有均衡相關。

五、 一維直線且有一固定攤販時，於任何攤販數皆可能形成均衡。

六、 當有一個固定攤販時，流動攤販的奈許均衡是否存在及均衡的型態只由固定攤販的位置決定。

七、 當 $n\geq 4$ 時，若 n 為奇數，則當固定攤販位於 $\frac{1}{n+1}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡；若 n 為偶數，則當固定攤販位於 $\frac{1}{n+2}<b<\frac{3}{2n-2}$ 時，其餘 n 家流動攤販無法形成奈許均衡。

八、 當 $0\leq b\leq\frac{1}{2}$ 的範圍中，除了 $b=\frac{3}{2n-2}$ 外，其餘位置若可以使均衡存在，都會存在某組均衡，使得固定攤販的利益 $f_b<\frac{1}{n-1}$ 。

九、 當 $n\geq 4$ 時，若固定攤販位於 $\frac{3}{2(n-1)}$ ，則流動攤販存在唯一的奈許均衡，此時固定攤販可獲得確定利益 $\frac{1}{n-1}$ ，並且其為最大保證獲利(*Max Minimum*)。

十、 當 $n\geq 4$ 時，若固定攤販位於 $\frac{3}{2(n-1)}$ ，則流動攤販存在唯一的奈許均衡，此均衡形式為除兩端攤販共點其餘攤販獨立，為可以達成最大公益的均衡形式，因此我們可以知道，固定攤販可以控制自己的位置使自己獲得一定獲利並使公共利益最大化。

十一、在常態分布且有一家固定攤販的模型中，達成均衡時，最多只會有兩家流動攤販

十二、平面模型中，假設流動攤販所在位置對應到的遊客人數為 h_1 ，而其與其他攤販利益平分線的遊客人數為 h_2 ，則流動攤販的共點條件為： $h_1\geq h_2$

十三、若 $g\left(\frac{a_{k-1}+a_k}{2}\right)<g\left(\frac{a_k+a_{k+1}}{2}\right)$ 或

$g\left(\frac{a_{k-1}+a_k}{2}\right)=g\left(\frac{a_k+a_{k+1}}{2}\right)$ 且 $g'\left(\frac{a_{k-1}+a_k}{2}\right)<g'\left(\frac{a_k+a_{k+1}}{2}\right)$ ，則流動攤販 a_k 會往右移。

若 $g\left(\frac{a_{k-1}+a_k}{2}\right)>g\left(\frac{a_k+a_{k+1}}{2}\right)$ 或

$g\left(\frac{a_{k-1}+a_k}{2}\right)=g\left(\frac{a_k+a_{k+1}}{2}\right)$ 且 $g'\left(\frac{a_{k-1}+a_k}{2}\right)>g'\left(\frac{a_k+a_{k+1}}{2}\right)$ ，則流動攤販 a_k 會往左移。

十四、M 型化模型中，由於只有四條線等高，因此均衡時最多只會有兩組流動攤販與一個固定攤販，且共點攤販只能位於端點，因此五家以上的流動攤販皆無法形成均衡。

十五、M 型化模型中，在 $n=1\sim 4$ 時，固定攤販的最大獲利模式與一維模型相同。

十六、根據程式模擬，當有固定攤販時，必可產生如定理四所預測之唯一均衡態，固定攤販可獲得最大利益，且達成均衡所需的回數隨著攤販數增加而增加。

十七、根據程式模擬，沒有固定攤販時，不一定可以達到均衡，而達成均衡時可以產生不只一種均衡型態，所需回數亦與攤販數無明顯相關。

柒、討論與應用

若我們將攤販視為商場上的企業，或是政治上不同的政黨，將其位置視為產品定位或是政策取向，奈許均衡的結果可以預測其定位所帶來的利益，或瓜分到的選票支持。而固定攤販可視為不隨著眼前利益而改變策略的公司。當模型為一維直線，即顧客平均分布在沙灘的情形，我們已證明出固定攤販的最大保證獲利為：當$n \geq 4$時，固定攤販位於$\frac{3}{2(n-1)}$，且固定攤販獲得利益$\frac{1}{n-1}$。由我們的結果可以得知，選取適當的固定策略可以保證帶來優於平均的利益，其獲利亦高於可藉由移動獲得更多利益的流動攤販。而若有一家以上的固定攤販，其獲利最大值亦為$\frac{1}{n-1}$。

在常態分布模型中，雖然並無許多種均衡型態，且固定攤販所獲得的利益也無法大於流動攤販的利益，但是在 M 型化分布模型中，可以得到與直線模型平行的結果，均衡時固定攤販亦可獲得較大的利益。

我們成功的歸納出一維模型中，n 家攤販與固定攤販的利益關係，也整理出固定攤販對於均衡型態的影響，亦分析出了常態分布、M 型化分布的均衡型態以及攤販所在位置之關係，得到在平面型中形成均衡的條件，而探討不均勻的遊客分佈情形，讓我們的研究結果貼近現實生活，增加了模型的應用範圍。

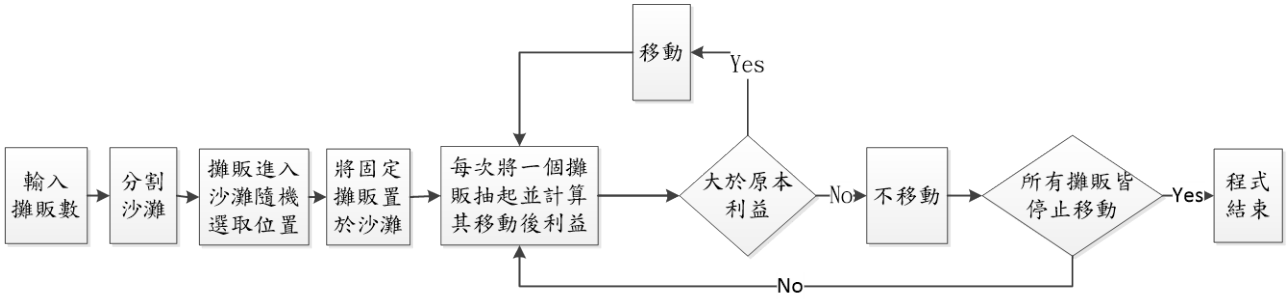
最後，我們以程式分別模擬了一維模型中有一家固定攤販的情形及沒有固定攤販的對照組，證明固定攤販有助於均衡的形成，並可以決定均衡的型態，使其餘攤販產生唯一均衡，亦證明我們提出的模型並非只是理論，是可以用來實際模擬現實生活中的競爭情況。

捌、未來展望

一、改變模擬程式中第一回合流動攤販位置的初始值

在原本模型中，我們是假設固定攤販先進入沙灘站定位，然後流動攤販依序進入沙灘選擇一個當下可以獲得最大利益的位置設攤。我們希望能更改程式，使流動攤販一開始是隨機選取一個位置，然後再依序調整設攤位置，並藉此程式模擬當固定攤販非第一家進入沙灘的攤販時，對均衡所造成的影響。在程式中，每家流動攤販都重新決定位置稱為一回，而為了使固定攤販可以位於公式所提出之均衡點上，我們依據公式將沙灘分割為約 100 格。

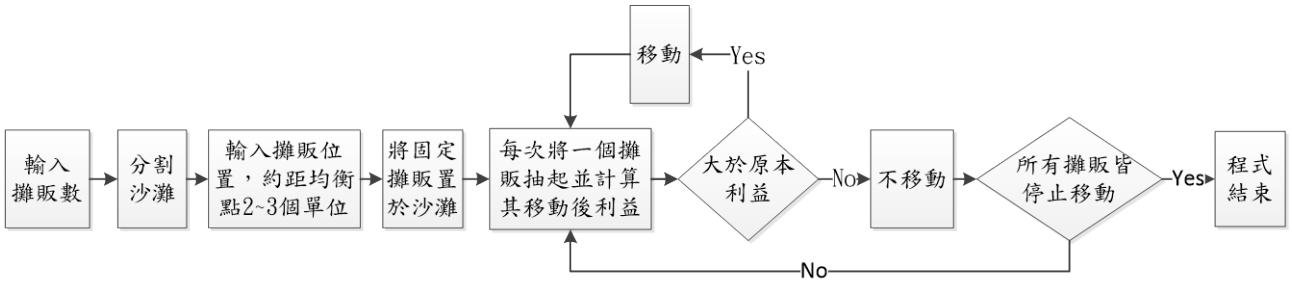
以下為程式流程圖：



二、觀察固定攤販對均衡穩定性的影響

由於當攤販隨著攤販數的增加，達到均衡需要移動的次數也會急遽增大，而在沒有固定攤販的模擬對照組中，常常達不到均衡。因此我們希望能觀察若將流動攤販的初始位置擺放在均衡點附近某個範圍內，是否能較快達到均衡？藉此分析當均衡受到輕微擾動的穩定性，並比較有無設置固定攤販兩種模型在此設定下達成均衡的成功率及速度快慢。藉此了解固定攤販是否會增加均衡的穩定性。在程式中，每家流動攤販都重新決定位置稱為一回，而為了使固定攤販可以位於公式所提出之均衡點上，我們依據公式將沙灘分割為約 100 格。

以下為程式流程圖：



玖、參考文獻

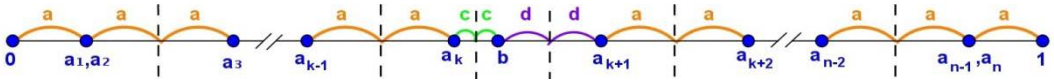
1. Hotelling, H. (1929), “ Stability in Competition”, Economic Journal, Vol. 39, No. 153, pp. 41–57
2. Eaton, B. and Lipsey R. G. (1975), “The Principle of Minimum Differentiation Reconsidered: Some New Developments in the Theory of Spatial Competition”, The Review of Economic Studies, Vol. 42, No. 1, pp. 27-4

