第十五屆旺宏科學獎

成果報告書

參賽編號：SA15-011

作品名稱：格子點上選擇位置之性質研究

姓名：鄭天鈞

關鍵字：組合數學、遞迴數列、數學歸納法

摘要

本文從一個現象開始:一群陌生人在坐一排座位時常不與他人鄰座,對此現象建立了一對數學模型探討。我們主要研究一維的情況下,能坐下的人數、以及若第一個人策略性的選擇,其至多能坐下的人數及策略、和隨著椅子數的增加至多能坐下的人數之變化。進一步地,我們將其中一個模型推廣到二維的情況,推導出在正方形的格子點中能坐下的人數。過程中我們先做一些預處理得到遞迴函式,然而常會遇到帶有高斯符號或是分段的遞迴函式,因此我們主要採取的手法是先猜出遞迴式的通式再歸納,或是利用調整法。最後雖然礙於二維以上的一般情況會因沒有規律而導不出完整的結論,但本文中我們做出了一些部分結果,未來也會試圖往更廣的情況突破。

目錄

壹、研究動機

在搭乘捷運的時候，有時會出現一種場景:雖有空位但卻有人站著。通常是因為不好意思坐在陌生人的旁邊。這種情況也會發生在演講廳或是電影院等公共場所。偶然地有一天看到別人提出了類似的問題模型，我想到這很像前面提到的情況。於是我想探討:如果大家都這樣坐，那究竟坐得下多少人?於是我想探討在這種情況下，能在一、二維空間下的格子點坐下多少人?

貳、研究目的

在一個 n 維度 $\mathbb{R}^n$ 空間中考慮區間 $I_n=[1,a_1]\times[1,a_2]\times[1,a_3]...\times[1,a_n]$，其中 $a_k\in N$，$\forall 1\le k\le n$，即 $I_n=\{(x_1,x_2,x_3...x_n)|\ x_k\in\mathbb{N},1\le x_k\le a_k,1\le k\le n\}$，我們假定在 I_n 中每個格子點有一個座位，依以下規則選擇座位:除第一人外，其餘每一個人依序選擇座位，使得所選擇的座位對於已被選取的座位之”最小點距”達到最大值，且規定不得和之前已選定位置者鄰座(即哈密頓距離為 1)。

我們主要目標是探討下列四個研究目的:

研究目的一: 當第一個人就近選擇最靠近原點的位置坐時，能坐下幾個人?

研究目的二: 當第一個人可以任意決定位置時，第一個人應該坐在哪裡才能使得總共坐下最多人?

研究目的三: 承上研究目的二條件下，決定能夠坐下總共多少人?

研究目的四: 承上研究目的二條件下，若增加椅子的數量，椅子數與至多能坐下的人數之關係為何?

由於在這個問題中可能會遇到有多個位置 $S_1,S_2,\cdots,S_k\in I_n$ 使得”最小點距”達到最大值，此時我們規定下列兩個情形的座位選擇方式：

若點 $S_1,S_2,\cdots,S_k\in I_n$，且 $\|S_1\|_1=\|S_2\|_1=\cdots=\|S_i\|_1<\|S_{i+1}\|_1\le\|S_{i+2}\|_1\le\cdots\le\|S_k\|_1$，則

(1) 從 $S_1,S_2,\cdots,S_i$ 中選擇出一點，比較第一分量小者優先，又相同時第二分量小者優先，依此類推，**稱此情況為第一種情形。**

(2)視為有一批人，同時選定所有那樣的位置並坐下。

令 $S_1,S_2,\cdots,S_k$ 皆被同時選取，若 $S_1,S_2,\cdots,S_k$ 部分點相鄰時，我們允許同次選擇者鄰座，**稱此情況為第二種情形。**

參、研究設備及器材

紙、筆、電腦、Code::Blocks、Microsoft Office Word、Mathtype。

肆、名詞定義與解釋

一、哈密頓距離

在一個 n 維度 $\mathbb{R}^n$ 空間中，若 $A(a_1,a_2,a_3,...a_n),B(b_1,b_2,b_3...b_n)\in\mathbb{R}^n$ 的**哈密頓距離**為