附錄一

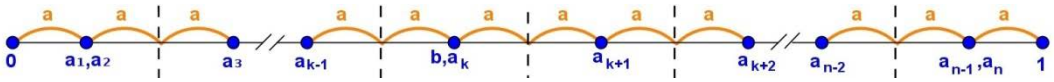
引理 4：當 $n \geq 6$ 時，對任意 $\frac{3}{2n-4} \leq b \leq 1-\frac{3}{2n-4}$ ，

$$a_1 = a_2 = \frac{1}{2n-4}, a_k = \frac{2k-3}{2n-4}\left(3 \leq k \leq n-2\right), a_{n-1} = a_n = \frac{2n-5}{2n-4}$$ **會是一組奈許均衡。**

證明：如下<圖 1>、<圖 2>，顯然不論 $\frac{2k-3}{2n-4} < b < \frac{2k-1}{2n-4}\left(3 \leq k \leq n-3\right)$ ，即 $a_k < b < a_{k+1}$ ，或 $b = a_k = \frac{2k-3}{2n-4}\left(3 \leq k \leq n-2\right)$，均沒有任何流動攤販能藉由單獨移動獲得更大利益，故其為一組均衡解，並且此時固定攤販的獲利為 $f_b = \frac{1}{2n-4}$ 。



<圖 1>



<圖 2>

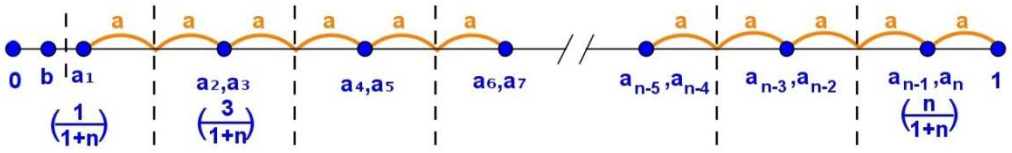
引理 5：當 $n \geq 3$ 時，若 n 為奇數，則對任意 $0 \leq b \leq \frac{1}{n+1}$ ，

$$a_1 = \frac{1}{n+1}, a_{2k} = a_{2k+1} = \frac{2k+1}{n+1}, \forall 1 \leq k \leq \left[\frac{n}{2}\right]$$ **會是一組奈許均衡；**

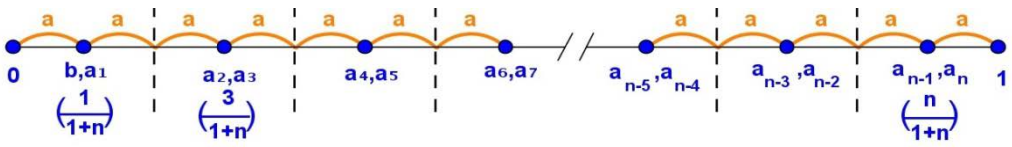
若 n 為偶數，則對任意 $0 \leq b \leq \frac{1}{n+2}$ ，

$$a_1 = \frac{1}{n+2}, a_2 = \frac{3}{n+2}, a_{2k-1} = a_{2k} = \frac{2k+1}{n+2}, \forall 2 \leq k \leq \left[\frac{n}{2}\right]$$ **會是一組奈許均衡。**

證明：若 n 為奇數，如<圖 3>~<圖 4>，顯然其為一組奈許均衡。



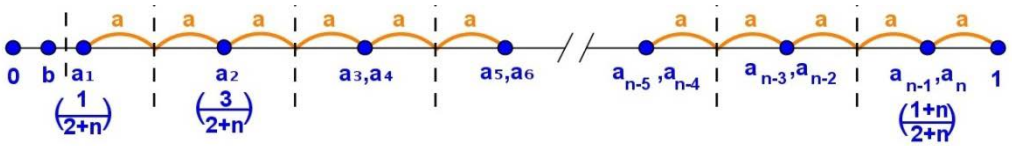
<圖 3>



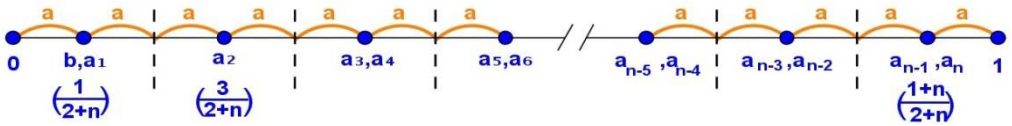
<圖 4>

並且此時固定攤販的獲利 $f_b \leq \frac{1}{n+1}$ 。

若 n 為偶數，如<圖 5>~<圖 6>，顯然其為一組奈許均衡。



<圖 5>



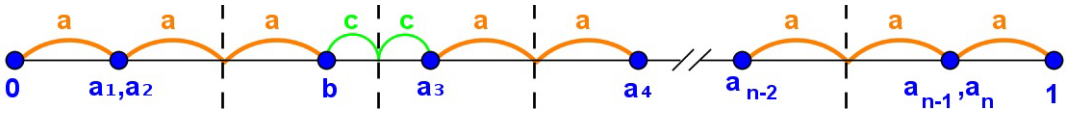
<圖 6>

並且此時固定攤販的獲利 $f_b \le \dfrac{1}{n+2}$。

引理 6：當 $n \ge 5$ 時，對任意 $\dfrac{3}{2n-2} \le b < \dfrac{3}{2n-4}$，

$a_1 = a_2 = \dfrac{b}{3}, a_k = 1 - \dfrac{(2n-2k-1)b}{3} (3 \le k \le n-2), a_{n-1} = a_n = 1 - \dfrac{b}{3}$ 會是一組奈許均衡。

證明：如<圖 7>，顯然其為一組奈許均衡。此時固定攤販的獲利為 $f_b = \dfrac{1}{2} - \dfrac{(n-3)b}{3}$，並且當 $b = \dfrac{3}{2n-2}$ 時固定攤販的獲利有最大值 $\dfrac{1}{n-1}$。



<圖 7>

引理 7：(i) 在均衡時，設共點攤販所分得的利益為 a，若兩組共點攤販間僅有奇數家獨立的流動攤販位於 $a_{m+1} < a_{m+2} < \cdots < a_{m+2k+1}$，則此 $2k+1$ 家流動攤販的利益總和

$$\sum_{j=1}^{2k+1} f_{m+j} \ge (2k+2)a$$

(ii) 在均衡時，若固定攤販位於最左側 b 的位置，而他與最近的共點攤販之間若有偶數家獨立攤販位於 $a_1 < a_2 < \cdots < a_{2k}$，則此 $2k$ 家流動攤販與固定攤販的利益總和

$$f_b + \sum_{j=1}^{2k} f_j \ge (2k+2)b$$

證明：

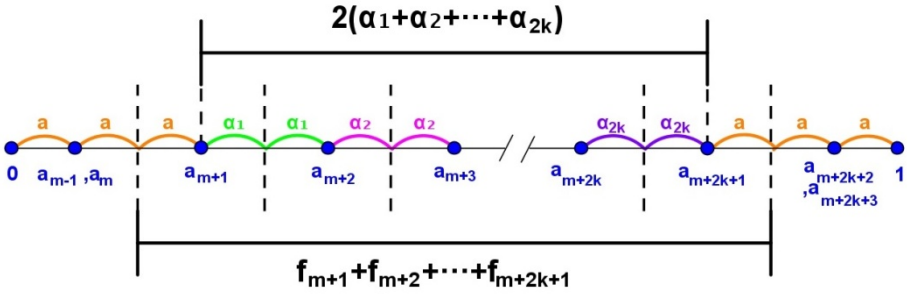
(i) 如<圖 8>，設 $\frac{a_{m+2}-a_{m+1}}{2}=\alpha_1, \frac{a_{m+3}-a_{m+2}}{2}=\alpha_2, \cdots, \frac{a_{m+2k+1}-a_{m+2k}}{2}=\alpha_{2k}$ ，

則

$$f_{m+1}=a+\alpha_1,\ f_{m+j}=\alpha_{j-1}+\alpha_j\ \ \forall 2\le j\le 2k,\ f_{2k+1}=\alpha_{2k}+a.$$

由於獨立攤販的獲利不小於共點攤販，故此 $2k+1$ 家流動攤販的利益總和

$$\sum_{j=1}^{2k+1} f_{m+j}=2a+2\sum_{j=1}^{2k}\alpha_j=2a+2\sum_{i=1}^{k} f_{m+2i}\ge(2+2k)a.$$



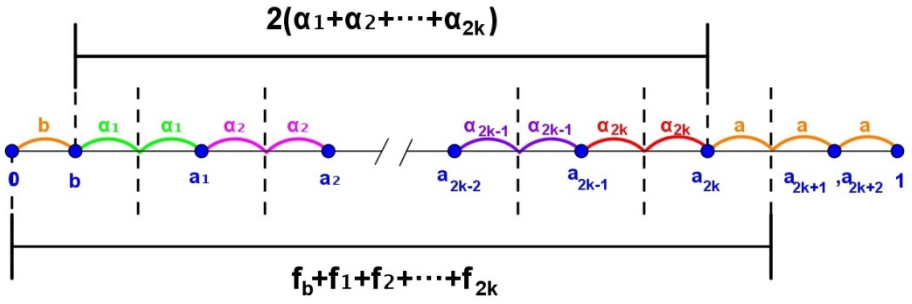
<圖 8>

(ii) 如<圖 9>，設 $\frac{a_1-b}{2}=\alpha_1, \frac{a_j-a_{j-1}}{2}=\alpha_j, \forall 2\le j\le 2k$ 且共點攤販所分得的利益為 a ，則

$f_b=b+\alpha_1,\ f_j=\alpha_j+\alpha_{j+1}\ \ \forall 1\le j\le 2k-1,\ f_{2k}=\alpha_{2k}+a$ 。

而此 $2k$ 家流動攤販與固定攤販的利益總和

$$f_b+\sum_{j=1}^{2k} f_j=b+2\sum_{j=1}^{2k}\alpha_j+a=b+a+2\sum_{i=1}^{k} f_{2i-1}\ge b+(1+2k)a\ge(2+2k)b.\blacksquare$$



<圖 9>

附錄二

有固定攤販模型之程式碼

```
// ConsoleApplication1.cpp : 定義主控台應用程式的進入點。

#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include <time.h>

#define C_MaxNumberOfBooth      100
#define C_MaxPosition           100
#define C_FixedBoothPosition 30

unsigned int g16PositionCnt, g16BoothCnt, g16TotalLength, g16IntervalCnt;
unsigned int g16arPosition[C_MaxPosition], g16arBoothCntOfPosition[C_MaxPosition];
unsigned int g16arBoothPositionIdx[C_MaxNumberOfBooth];
unsigned int g16arInterval[C_MaxPosition + 1];
unsigned int g16BoothIsOnLine[C_MaxNumberOfBooth];
unsigned seed;

double    gdarOldRight[C_MaxNumberOfBooth], gdarRight[C_MaxNumberOfBooth];
unsigned int g16CurrentNumOfBoothOnLine;

void CalRightOfBoothOnLine(void)
{
     unsigned int i, u16PosIdx;
     double dBoothRight,dLRight,dRRight;
     for (i = 0; i < g16BoothCnt; i++)
     {
          if (g16BoothIsOnLine[i])
          {
               u16PosIdx = g16arBoothPositionIdx[i];
               dLRight = g16arInterval[u16PosIdx];
               if (u16PosIdx != 0) dLRight = dLRight / 2;        //(J)/2 if not first position

               dRRight = g16arInterval[u16PosIdx + 1];
               if (u16PosIdx != (g16PositionCnt - 1))  dRRight = dRRight / 2;   //(J)/2 id not first
postion

               dBoothRight = dLRight + dRRight;
               if (g16arBoothCntOfPosition[u16PosIdx] == 2) dBoothRight = dBoothRight / 2;
               gdarRight[i] = dBoothRight;
          }
```

```
            else
                gdarRight[i] = 0;
        }
}

int _tmain(int argc, _TCHAR* argv[])
{
        unsigned int u16IntervalIdx, u16Booth, u16PositonIdx, u16OrgIntervalIdx;
        unsigned int i, j, k, t, lenth, u16Round, u16ChangePosCnt;
        unsigned int u16PositionIdxOfBooth, u16OrgPosition;
        double dRight, dMaxRight, dOrgRight;
        bool bOk;

        printf("Please enter number of booth... ");
        scanf_s("%d", &g16BoothCnt);

            if (g16BoothCnt == 0)
            {
                printf("Invlid number of booth! Program terminated!\n");
                return 0;
            }
            lenth = 100 / ((g16BoothCnt * 2) - 2);
            g16TotalLength = ((g16BoothCnt * 2) - 2) * lenth;
            g16BoothCnt++;      //(J)total number of booth is iNumberOfBooth+1 for a fixed position booth
            printf("The total length is %d\n", g16TotalLength);

            for (i = 0; i < g16BoothCnt; i++)
            {
                g16BoothIsOnLine[i] = 0;
                gdarRight[i] = 0;
                gdarOldRight[i] = 0;
            }

            g16BoothIsOnLine[0] = 1;                                    //(J)fixed booth0 is already on
line
            g16arPosition[0] = 3 * lenth;
            g16CurrentNumOfBoothOnLine = 1;
            g16arBoothCntOfPosition[0] = 1;
            g16arBoothPositionIdx[0] = 0;                               //(J)means booth_0 at position_0
            g16PositionCnt = 1;
            g16IntervalCnt = g16PositionCnt + 1;
            g16arInterval[0] = g16arPosition[0];                    //(J)-1 for start from 1
            g16arInterval[1] = g16TotalLength - g16arPosition[0];
            gdarRight[0] = g16TotalLength;
```

```
u16Round = 1;
bOk = 0;
while (!bOk)
{
	u16Booth = 0;
	for (j = 0; j < g16PositionCnt; j++)
	{
		u16Booth += g16arBoothCntOfPosition[j];
	}
	u16PositonIdx = 0;
	for (j = 0; j < g16BoothCnt; j++)
	{
		if (g16BoothIsOnLine[j]) u16PositonIdx++;
	}
	if (u16Booth != u16PositonIdx)
	{
		printf("shit");
	}

	u16ChangePosCnt = 0;
	printf("Round%d start...\n", u16Round);
	for (i = 1; i < g16BoothCnt; i++)
	{	//(J)1 by 1 put booth on line
		u16Booth = 0;
		for (j = 0; j < g16PositionCnt; j++)
		{
			u16Booth += g16arBoothCntOfPosition[j];
		}
		u16PositonIdx = 0;
		for (j = 0; j < g16BoothCnt; j++)
		{
			if (g16BoothIsOnLine[j]) u16PositonIdx++;
		}
		if (u16Booth != u16PositonIdx)
		{
			printf("shit");
		}

		dOrgRight = 0;
		u16OrgPosition = 0;

		if (g16BoothIsOnLine[i])
		{	//(J)booth already online-> pick booth from line and re-calculate right of each
```

```
booth
                        dOrgRight = gdarRight[i];
                        u16OrgPosition = g16arPosition[g16arBoothPositionIdx[i]];

                        u16PositionIdxOfBooth = g16arBoothPositionIdx[i];
                        if (g16arBoothCntOfPosition[u16PositionIdxOfBooth] > 1)
                        {    //(J)number of booth at position is 2
                             g16arBoothCntOfPosition[u16PositionIdxOfBooth]--;
                        }
                        else
                        {    //(J)number of booth at position is 1
                             for (j = 0; j <= g16BoothCnt; j++)
                             {
                                  if ((g16BoothIsOnLine[j]) && (g16arBoothPositionIdx[j] >
u16PositionIdxOfBooth))
                                       g16arBoothPositionIdx[j]--;             //(J)increase
position index of all booth
                             }

                             for (j = u16PositionIdxOfBooth; j < (g16PositionCnt - 1); j++)
                             {
                                  g16arPosition[j] = g16arPosition[j + 1];
                                  g16arBoothCntOfPosition[j] = g16arBoothCntOfPosition[j + 1];
                             }
                             g16arPosition[j] = 0;
                             g16arBoothCntOfPosition[j] = 0;
                             g16PositionCnt--;

                             g16arInterval[u16PositionIdxOfBooth] =
g16arInterval[u16PositionIdxOfBooth] + g16arInterval[u16PositionIdxOfBooth + 1];
                             for (j = u16PositionIdxOfBooth + 1; j < (g16IntervalCnt - 1); j++)
                                  g16arInterval[j] = g16arInterval[j + 1];

                             g16arInterval[j] = 0;
                             g16IntervalCnt--;
                        }
                        g16BoothIsOnLine[i] = 0;
                        g16CurrentNumOfBoothOnLine--;
                        CalRightOfBoothOnLine();
                   }//(J)find max right to put new booth

                   dMaxRight = 0;
                   u16IntervalIdx = 0;
                   for (j = 0; j < (g16IntervalCnt + g16PositionCnt); j++)      //(J)check every on
```

```
interval case and overlap booth case
                    {
                         if (j == 0)
                         {
                              dRight = (gl6arInterval[0] - 1) + 0.5;            //(J)first interval
                         }
                         else if (j == (gl6IntervalCnt - 1))
                         {
                              dRight = (gl6arInterval[j] - 1) + 0.5;       //(J)last interval
                         }
                         else if (j >= gl6IntervalCnt)                //(J)overlap case
                         {
                              if (gl6arBoothCntOfPosition[j - gl6IntervalCnt] < 2)
                              {
                                   for (ul6Booth = 0; ul6Booth < gl6BoothCnt; ul6Booth++)
                                   {
                                        if ((gl6BoothIsOnLine[ul6Booth]) &&
(gl6arBoothPositionIdx[ul6Booth] == (j - gl6IntervalCnt)))
                                             break;
                                   }
                                   dRight = ((double)gdarRight[ul6Booth]) / 2;
                              }
                              else
                                   dRight = 0;
                         }
                         else
                         {    //(J)put booth at middle of interval
                              dRight = ((double)gl6arInterval[j]) / 2;
                         }

                         if (dRight > dMaxRight)
                         {
                              ul6IntervalIdx = j;
                              dMaxRight = dRight;
                         }
                         if (dRight == dMaxRight)         //(i)when there are two same dmaxright, choose
the random position
                         {
                              ul6OrgIntervalIdx = ul6IntervalIdx;