$\|A-B\|_1=\sum_{i=1}^{n}|a_i-b_i|$ 。

二、最小點距

定義一個點 P 到其他點 $Q_1,Q_2,...,Q_k$ 的距離為 $\min_{1\le i\le k}\{\|P-Q_i\|_1\}$，稱為 P 點的「**最小點距**」。

三、最佳位置

當第一個人所坐的位置能使得其他人坐下的總人數最多時，我們稱第一個人所在位置為「**最佳位置**」。

四、相鄰的格子與鄰居

兩個格子被視為是相鄰的，如果他們的哈密頓距離為 1。

五、鄰居

在一維數線 $\mathbb{R}^1$ 上，兩個人視為是鄰居，如果在當時他們之間沒有其他人。

六、賦值法

將問題的某些變量賦予數值以進行運算，觀察數值的變化，以代表整體。

七、調整法

將問題的所有情況經過步步調整歸依為單一情況，簡化一般性的問題為特例。

八、強數學歸納法

先奠基驗證 $n=k_0$ 的情況，在假設 $n<k$ 時成立，證明 $n=k$ 時成立。如此可得到對於 $n\ge k_0,n\in\mathbb{N}$ 時成立。

九、高斯符號

下高斯符號：記 $\lfloor x\rfloor$ 表示不大於 x 的最大整數值。

上高斯符號：記 $\lceil x\rceil$ 表示不小於 x 的最小整數值。

十、討論的情況簡記

在問題討論中我們考慮在 n 維度 $\mathbb{R}^n$ 空間裡區間 $I_n=[1,a_1]\times[1,a_2]\times[1,a_3]...\times[1,a_n]$ 中的格子點做為選擇位置的討論，將此情況簡記做 $P(n,a_1,a_2,a_3...a_n)$ 。

當第一個人的位置在 $(1,1,..1)$ ，記做**角落情況**。

當 $a_1=a_2=...=a_n$, 記做**正 n 維情況**。

本文中所討論的記數符號分別如下:

(1)坐下的人個數在第一種情形記作 $f(a_1,a_2,..a_n)$ 或是 $f(A)$ ，第二種情形則為 $e(a_1,a_2,..a_n)$ 或是 $e(A)$ 。

(2)當第一個人坐角落時能坐下的人數在第一種情形記為 $c(a_1,a_2,...a_n)$ 或 $c(A)$ ，在第二種情形則為 $d(a_1,a_2,...a_n)$ 或 $d(A)$ 。

(3)一個條件下最多人坐下的個數在第一種情形為 $g(a_1,a_2...a_n)$ 或 $g(A)$ ，在第二種情形則為 $h(a_1,a_2...a_n)$ 或 $h(A)$ 。

伍、研究過程

[**第一種情形**]首先我們先討論一維度 $I_1=[1,n]$，即 $P(1,n)$.

我們從最簡單的情況開始討論，然後先給出一些基本性質。

[性質 1] $\left\lceil \frac{n}{2} \right\rceil \geq f(n) \geq \left\lceil \frac{n}{3} \right\rceil$.

證明:

左式:考慮賦值法，對被填入的每一個座位賦值 1，其左右兩側賦值 $\frac{1}{2}$。於是能填入的數值上限為第 0 和第 $n+1$ 格(假想存在)至多各 $\frac{1}{2}$。對於 $1 \leq i \leq n$，第 i 格至多填入 1。所以總共至多能填 $\frac{1}{2}\cdot 2+1\cdot n=n+1$。然而每填一個位置都使數值增加 $1+\frac{1}{2}\cdot 2=2$，所以 $f(n) \leq \left\lfloor \frac{n+1}{2} \right\rfloor \leq \left\lceil \frac{n}{2} \right\rceil$.