                              ul6IntervalIdx = j;
                              k = rand() % (100);
                              t = k % 2;
                              if (t = 0)
                              {
```

```
                    u16IntervalIdx = u16OrgIntervalIdx;
                }
                else
                {
                    u16IntervalIdx = j;
                }
            }
        }
        if (dMaxRight <= dOrgRight)
        {//(J)right doesn't increase after change-> put booth back to original position
            u16PositonIdx = 0;
            while ((g16arPosition[u16PositonIdx] < u16OrgPosition) && (u16PositonIdx <
g16PositionCnt))    u16PositonIdx++;    //(J)find the first position which is on the right hand of the
original position
            g16arBoothPositionIdx[i] = u16PositonIdx;
            if (g16arPosition[u16PositonIdx] == u16OrgPosition)
            {
                g16arBoothCntOfPosition[u16PositonIdx]++;          //(J)booth cnt at
position +1
            }
            else
            {
                for (j = 0; j <= g16BoothCnt; j++)
                {
                    if ((g16BoothIsOnLine[j]) && (g16arBoothPositionIdx[j] >=
u16PositonIdx))
                        g16arBoothPositionIdx[j]++;              //(J)increase
position index of all booth on the right hand of the inserted booth
                }
                for (j = g16PositionCnt; j > u16PositonIdx; j--)
                {
                    g16arPosition[j] = g16arPosition[j - 1];
                    g16arBoothCntOfPosition[j] = g16arBoothCntOfPosition[j - 1];
                }
                g16arPosition[j] = u16OrgPosition;
                g16arBoothCntOfPosition[j] = 1;
                g16PositionCnt++;
                for (j = 0; j <= g16PositionCnt; j++)
                {
                    if (j == 0)
                    {
                        g16arInterval[0] = g16arPosition[0];
                    }
                    else if (j == g16PositionCnt)
```

```
                    {
                        g16arInterval[j] = g16TotalLength - g16arPosition[j - 1];
                    }
                    else
                    {
                        g16arInterval[j] = g16arPosition[j] - g16arPosition[j - 1];
                    }
                }
                g16IntervalCnt++;
            }
        }
        else
        {
            //(J)insert booth to line
            u16ChangePosCnt++;
            if (u16IntervalIdx == 0)
            {
                for (j = g16PositionCnt; j > 0; j--)         //(J)shift position to
right
                {
                    g16arPosition[j] = g16arPosition[j - 1];
                    g16arBoothCntOfPosition[j] = g16arBoothCntOfPosition[j - 1];
                }
                g16arPosition[0] = g16arPosition[j + 1] - 1;
                g16arBoothCntOfPosition[0] = 1;
                g16PositionCnt++;

                for (j = g16IntervalCnt; j > 1; j--)
                {
                    g16arInterval[j] = g16arInterval[j - 1];
                }
                g16arInterval[1] = 1;
                g16arInterval[0] = g16arInterval[0] - 1;
                g16IntervalCnt++;

                for (j = 0; j <= g16BoothCnt; j++)
                {
                    if (g16BoothIsOnLine[j])
                        g16arBoothPositionIdx[j]++;             //(J)increase
position index of all booth
                }
                g16arBoothPositionIdx[i] = 0;                //(J)means booth_i at
position_0
            }
```

```
                    else if (u16IntervalIdx == (g16IntervalCnt - 1))
                    {
                        g16arBoothPositionIdx[i] = g16PositionCnt;    //(J)booth at the last
position
                        g16arPosition[g16PositionCnt] = g16arPosition[g16PositionCnt - 1] + 1;
                        g16arBoothCntOfPosition[g16PositionCnt] = 1;
                        g16PositionCnt++;

                        g16arInterval[g16IntervalCnt] = g16arInterval[g16IntervalCnt - 1] - 1;
                        g16arInterval[g16IntervalCnt - 1] = 1;
                        g16IntervalCnt++;

                        g16arBoothPositionIdx[i] = g16PositionCnt - 1;          //(J)means
booth_i at the last position
                    }
                    else if (u16IntervalIdx >= g16IntervalCnt)                  //(J)overlap
case --> interval count not change
                    {
                        //u16Booth = u16IntervalIdx - g16IntervalCnt;
                        //u16PositonIdx = g16arBoothPositionIdx[u16Booth];
                        u16PositonIdx = u16IntervalIdx - g16IntervalCnt;

                        g16arBoothCntOfPosition[u16PositonIdx]++;          //(J)booth cnt at
position +1
                        g16arBoothPositionIdx[i] = u16PositonIdx;
                    }
                    else //(j)u16IntervalIdx>0 && u16IntervalIdx< (g16IntervalCnt - 1)
                    {          //(J)inster booth in interval
                        for (j = 0; j <= g16BoothCnt; j++)
                        {
                            if ((g16BoothIsOnLine[j]) && (g16arBoothPositionIdx[j] >=
u16IntervalIdx))
                                g16arBoothPositionIdx[j]++;             //(J)increase
position index of all booth
                        }
                        g16arBoothPositionIdx[i] = u16IntervalIdx;
                        for (j = g16IntervalCnt; j > (u16IntervalIdx + 1); j--)
                        {
                            g16arInterval[j] = g16arInterval[j - 1];
                            if ((j - u16IntervalIdx) >= 2)
                            {
                                g16arPosition[j - 1] = g16arPosition[j - 2];
                                g16arBoothCntOfPosition[j - 1] = g16arBoothCntOfPosition[j - 2];
                            }
```

```
                                }
                                g16arBoothCntOfPosition[j - 1] = 1;
                                seed = (unsigned)time(NULL); // 取得時間序列
                                srand(seed); // 以時間序列當亂數種子
                                j = rand() % (g16arInterval[u16IntervalIdx] - 1);       //(J)for put the
booth at random position of the interval
                                j = j + 1;
                                g16arInterval[u16IntervalIdx + 1] = g16arInterval[u16IntervalIdx] - j;
                                g16arInterval[u16IntervalIdx] = j;
                                g16IntervalCnt++;
                                g16arPosition[u16IntervalIdx] = g16arPosition[u16IntervalIdx - 1] +
g16arInterval[u16IntervalIdx];
                                g16PositionCnt++;
                           }
                     }
                     g16BoothIsOnLine[i] = 1;
                     g16CurrentNumOfBoothOnLine++;
                     CalRightOfBoothOnLine();
                }
                //(J)display result
                printf("**********************************\n");
                printf("Total position count=%d \n", g16PositionCnt);
                for (i = 0; i < g16PositionCnt; i++)
                     printf("Position[%d]=%d \n", i, g16arPosition[i]);

                printf("Total booth count=%d \n", g16BoothCnt);

                for (i = 0; i < g16BoothCnt; i++)
                {
                     printf("Booth[%d] at Position[%d] has right of %6.4f\n", i, g16arBoothPositionIdx[i],
gdarRight[i]);
                }
                //(J)check stable
                if (u16ChangePosCnt == 0)
                {
                     bOk = 1;
                     printf("Round%d start...\n", u16Round);
                     printf("The total length is %d\n", g16TotalLength);
                     printf("All booths reach stable!\n");
                     system("pause");
                }
                else
                {    //(J)backup right of all booth
                     for (i = 0; i < g16BoothCnt; i++)
```

```
                    {
                        gdarOldRight[i] = gdarRight[i];
                    }
                    printf("All booths are not stable yet, next round!\n");
                    u16Round++;
                }
                printf(" \n");
            }
            return 0;
        }
```

無固定攤販模型之程式碼

```
// ConsoleApplication1.cpp : 定義主控台應用程式的進入點。

#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include <time.h>

#define C_MaxNumberOfBooth      100
#define C_MaxPosition           100
#define C_FixedBoothPosition 30

unsigned int gl6PositionCnt, gl6BoothCnt, gl6TotalLength, gl6IntervalCnt;
unsigned int gl6arPosition[C_MaxPosition], gl6arBoothCntOfPosition[C_MaxPosition];
unsigned int gl6arBoothPositionIdx[C_MaxNumberOfBooth];
unsigned int gl6arInterval[C_MaxPosition + 1];
unsigned int gl6BoothIsOnLine[C_MaxNumberOfBooth];
unsigned seed;

double    gdarOldRight[C_MaxNumberOfBooth], gdarRight[C_MaxNumberOfBooth];
unsigned int gl6CurrentNumOfBoothOnLine;

void CalRightOfBoothOnLine(void)
{
    unsigned int i, u16PosIdx;
    double dBoothRight,dLRight,dRRight;
    for (i = 0; i < gl6BoothCnt; i++)
    {
        if (gl6BoothIsOnLine[i])
        {
            u16PosIdx = gl6arBoothPositionIdx[i];
            dLRight = gl6arInterval[u16PosIdx];
            if (u16PosIdx != 0) dLRight = dLRight / 2;        //(J)/2 if not first position
```

```
                dRRight = gl6arInterval[ul6PosIdx + 1];
                if (ul6PosIdx != (gl6PositionCnt - 1))  dRRight = dRRight / 2;   //(J)/2 id not first
postion
                dBoothRight = dLRight + dRRight;
                if (gl6arBoothCntOfPosition[ul6PosIdx] == 2) dBoothRight = dBoothRight / 2;
                gdarRight[i] = dBoothRight;
           }
           else
                gdarRight[i] = 0;
     }
}
int _tmain(int argc, _TCHAR* argv[])
{
     unsigned int ul6IntervalIdx, ul6Booth, ul6PositonIdx, ul6OrgIntervalIdx;
     unsigned int i, j, k,t,ul6Round,ul6ChangePosCnt;
     unsigned int ul6PositionIdxOfBooth, ul6OrgPosition;
     double dRight,dMaxRight, dOrgRight;
     bool bOk;

     printf("Please enter number of booth... ");
     scanf_s("%d", &gl6BoothCnt);
     if (gl6BoothCnt == 0)
     {
          printf("Invlid number of booth! Program terminated!\n");
          return 0;
     }
     gl6TotalLength =100;
     printf("The total length is %d\n", gl6TotalLength);

     for (i = 0; i < gl6BoothCnt; i++)
     {
          gl6BoothIsOnLine[i] = 0;
          gdarRight[i] = 0;
          gdarOldRight[i] = 0;
     }
     gl6BoothIsOnLine[0] = 1;                    //(J)fixed booth0 is already on line
     seed = (unsigned)time(NULL); // 取得時間序列
     srand(seed); // 以時間序列當亂數種子
     gl6arPosition[0] = rand() % (96);
     gl6CurrentNumOfBoothOnLine = 1;
     gl6arBoothCntOfPosition[0] = 1;
     gl6arBoothPositionIdx[0] = 0;                    //(J)means booth_0 at position_0
     gl6PositionCnt = 1;
     gl6IntervalCnt = gl6PositionCnt + 1;
```

```
g16arInterval[0] = g16arPosition[0];                    //(J)-1 for start from 1
g16arInterval[1] = g16TotalLength - g16arPosition[0];
gdarRight[0] = g16TotalLength;
u16Round = 1;
bOk = 0;
while (!bOk)
{
     u16Booth = 0;
     for (j = 0; j < g16PositionCnt; j++)
     {
          u16Booth += g16arBoothCntOfPosition[j];
     }
     u16PositonIdx = 0;
     for (j = 0; j < g16BoothCnt; j++)
     {
          if (g16BoothIsOnLine[j]) u16PositonIdx++;
     }
     if (u16Booth != u16PositonIdx)
     {
          printf("shit");
     }
     u16ChangePosCnt = 0;
     printf("Round%d start...\n", u16Round);
     for (i = 0; i < g16BoothCnt; i++)
     {    //(J)1 by 1 put booth on line
          u16Booth = 0;
          for (j = 0; j < g16PositionCnt; j++)
          {
               u16Booth += g16arBoothCntOfPosition[j];
          }
          u16PositonIdx = 0;
          for (j = 0; j < g16BoothCnt; j++)
          {
               if (g16BoothIsOnLine[j]) u16PositonIdx++;
          }
          if (u16Booth != u16PositonIdx)
          {
               printf("shit");
          }
          dOrgRight = 0;
          u16OrgPosition = 0;
          if (g16BoothIsOnLine[i])
          {//(J)booth already online-> pick booth from line and re-calculate right of each booth
               dOrgRight = gdarRight[i];
```

```
                    u16OrgPosition = g16arPosition[g16arBoothPositionIdx[i]];
                    u16PositionIdxOfBooth = g16arBoothPositionIdx[i];
                    if (g16arBoothCntOfPosition[u16PositionIdxOfBooth] > 1)
                    {    //(J)number of booth at position is 2
                         g16arBoothCntOfPosition[u16PositionIdxOfBooth]--;
                    }
                    else
                    {    //(J)number of booth at position is 1
                         for (j = 0; j <= g16BoothCnt; j++)
                         {
                              if((g16BoothIsOnLine[j]) &&
(g16arBoothPositionIdx[j]>u16PositionIdxOfBooth))
                                   g16arBoothPositionIdx[j]--;             //(J)increase position
index of all booth
                         }
                         for (j = u16PositionIdxOfBooth; j < (g16PositionCnt - 1); j++)
                         {
                              g16arPosition[j] = g16arPosition[j + 1];
                              g16arBoothCntOfPosition[j] = g16arBoothCntOfPosition[j + 1];
                         }
                         g16arPosition[j] = 0;
                         g16arBoothCntOfPosition[j] = 0;
                         g16PositionCnt--;
                         g16arInterval[u16PositionIdxOfBooth] = g16arInterval[u16PositionIdxOfBooth] +
g16arInterval[u16PositionIdxOfBooth + 1];
                         for (j = u16PositionIdxOfBooth + 1; j < (g16IntervalCnt - 1); j++)
                              g16arInterval[j] = g16arInterval[j + 1];
                         g16arInterval[j] = 0;
                         g16IntervalCnt--;
                    }
                    g16BoothIsOnLine[i] = 0;
                    g16CurrentNumOfBoothOnLine--;
                    CalRightOfBoothOnLine();
               }
               //(J)find max right to put new booth
               dMaxRight = 0;
               u16IntervalIdx = 0;
               for (j = 0; j < (g16IntervalCnt+g16PositionCnt); j++)       //(J)chk every on interval
case and overlap booth case
               {
                    if (j == 0)
                    {
                         dRight = (g16arInterval[0]-1)+0.5;      //(J)first interval
                    }
```

```
                    else if (j == (gl6IntervalCnt-1))
                    {
                         dRight = (gl6arInterval[j] - 1)+0.5;     //(J)last interval
                    }
                    else if (j >= gl6IntervalCnt)                //(J)overlap case
                    {
                         if (gl6arBoothCntOfPosition[j - gl6IntervalCnt] < 2)
                         {
                              for (u16Booth = 0; u16Booth < gl6BoothCnt; u16Booth++)
                              {
                                  if( (gl6BoothIsOnLine[u16Booth]) && (gl6arBoothPositionIdx[u16Booth]
== (j - gl6IntervalCnt)))
                                        break;
                              }
                              dRight = ((double)gdarRight[u16Booth]) / 2;
                         }
                         else
                              dRight = 0;
                    }
                    else
                    {    //(J)put booth at middle of interval
                         dRight = ((double)gl6arInterval[j]) / 2;
                    }
                    if (dRight > dMaxRight)
                    {
                         u16IntervalIdx = j;
                         dMaxRight = dRight;
                    }
                    if (dRight == dMaxRight)         //(i)when there are two same dmaxright, choose the
random position
                    {
                         u16OrgIntervalIdx = u16IntervalIdx;
                         u16IntervalIdx = j;
                         k = rand()%(100);
                         t = k % 2;
                         if (t = 0)
                         {
                              u16IntervalIdx = u16OrgIntervalIdx;
                         }
                         else
                         {
                              u16IntervalIdx = j;
                         }
                    }
```

```
                }
                if (dMaxRight <= dOrgRight)
                {    //(J)right doesn't increase after change to new position --> put booth back to original position
                     u16PositonIdx = 0;
                     while ((g16arPosition[u16PositonIdx] < u16OrgPosition) && (u16PositonIdx<g16PositionCnt))    u16PositonIdx++;    //(J)find the first position which is on the right hand of the original position
                     g16arBoothPositionIdx[i] = u16PositonIdx;

                     if (g16arPosition[u16PositonIdx] == u16OrgPosition)
                     {
                          g16arBoothCntOfPosition[u16PositonIdx]++;    //(J)booth cnt at position +1
                     }
                     else
                     {
                          for (j = 0; j <= g16BoothCnt; j++)
                          {
                               if ((g16BoothIsOnLine[j]) && (g16arBoothPositionIdx[j] >= u16PositonIdx))
                                    g16arBoothPositionIdx[j]++;              //(J)increase position index of all booth on the right hand of the inserted booth
                          }
                          for (j = g16PositionCnt; j > u16PositonIdx; j--)
                          {
                               g16arPosition[j] = g16arPosition[j - 1];
                               g16arBoothCntOfPosition[j] = g16arBoothCntOfPosition[j - 1];
                          }
                          g16arPosition[j] = u16OrgPosition;
                          g16arBoothCntOfPosition[j] = 1;
                          g16PositionCnt++;
                          for (j = 0; j <=g16PositionCnt; j++)
                          {
                               if (j == 0)
                               {
                                    g16arInterval[0] = g16arPosition[0];
                               }
                               else if (j==g16PositionCnt)
                               {
                                    g16arInterval[j] = g16TotalLength - g16arPosition[j - 1];
                               }
                               else
                               {
                                    g16arInterval[j] = g16arPosition[j] - g16arPosition[j - 1];
                               }
```

```
                    }
                    gl6IntervalCnt++;
               }
          }
          else
          {
               //(J)insert booth to line
               ul6ChangePosCnt++;
               if (ul6IntervalIdx == 0)
               {
                    for (j = gl6PositionCnt; j > 0; j--)   //(J)shift position to right
                    {
                         gl6arPosition[j] = gl6arPosition[j - 1];
                         gl6arBoothCntOfPosition[j] = gl6arBoothCntOfPosition[j - 1];
                    }
                    gl6arPosition[0] = gl6arPosition[j + 1] - 1;
                    gl6arBoothCntOfPosition[0] = 1;
                    gl6PositionCnt++;
                    for (j = gl6IntervalCnt; j > 1; j--)
                    {
                         gl6arInterval[j] = gl6arInterval[j - 1];
                    }
                    gl6arInterval[1] = 1;
                    gl6arInterval[0] = gl6arInterval[0] - 1;
                    gl6IntervalCnt++;
                    for (j = 0; j <= gl6BoothCnt; j++)
                    {
                         if (gl6BoothIsOnLine[j])
                              gl6arBoothPositionIdx[j]++;             //(J)increase position
index of all booth
                    }
                    gl6arBoothPositionIdx[i] = 0; //(J)means booth_i at position_0
               }
               else if (ul6IntervalIdx == (gl6IntervalCnt - 1))
               {
                    gl6arBoothPositionIdx[i] = gl6PositionCnt;    //(J)booth at the last position
                    gl6arPosition[gl6PositionCnt] = gl6arPosition[gl6PositionCnt - 1] + 1;
                    gl6arBoothCntOfPosition[gl6PositionCnt] = 1;
                    gl6PositionCnt++;

                    gl6arInterval[gl6IntervalCnt] = gl6arInterval[gl6IntervalCnt - 1] - 1;
                    gl6arInterval[gl6IntervalCnt - 1] = 1;
                    gl6IntervalCnt++;
                    gl6arBoothPositionIdx[i] = gl6PositionCnt - 1;         //(J)means booth_i at
```

```
the last position
                    }
                    else if (u16IntervalIdx >= g16IntervalCnt)                    //(J)overlap case -->
interval count not change
                    {
                         u16PositonIdx =u16IntervalIdx - g16IntervalCnt;
                         g16arBoothCntOfPosition[u16PositonIdx]++;          //(J)booth cnt at position
+1
                         g16arBoothPositionIdx[i] = u16PositonIdx;
                    }
                    else //(j)u16IntervalIdx>0 && u16IntervalIdx< (g16IntervalCnt - 1)
                    {          //(J)inster booth in interval
                         for (j = 0; j <= g16BoothCnt; j++)
                         {
                              if ((g16BoothIsOnLine[j]) && (g16arBoothPositionIdx[j] >=
u16IntervalIdx))
                                   g16arBoothPositionIdx[j]++;              //(J)increase position
index of all booth
                         }
                         g16arBoothPositionIdx[i] = u16IntervalIdx;
                         for (j = g16IntervalCnt; j > (u16IntervalIdx + 1); j--)
                         {
                              g16arInterval[j] = g16arInterval[j - 1];
                              if ((j - u16IntervalIdx) >= 2)
                              {
                                   g16arPosition[j - 1] = g16arPosition[j - 2];
                                   g16arBoothCntOfPosition[j - 1] = g16arBoothCntOfPosition[j - 2];
                              }
                         }
                         g16arBoothCntOfPosition[j - 1] = 1;
                         seed = (unsigned)time(NULL); // 取得時間序列
                         srand(seed); // 以時間序列當亂數種子
                         j = rand() % (g16arInterval[u16IntervalIdx]-1);         //(J)for put the booth
at random position of the interval
                         j = j + 1;
                         //*****************************************************
                         g16arInterval[u16IntervalIdx + 1] = g16arInterval[u16IntervalIdx] - j;
                         g16arInterval[u16IntervalIdx] = j;
                         g16IntervalCnt++;
                         g16arPosition[u16IntervalIdx] = g16arPosition[u16IntervalIdx - 1] +
g16arInterval[u16IntervalIdx];
                         g16PositionCnt++;
                    }
               }
```

```
                g16BoothIsOnLine[i] = 1;
                g16CurrentNumOfBoothOnLine++;
                CalRightOfBoothOnLine();
            }
            //(J)display result
            printf("**********************************\n");
            printf("Total position count=%d \n", g16PositionCnt);
            for (i = 0; i < g16PositionCnt; i++)
                printf("Position[%d]=%d \n", i, g16arPosition[i]);
            printf("Total booth count=%d \n", g16BoothCnt);
            for (i = 0; i < g16BoothCnt; i++)
            {
                printf("Booth[%d] at Position[%d] has right of %6.4f\n", i, g16arBoothPositionIdx[i],
gdarRight[i]);
            }

            //(J)check stable
            if (u16ChangePosCnt==0)
            {
                bOk = 1;
                printf("The total length is %d\n", g16TotalLength);
                printf("All booths reach stable!\n");
                system("pause");
            }
            else
            {   //(J)backup right of all booth
                for (i = 0; i < g16BoothCnt; i++)
                {
                    gdarOldRight[i] = gdarRight[i];
                }
                printf("All booths are not stable yet, next round!\n");
                u16Round++;
            }
            printf(" \n");
        }
        return 0;
}
```