右式:由於每次填入座位至多減少 3 個可填座位(本身和相鄰兩個)，故 $f(n) \geq \left\lceil \frac{n}{3} \right\rceil$，否則將仍可再填。故得證。

為了簡化問題，以下我們先考慮第一個人坐在角落的情況。

[**角落情況**]

每次第二個填的為第 n 格，接著為第 $\left\lfloor \frac{n+1}{2} \right\rfloor$ 格。容易想到最平均分布的情況是與 2 的冪次有關的。此時會注意到:第 $\left\lfloor \frac{n+1}{2} \right\rfloor$ 格把長條座位切成兩段，而前後兩段分別正如當初所填的，第一格和最後一格。這讓我們想到了遞迴公式，猜測

$c(n)=c(\left\lfloor \frac{n+1}{2} \right\rfloor)+c(n-\left\lfloor \frac{n+1}{2} \right\rfloor+1)$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$c(n)$	1	1	2	2	3	3	3	4	5	5	5	5	5	6	7	8	9	9	9	9

[回到一般情況]

[性質 2]關於一個點對所有點的最小點距，事實上只和左右兩側最接近它已被填入的位置有關。

而這是顯然的，但這引發我想到或許可以不必按照順序坐座位，而是只要確定其坐定時的鄰居就可以了。

[推論 3]只要一個人C預定要坐下時的左右鄰居都已坐定，分別在$A(a),B(b)$。則C的位置便可坐定，且其位置為那兩個人的中點$\left\lfloor \frac{a+b}{2} \right\rfloor$.

[引理 4]我們可以每次任找兩個鄰居(P,Q)構成區間α(特別的，也可以找最左或最右的人和牆壁)，令一個人在區間內坐到合理的位置T(中點或靠牆)。如此也不會影響結果。

證明:

假設依照原本的方法有人左下的位置集合為A，引理 4 的方法為B。我們要證明$A=B$。

($B\subseteq A$):

由推論 3，我們僅須證明位置T能被坐下且坐下時其左右鄰居必為P,Q。

然而除非與T相鄰的格子被佔據，否則T能被坐下。而與其相鄰的格子必屬於α，又其左右鄰居為P,Q的充要條件為T是α中第一個坐下的人，而確實如此，故得證。

($A\subseteq B$):

已知$B\subseteq A$，反設A中有一元素F不在B中。設F的左右鄰居為G,H。於是，F格與G,H的距離大於 1，從而由性質 2 知F與B的最小點距大於 1。這是一個不合法的狀態，矛盾，故得證。

於是我們得到了一個重要的結論。

[推論 5]若第一個人坐在座標a的地方，則$f(n)=f(a)+f(n-a+1)-1$.

由推論 5 便能得到我們前面的猜測。

[推論 6] $c(n)=c(\left\lfloor\frac{n+1}{2}\right\rfloor)+c(n-\left\lfloor\frac{n+1}{2}\right\rfloor+1)-1.$

[角落情況]

有了推論 6 的遞迴式，我們可以列出如下的表格。

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$c(n)$	1	1	2	2	3	3	3	4	5	5	5	5	5	6	7	8	9	9
n	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
$c(n)$	9	9	9	9	9	9	9	10	11	12	13	14	15	16	17	17	17	17

由性質 1 知 $c(n)\le\left\lceil\frac{n}{2}\right\rceil=\left\lfloor\frac{n+1}{2}\right\rfloor\le\frac{n+1}{2}$，先試著找出使等號成立的 n 值。

這樣的 n 值為 $1, 3, 5, 9, 17, 33\cdots$，會發現當 $n\ge 3$ 時這個數列是 $2^k+1\ (k\in\mathbb{N})$.

[引理 7] $c(2^k+1)=2^{k-1}+1,\quad \forall k\in\mathbb{N}.$

證明: 利用數學歸納法，$k=1$ 時查表即得。設 $k=i$ 時成立。

$k=i+1$ 時，利用推論 6，$c(2^{k+1}+1)=2\cdot c(2^k)-1=2\cdot(2^k+1)-1=2^{k+1}+1$，成立。

事實上，利用表格可以直接得到規律，如果把它的圖繪出來以直線段連接，可以得到一條折線，以斜率 0 和 1 交錯。

[結論 8] $c(x)=\begin{cases}(2^{k-1}+1), \text{if } x=2^k+m\ (k\ge 2, 1\le m\le 2^{k-1})\\ (2^{k-1}+m), \text{if } x=2^k+2^{k-1}+m\ (k\ge 2, 1\le m\le 2^{k-1})\end{cases}$，$\quad \forall x\ge 4.$

證明: 同樣可以利用數學歸納法及推論 6。

對 k 應用歸納法，奠基 $k=2$ 的部分直接查表驗證。

由於

當 $1\le m\le 2^{k-1}$ 時 $\Rightarrow 1=\left\lfloor\frac{1+1}{2}\right\rfloor\le\left\lfloor\frac{m+1}{2}\right\rfloor\le\left\lfloor\frac{2^{k-1}+1}{2}\right\rfloor=2^{k-2}$,

當 $1\le m\le 2^{k-1}$ 時，$1\le 1-\left\lfloor\frac{1+1}{2}\right\rfloor+1\le m-\left\lfloor\frac{m+1}{2}\right\rfloor+1\le 2^{k-1}-\left\lfloor\frac{2^{k-1}+1}{2}\right\rfloor+1=2^{k-2}+1.$

注意到原式 m 的範圍改寫成以下也是等價的:

$$c(x)=\begin{cases}(2^{k-1}+1), \text{if } x=2^k+m\ (k\geq 2, 1\leq m\leq 2^{k-1}+1).....(1)\\ (2^{k-1}+m), \text{if } x=2^k+2^{k-1}+m\ (k\geq 2, 1\leq m\leq 2^{k-1}+1)...(2)\end{cases}$$ 對於 $x\geq 4$。

因此，在以下兩種情況中:

(1)

$$c(2^k+m)=c(2^{k-1}+\left\lfloor\frac{m+1}{2}\right\rfloor)+c(2^{k-1}+m-\left\lfloor\frac{m+1}{2}\right\rfloor)-1=(2^{k-2}+1)+(2^{k-2}+1)-1=2^{k-1}+1$$

(2)

$$c(2^k+2^{k-1}+m)=c(2^{k-1}+2^{k-2}+\left\lfloor\frac{m+1}{2}\right\rfloor)+c(2^{k-1}+2^{k-2}+m-\left\lfloor\frac{m+1}{2}\right\rfloor)$$

$$=(2^{k-2}+\left\lfloor\frac{m+1}{2}\right\rfloor)+(2^{k-2}+m-\left\lfloor\frac{m+1}{2}\right\rfloor)=2^{k-1}+m$$

皆成立，故由數學歸納法得證。

進一步的我想探討若第一個人並非就近選擇第一個位置坐，而是經過挑選，那至多能坐下多少人?他又應該選擇哪裡呢?又這些值和椅子數有何關係?

我們可以借助推論五。以下為了簡稱，把這樣的位置稱作「**最佳位置**」。

[一般情況]

我們的目標是在 $f(n)=c(a)+c(n-a+1)-1$ 的情況下，對每個 n 值找出一個 a，而使 $f(n)$ 最大。

在前面證明結論八 $c(x)$ 的一般式前便注意到:這條折線的斜率可能是單純的 0 或 1，而由結論 8 知確實如此，寫成數學式可得以下。

> **[推論 9]** $c(n+1)-c(n)=0\vee 1.$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$g(n)$			1	2	3	3	3	4	5	5	6	6	7	7	7	8	9	9	10	10
n	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
$g(n)$	11	11	11	12	13	13	13	13	13	14	15	16	17	17	18	18	19	19	19	20

我們也關心 $g(x)$ 的斜率，那代表著增加的椅子數和能多做下的人數的關係。這讓我們想到了差分，令 $d_n = c(n) - c(n-1)$，於是 $d_i = 0 \vee 1$，而 $c(n) = \sum_{i=1}^{n} d_i + c(0)$。故在 $f(n) = c(a) + c(n-a+1) - 1$ 之下，我們希望的是我們加到的 $d_i = 1$ 愈多愈好。在結論 8 中，(1)的情況斜率是 0，(2)的情況斜率是 1。