附錄三

一、無固定攤販的模型

(一)四家流動攤販

第一次

Round16 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[0] has right of 25.5000

Booth[1] at Position[2] has right of 24.7500

Booth[2] at Position[1] has right of 25.0000

Booth[3] at Position[2] has right of 24.7500

All booths reach stable!

第二次

Round15 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[2] has right of 24.7500

Booth[1] at Position[1] has right of 25.0000

Booth[2] at Position[2] has right of 24.7500

Booth[3] at Position[0] has right of 25.5000

All booths reach stable!

第三次

Round19 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[2] has right of 24.7500

Booth[1] at Position[0] has right of 25.5000

Booth[2] at Position[2] has right of 24.7500

Booth[3] at Position[1] has right of 25.0000

All booths reach stable!

第四次

Round18 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[1] has right of 25.0000

Booth[1] at Position[2] has right of 24.7500

Booth[2] at Position[0] has right of 25.5000

Booth[3] at Position[2] has right of 24.7500

All booths reach stable!

第五次

Round20 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[1] has right of 25.0000

Booth[1] at Position[2] has right of 24.7500

Booth[2] at Position[0] has right of 25.5000

Booth[3] at Position[2] has right of 24.7500

All booths reach stable!

第六次

Round18 start...

****

Total position count=3

Position[0]=25

Position[1]=26

Position[2]=75

Total booth count=4

Booth[0] at Position[0] has right of 25.5000

Booth[1] at Position[2] has right of 24.7500

Booth[2] at Position[1] has right of 25.0000

Booth[3] at Position[2] has right of 24.7500
All booths reach stable!

第七次
Round25 start...
****
Total position count=3
Position[0]=25
Position[1]=26
Position[2]=75
Total booth count=4
Booth[0] at Position[0] has right of 25.5000
Booth[1] at Position[2] has right of 24.7500
Booth[2] at Position[1] has right of 25.0000
Booth[3] at Position[2] has right of 24.7500
All booths reach stable!

第八次
Round21 start...
****
Total position count=3
Position[0]=25
Position[1]=26
Position[2]=75
Total booth count=4
Booth[0] at Position[2] has right of 24.7500
Booth[1] at Position[0] has right of 25.5000
Booth[2] at Position[2] has right of 24.7500
Booth[3] at Position[1] has right of 25.0000
All booths reach stable!

第九次
Round15 start...
****
Total position count=3
Position[0]=25
Position[1]=26
Position[2]=75
Total booth count=4
Booth[0] at Position[2] has right of 24.7500
Booth[1] at Position[1] has right of 25.0000
Booth[2] at Position[2] has right of 24.7500
Booth[3] at Position[0] has right of 25.5000
All booths reach stable!

第十次
Round541580 start...
****
Round20 start...
****
Total position count=3
Position[0]=25
Position[1]=26
Position[2]=75
Total booth count=4
Booth[0] at Position[0] has right of 25.5000
Booth[1] at Position[2] has right of 24.7500
Booth[2] at Position[1] has right of 25.0000
Booth[3] at Position[2] has right of 24.7500
All booths reach stable!

(二)五家流動攤販

第一次
Round4387 start...
****
Total position count=4
Position[0]=17
Position[1]=49
Position[2]=82
Position[3]=83
Total booth count=5
Booth[0] at Position[3] has right of 17.5000
Booth[1] at Position[0] has right of 16.5000
Booth[2] at Position[1] has right of 32.5000
Booth[3] at Position[2] has right of 17.0000
Booth[4] at Position[0] has right of 16.5000
All booths reach stable!

第二次
734104 無均衡,集中於中央

第三次
711115 無均衡,集中於中央

第四次
771674 無均衡,集中於中央

第五次

721622 無均衡,集中於中央

第六次

750987 無均衡,集中於中央

第七次

752297 無均衡,集中於中央

第八次

736691 無均衡,集中於中央

第九次

770769 無均衡,集中於中央

第十次

722450 無均衡,集中於中央

(三)六家固定攤販

第一次

Round18 start...

****

Total position count=4

Position[0]=15

Position[1]=44

Position[2]=57

Position[3]=85

Total booth count=6

Booth[0] at Position[3] has right of 14.5000

Booth[1] at Position[0] has right of 14.7500

Booth[2] at Position[1] has right of 21.0000

Booth[3] at Position[3] has right of 14.5000

Booth[4] at Position[0] has right of 14.7500

Booth[5] at Position[2] has right of 20.5000

All booths reach stable!

第二次

Round535 start...

****

Total position count=6

Position[0]=15

Position[1]=16

Position[2]=45

Position[3]=54

Position[4]=84

Position[5]=85

Total booth count=6

Booth[0] at Position[2] has right of 19.0000

Booth[1] at Position[0] has right of 15.5000

Booth[2] at Position[1] has right of 15.0000

Booth[3] at Position[3] has right of 19.5000

Booth[4] at Position[4] has right of 15.5000

Booth[5] at Position[5] has right of 15.5000

All booths reach stable!

第三次

765701 攤販無均衡，集中在沙灘中央

第四次

Round451 start...

****

Total position count=5

Position[0]=14

Position[1]=41

Position[2]=58

Position[3]=85

Position[4]=86

Total booth count=6

Booth[0] at Position[3] has right of 14.0000

Booth[1] at Position[0] has right of 13.7500

Booth[2] at Position[4] has right of 14.5000

Booth[3] at Position[1] has right of 22.0000

Booth[4] at Position[2] has right of 22.0000

Booth[5] at Position[0] has right of 13.7500

All booths reach stable!

第五次

Round636 start...

****

Total position count=4

Position[0]=17

Position[1]=49

Position[2]=51

Position[3]=83

Total booth count=6

Booth[0] at Position[0] has right of 16.5000

Booth[1] at Position[3] has right of 16.5000

Booth[2] at Position[3] has right of 16.5000

Booth[3] at Position[0] has right of 16.5000

Booth[4] at Position[1] has right of 17.0000
Booth[5] at Position[2] has right of 17.0000
All booths reach stable!

第六次
723659 無均衡,集中於中央

第七次
784202 無均衡,集中於中央

第八次
Round1354 start...
****
Total position count=4
Position[0]=14
Position[1]=40
Position[2]=60
Position[3]=86
Total booth count=6
Booth[0] at Position[2] has right of 23.0000
Booth[1] at Position[3] has right of 13.5000
Booth[2] at Position[0] has right of 13.5000
Booth[3] at Position[1] has right of 23.0000
Booth[4] at Position[3] has right of 13.5000
Booth[5] at Position[0] has right of 13.5000
All booths reach stable!

第九次
Round17 start...
****
Total position count=4
Position[0]=15
Position[1]=44
Position[2]=57
Position[3]=85
Total booth count=6
Booth[0] at Position[3] has right of 14.5000
Booth[1] at Position[2] has right of 20.5000
Booth[2] at Position[0] has right of 14.7500
Booth[3] at Position[3] has right of 14.5000
Booth[4] at Position[0] has right of 14.7500
Booth[5] at Position[1] has right of 21.0000
All booths reach stable!

第十次
Round15 start...
****
Total position count=6
Position[0]=16
Position[1]=17
Position[2]=48
Position[3]=51
Position[4]=83
Position[5]=84
Total booth count=6
Booth[0] at Position[5] has right of 16.5000
Booth[1] at Position[0] has right of 16.5000
Booth[2] at Position[1] has right of 16.0000
Booth[3] at Position[2] has right of 17.0000
Booth[4] at Position[3] has right of 17.5000
Booth[5] at Position[4] has right of 16.5000
All booths reach stable!

(四)七家流動攤販

第一次
700217 無均衡,集中於中央

第二次
Round521 start...
****
Total position count=6
Position[0]=12
Position[1]=13
Position[2]=36
Position[3]=56
Position[4]=65
Position[5]=88
Total booth count=7
Booth[0] at Position[5] has right of 11.7500
Booth[1] at Position[2] has right of 21.5000
Booth[2] at Position[3] has right of 14.5000
Booth[3] at Position[0] has right of 12.5000
Booth[4] at Position[4] has right of 16.0000
Booth[5] at Position[5] has right of 11.7500
Booth[6] at Position[1] has right of 12.0000
All booths reach stable!

第三次

794025 無均衡,集中於中央

第四次

Round527 start...

****

Total position count=7

Position[0]=12

Position[1]=13

Position[2]=36

Position[3]=55

Position[4]=64

Position[5]=87

Position[6]=88

Total booth count=7

Booth[0] at Position[1] has right of 12.0000

Booth[1] at Position[4] has right of 16.0000

Booth[2] at Position[6] has right of 12.5000

Booth[3] at Position[2] has right of 21.0000

Booth[4] at Position[0] has right of 12.5000

Booth[5] at Position[3] has right of 14.0000

Booth[6] at Position[5] has right of 12.0000

All booths reach stable!

第五次

743186 無均衡,集中於中央

第六次

751122 無均衡,集中於中央

第七次

Round1223 start...

****

Total position count=5

Position[0]=12

Position[1]=35

Position[2]=45

Position[3]=65

Position[4]=88

Total booth count=7

Booth[0] at Position[1] has right of 16.5000

Booth[1] at Position[0] has right of 11.7500

Booth[2] at Position[3] has right of 21.5000

Booth[3] at Position[4] has right of 11.7500

Booth[4] at Position[0] has right of 11.7500

Booth[5] at Position[2] has right of 15.0000

Booth[6] at Position[4] has right of 11.7500

All booths reach stable!

第八次

Round977 start...

****

Total position count=6

Position[0]=11

Position[1]=12

Position[2]=34

Position[3]=47

Position[4]=67

Position[5]=89

Total booth count=7

Booth[0] at Position[4] has right of 21.0000

Booth[1] at Position[2] has right of 17.5000

Booth[2] at Position[5] has right of 11.0000

Booth[3] at Position[1] has right of 11.5000

Booth[4] at Position[0] has right of 11.5000

Booth[5] at Position[5] has right of 11.0000

Booth[6] at Position[3] has right of 16.5000

All booths reach stable!

第九次

Round1015 start...

****

Total position count=7

Position[0]=12

Position[1]=13

Position[2]=36

Position[3]=47

Position[4]=64

Position[5]=87

Position[6]=88

Total booth count=7

Booth[0] at Position[6] has right of 12.5000

Booth[1] at Position[0] has right of 12.5000

Booth[2] at Position[3] has right of 14.0000

Booth[3] at Position[5] has right of 12.0000

Booth[4] at Position[2] has right of 17.0000

Booth[5] at Position[4] has right of 20.0000

Booth[6] at Position[1] has right of 12.0000

All booths reach stable!

第十次

Round2560 start...

****

Total position count=7

Position[0]=11

Position[1]=12

Position[2]=33

Position[3]=49

Position[4]=67

Position[5]=88

Position[6]=89

Total booth count=7

Booth[0] at Position[5] has right of 11.0000

Booth[1] at Position[3] has right of 17.0000

Booth[2] at Position[4] has right of 19.5000

Booth[3] at Position[1] has right of 11.0000

Booth[4] at Position[6] has right of 11.5000

Booth[5] at Position[0] has right of 11.5000

Booth[6] at Position[2] has right of 18.5000

All booths reach stable!

(五)八家流動攤販

第一次

Round15 start...

****

Total position count=6

Position[0]=12

Position[1]=34

Position[2]=38

Position[3]=61

Position[4]=65

Position[5]=88

Total booth count=8

Booth[0] at Position[5] has right of 11.7500

Booth[1] at Position[2] has right of 13.5000

Booth[2] at Position[5] has right of 11.7500

Booth[3] at Position[0] has right of 11.5000

Booth[4] at Position[0] has right of 11.5000

Booth[5] at Position[4] has right of 13.5000

Booth[6] at Position[3] has right of 13.5000

Booth[7] at Position[1] has right of 13.0000

All booths reach stable!

第二次

Round48 start...

****

Total position count=7

Position[0]=11

Position[1]=12

Position[2]=33

Position[3]=44

Position[4]=63

Position[5]=69

Position[6]=89

Total booth count=8

Booth[0] at Position[1] has right of 11.0000

Booth[1] at Position[0] has right of 11.5000

Booth[2] at Position[4] has right of 12.5000

Booth[3] at Position[2] has right of 16.0000

Booth[4] at Position[3] has right of 15.0000

Booth[5] at Position[5] has right of 13.0000

Booth[6] at Position[6] has right of 10.5000

Booth[7] at Position[6] has right of 10.5000

All booths reach stable!

第三次

751324 無均衡,集中於中央

第四次

Round5542 start...

****

Total position count=6

Position[0]=11

Position[1]=32

Position[2]=43

Position[3]=64

Position[4]=68

Position[5]=89

Total booth count=8

Booth[0] at Position[0] has right of 10.7500

Booth[1] at Position[2] has right of 16.0000

Booth[2] at Position[5] has right of 10.7500

Booth[3] at Position[5] has right of 10.7500

Booth[4] at Position[1] has right of 16.0000

Booth[5] at Position[4] has right of 12.5000

Booth[6] at Position[3] has right of 12.5000

Booth[7] at Position[0] has right of 10.7500

All booths reach stable!

第五次

Round162 start...

****

Total position count=6

Position[0]=12

Position[1]=34

Position[2]=36

Position[3]=57

Position[4]=65

Position[5]=88

Total booth count=8

Booth[0] at Position[4] has right of 15.5000

Booth[1] at Position[2] has right of 11.5000

Booth[2] at Position[0] has right of 11.5000

Booth[3] at Position[0] has right of 11.5000

Booth[4] at Position[1] has right of 12.0000

Booth[5] at Position[5] has right of 11.7500

Booth[6] at Position[5] has right of 11.7500

Booth[7] at Position[3] has right of 14.5000

All booths reach stable!

第六次

Round344 start...

****

Total position count=7

Position[0]=10

Position[1]=29

Position[2]=41

Position[3]=60

Position[4]=70

Position[5]=89

Position[6]=90

Total booth count=8

Booth[0] at Position[0] has right of 9.7500

Booth[1] at Position[1] has right of 15.5000

Booth[2] at Position[6] has right of 10.5000

Booth[3] at Position[2] has right of 15.5000

Booth[4] at Position[3] has right of 14.5000

Booth[5] at Position[0] has right of 9.7500

Booth[6] at Position[5] has right of 10.0000

Booth[7] at Position[4] has right of 14.5000

All booths reach stable!

第七次

759190 無均衡,集中於中央

第八次

Round18 start...

****

Total position count=8

Position[0]=10

Position[1]=11

Position[2]=30

Position[3]=40

Position[4]=58

Position[5]=70

Position[6]=89

Position[7]=90

Total booth count=8

Booth[0] at Position[2] has right of 14.5000

Booth[1] at Position[1] has right of 10.0000

Booth[2] at Position[7] has right of 10.5000

Booth[3] at Position[5] has right of 15.5000

Booth[4] at Position[3] has right of 14.0000

Booth[5] at Position[0] has right of 10.5000

Booth[6] at Position[4] has right of 15.0000

Booth[7] at Position[6] has right of 10.0000

All booths reach stable!

第九次

Round2907 start...

****

Total position count=6

Position[0]=11

Position[1]=32

Position[2]=44

Position[3]=64

Position[4]=68

Position[5]=89

Total booth count=8

Booth[0] at Position[3] has right of 12.0000

Booth[1] at Position[1] has right of 16.5000

Booth[2] at Position[5] has right of 10.7500

Booth[3] at Position[0] has right of 10.7500

Booth[4] at Position[5] has right of 10.7500

Booth[5] at Position[4] has right of 12.5000

Booth[6] at Position[2] has right of 16.0000
Booth[7] at Position[0] has right of 10.7500
All booths reach stable!

第十次

Round18 start...
****
Total position count=6
Position[0]=10
Position[1]=28
Position[2]=47
Position[3]=63
Position[4]=72
Position[5]=90
Total booth count=8
Booth[0] at Position[5] has right of 9.5000
Booth[1] at Position[4] has right of 13.5000
Booth[2] at Position[0] has right of 9.5000
Booth[3] at Position[3] has right of 12.5000
Booth[4] at Position[0] has right of 9.5000
Booth[5] at Position[5] has right of 9.5000
Booth[6] at Position[2] has right of 17.5000
Booth[7] at Position[1] has right of 18.5000
All booths reach stable!

(六)九家流動攤販

第一次

711254 無均衡,集中於中央

第二次

Round222 start...
****
Total position count=8
Position[0]=9
Position[1]=26
Position[2]=34
Position[3]=50
Position[4]=65
Position[5]=73
Position[6]=90
Position[7]=91
Total booth count=9
Booth[0] at Position[0] has right of 8.7500
Booth[1] at Position[4] has right of 11.5000
Booth[2] at Position[7] has right of 9.5000
Booth[3] at Position[0] has right of 8.7500
Booth[4] at Position[6] has right of 9.0000
Booth[5] at Position[3] has right of 15.5000
Booth[6] at Position[2] has right of 12.0000
Booth[7] at Position[5] has right of 12.5000
Booth[8] at Position[1] has right of 12.5000
All booths reach stable!

第三次

Round501 start...
****
Total position count=7
Position[0]=9
Position[1]=25
Position[2]=31
Position[3]=45
Position[4]=62
Position[5]=75
Position[6]=91
Total booth count=9
Booth[0] at Position[6] has right of 8.5000
Booth[1] at Position[0] has right of 8.5000
Booth[2] at Position[6] has right of 8.5000
Booth[3] at Position[3] has right of 15.5000
Booth[4] at Position[0] has right of 8.5000
Booth[5] at Position[5] has right of 14.5000
Booth[6] at Position[2] has right of 10.0000
Booth[7] at Position[4] has right of 15.0000
Booth[8] at Position[1] has right of 11.0000
All booths reach stable!

第四次

Round1769 start...
****
Total position count=7
Position[0]=9
Position[1]=26
Position[2]=29
Position[3]=46
Position[4]=58
Position[5]=75
Position[6]=91
Total booth count=9

Booth[0] at Position[5] has right of 16.5000
Booth[1] at Position[0] has right of 8.7500
Booth[2] at Position[6] has right of 8.5000
Booth[3] at Position[4] has right of 14.5000
Booth[4] at Position[2] has right of 10.0000
Booth[5] at Position[6] has right of 8.5000
Booth[6] at Position[0] has right of 8.7500
Booth[7] at Position[3] has right of 14.5000
Booth[8] at Position[1] has right of 10.0000
All booths reach stable!

第五次

Round528 start...
****
Total position count=8
Position[0]=8
Position[1]=9
Position[2]=24
Position[3]=39
Position[4]=47
Position[5]=62
Position[6]=77
Position[7]=92
Total booth count=9
Booth[0] at Position[5] has right of 15.0000
Booth[1] at Position[1] has right of 8.0000
Booth[2] at Position[4] has right of 11.5000
Booth[3] at Position[2] has right of 15.0000
Booth[4] at Position[7] has right of 7.7500
Booth[5] at Position[0] has right of 8.5000
Booth[6] at Position[7] has right of 7.7500
Booth[7] at Position[3] has right of 11.5000
Booth[8] at Position[6] has right of 15.0000
All booths reach stable!

第六次

754150 無均衡,集中於中央

第七次

Round1344 start...
****
Total position count=8
Position[0]=10
Position[1]=29
Position[2]=36
Position[3]=53
Position[4]=56
Position[5]=72
Position[6]=90
Position[7]=91
Total booth count=9
Booth[0] at Position[7] has right of 9.5000
Booth[1] at Position[0] has right of 9.7500
Booth[2] at Position[1] has right of 13.0000
Booth[3] at Position[3] has right of 10.0000
Booth[4] at Position[4] has right of 9.5000
Booth[5] at Position[5] has right of 17.0000
Booth[6] at Position[2] has right of 12.0000
Booth[7] at Position[0] has right of 9.7500
Booth[8] at Position[6] has right of 9.5000
All booths reach stable!

第八次

Round1558 start...
****
Total position count=7
Position[0]=9
Position[1]=26
Position[2]=37
Position[3]=52
Position[4]=60
Position[5]=74
Position[6]=91
Total booth count=9
Booth[0] at Position[6] has right of 8.7500
Booth[1] at Position[2] has right of 13.0000
Booth[2] at Position[1] has right of 14.0000
Booth[3] at Position[0] has right of 8.7500
Booth[4] at Position[4] has right of 11.0000
Booth[5] at Position[6] has right of 8.7500
Booth[6] at Position[3] has right of 11.5000
Booth[7] at Position[5] has right of 15.5000
Booth[8] at Position[0] has right of 8.7500
All booths reach stable!

第九次

Round2621 start...
****

Total position count=8
Position[0]=8
Position[1]=9
Position[2]=24
Position[3]=39
Position[4]=53
Position[5]=64
Position[6]=78
Position[7]=92
Total booth count=9
Booth[0] at Position[5] has right of 12.5000
Booth[1] at Position[6] has right of 14.0000
Booth[2] at Position[3] has right of 14.5000
Booth[3] at Position[1] has right of 8.0000
Booth[4] at Position[7] has right of 7.5000
Booth[5] at Position[4] has right of 12.5000
Booth[6] at Position[2] has right of 15.0000
Booth[7] at Position[0] has right of 8.5000
Booth[8] at Position[7] has right of 7.5000
All booths reach stable!

第十次
727911 無均衡,
分散於 9,10,28,41,52,70,88,89,90

(七)十家流動攤販

第一次
Round16 start...
****
Total position count=9
Position[0]=8
Position[1]=9
Position[2]=24
Position[3]=35
Position[4]=48
Position[5]=53
Position[6]=64
Position[7]=78
Position[8]=92
Total booth count=10
Booth[0] at Position[6] has right of 12.5000
Booth[1] at Position[5] has right of 8.0000
Booth[2] at Position[8] has right of 7.5000
Booth[3] at Position[3] has right of 12.0000
Booth[4] at Position[1] has right of 8.0000
Booth[5] at Position[7] has right of 14.0000
Booth[6] at Position[2] has right of 13.0000
Booth[7] at Position[8] has right of 7.5000
Booth[8] at Position[0] has right of 8.5000
Booth[9] at Position[4] has right of 9.0000
All booths reach stable!

第二次
733257 無均衡,集中於中央

第三次
Round60 start...
****
Total position count=9
Position[0]=7
Position[1]=8
Position[2]=22
Position[3]=34
Position[4]=46
Position[5]=59
Position[6]=71
Position[7]=79
Position[8]=93
Total booth count=10
Booth[0] at Position[0] has right of 7.5000
Booth[1] at Position[2] has right of 13.0000
Booth[2] at Position[8] has right of 7.0000
Booth[3] at Position[6] has right of 10.0000
Booth[4] at Position[7] has right of 11.0000
Booth[5] at Position[8] has right of 7.0000
Booth[6] at Position[1] has right of 7.5000
Booth[7] at Position[3] has right of 12.0000
Booth[8] at Position[5] has right of 12.5000
Booth[9] at Position[4] has right of 12.5000
All booths reach stable!

第四次
707717 無均衡,集中於中央

第五次
Round1790 start...
****

Total position count=9
Position[0]=8
Position[1]=23
Position[2]=35
Position[3]=45
Position[4]=57
Position[5]=71
Position[6]=76
Position[7]=91
Position[8]=92
Total booth count=10
Booth[0] at Position[0] has right of 7.7500
Booth[1] at Position[8] has right of 8.5000
Booth[2] at Position[3] has right of 11.0000
Booth[3] at Position[1] has right of 13.5000
Booth[4] at Position[0] has right of 7.7500
Booth[5] at Position[7] has right of 8.0000
Booth[6] at Position[4] has right of 13.0000
Booth[7] at Position[5] has right of 9.5000
Booth[8] at Position[2] has right of 11.0000
Booth[9] at Position[6] has right of 10.0000
All booths reach stable!

第六次
728968 無均衡,集中於中央

第七次
Round186 start...
****
Total position count=10
Position[0]=8
Position[1]=9
Position[2]=24
Position[3]=34
Position[4]=41
Position[5]=55
Position[6]=62
Position[7]=76
Position[8]=91
Position[9]=92
Total booth count=10
Booth[0] at Position[2] has right of 12.5000
Booth[1] at Position[7] has right of 14.5000
Booth[2] at Position[4] has right of 10.5000
Booth[3] at Position[5] has right of 10.5000
Booth[4] at Position[0] has right of 8.5000
Booth[5] at Position[8] has right of 8.0000
Booth[6] at Position[1] has right of 8.0000
Booth[7] at Position[9] has right of 8.5000
Booth[8] at Position[6] has right of 10.5000
Booth[9] at Position[3] has right of 8.5000
All booths reach stable!

第八次
Round2045 start...
****
Total position count=9
Position[0]=8
Position[1]=9
Position[2]=24
Position[3]=31
Position[4]=45
Position[5]=59
Position[6]=71
Position[7]=78
Position[8]=92
Total booth count=10
Booth[0] at Position[4] has right of 14.0000
Booth[1] at Position[7] has right of 10.5000
Booth[2] at Position[8] has right of 7.5000
Booth[3] at Position[0] has right of 8.5000
Booth[4] at Position[8] has right of 7.5000
Booth[5] at Position[6] has right of 9.5000
Booth[6] at Position[2] has right of 11.0000
Booth[7] at Position[5] has right of 13.0000
Booth[8] at Position[1] has right of 8.0000
Booth[9] at Position[3] has right of 10.5000
All booths reach stable!

第九次
Round190 start...
****
Total position count=8
Position[0]=8
Position[1]=23
Position[2]=38
Position[3]=52
Position[4]=60

Position[5]=72
Position[6]=78
Position[7]=92
Total booth count=10
Booth[0] at Position[5] has right of 9.0000
Booth[1] at Position[7] has right of 7.5000
Booth[2] at Position[0] has right of 7.7500
Booth[3] at Position[1] has right of 15.0000
Booth[4] at Position[0] has right of 7.7500
Booth[5] at Position[2] has right of 14.5000
Booth[6] at Position[7] has right of 7.5000
Booth[7] at Position[4] has right of 10.0000
Booth[8] at Position[6] has right of 10.0000
Booth[9] at Position[3] has right of 11.0000
All booths reach stable!

第十次

Round142 start...
****
Total position count=9
Position[0]=8
Position[1]=23
Position[2]=31
Position[3]=43
Position[4]=54
Position[5]=68
Position[6]=76
Position[7]=91
Position[8]=92
Total booth count=10
Booth[0] at Position[0] has right of 7.7500
Booth[1] at Position[5] has right of 11.0000
Booth[2] at Position[6] has right of 11.5000
Booth[3] at Position[2] has right of 10.0000
Booth[4] at Position[7] has right of 8.0000
Booth[5] at Position[0] has right of 7.7500
Booth[6] at Position[1] has right of 11.5000
Booth[7] at Position[4] has right of 12.5000
Booth[8] at Position[3] has right of 11.5000
Booth[9] at Position[8] has right of 8.5000
All booths reach stable