從表中看不出明顯的規律。於是只好從前面提到讓 $d_i = 1$ 最多的想法進行推理。不過有了推論 9，我們可以知道對於 d_i 的值，0 是最大、1 是最小。

[性質 10-1]如果 $d_{i+1} = 1$ 且 a 是最佳位置，則 $a+1$ 亦為最佳位置。

[性質 10-2]如果 $d_{i+1} = 0$ 且 $a+1$ 為最佳位置，則 a 亦為最佳位置。

事實上以上兩個性質對偶的。於是，由性質 10-2，我們可以知道

[引理 11]對於任意的 n，座標最大的最佳位置必為 $2^i + 1$ 的形式，其中 $i \in \mathbb{N}$.

然而，注意到 $f(n) = c(a) + c(n-a+1) - 1$ 的對稱性。

[性質 12]若 t 是 $f(n)$ 的一個最佳位置，則 $n+1-t$ 也是 $f(n)$ 的一個最佳位置。

因此，總有個最佳位置他的坐標是不小於 $\frac{n+1}{2}$ 的，再結合引理 11。又滿足 $\frac{n}{2} < 2^k \le n$ 的 k 值總恰有一個。故除非 $n = 2^j + 1$，對於某個 j，否則存在一個 k 使 $\frac{n}{2} < 2^k + 1 \le n$. 這個 $2^k + 1$ 就是一個最佳位置，而且是座標最大者。又由引理 7 知 $n = 2^j + 1$ 時 n 本身就是最佳位置。

[結論 13] $2^{\lfloor \log_2(n-1) \rfloor} + 1$ 是對 $f(n)$ 的最佳位置，且為所有最佳位置中座標最大者，對於所有 $n \in \mathbb{N}$.

有了一個最佳位置，我們便能推出至多能坐下多少人了。

[結論 14]設 $x = 2^{\lfloor \log_2(n-1) \rfloor} + 1$，則 $g(n) = c(x) + c(n-x+1) = (\frac{x+1}{2}) + c(n-x+1)$. 其中 $c(x)$ 的表達式如結論 8 所示。

於是，$g(n) - g(n-1) = 0 \vee 1$，g 是一個非嚴格遞增的函數，接著探討其一般式。

當 $n \neq 2^k + 1, \forall k \in \mathbb{N}$ 時，可令 $2^{\lfloor \log_2(n-1) \rfloor - 1} = 2^{\lfloor \log_2(n-2) \rfloor - 1} = x$，則

$g(n) - g(n-1) = c(n-x+1) - c(n-x)$.

又當 $n=2^k+1, \forall k \in \mathbb{N}$ 時，可以利用性質 1 和引理 7 巧妙的解決，證明其為

$1: g(2^k+1) \geq c(2^k+1) = 2^{k-1}+1 = \left\lceil \frac{2^k}{2} \right\rceil + 1 \geq g(2^k)+1$，整理便可得到結論。

[結論 15] $g(n)-g(n-1) = \begin{cases} 0, \text{if } x = 2^k + m \ (k \geq 2, 1 < m \leq 2^{k-1}+1) \\ 1, \text{if } x = 2^k + 2^{k-1} + m \ (k \geq 2, 1 < m \leq 2^{k-1}+1) \end{cases}$.

[**第二種情形**]我們考慮一維度，$I_1 = [1, n]$ 的情形，即 $P(1, n)$。

這次我先從觀察規律開始，並部分依循著前面的方向求解。

[**角落情況**]

預先地，我們必須宣告類似推論 3 和引理 4 的性質在這裡也會成立。

[性質 16]推廣引理 4，在第二種情形下也會對。

而事實上在論證中我們僅需將第二種情形中同時填入且相鄰的兩人視為同一組，證明是與引理 4 類似的。

於是我們便可列出遞迴式:

[引理 17] $d(1) = d(2) = 1, d(3) = d(4) = 2$ ； $\begin{cases} d(2k) = 2\mathrm{d(k)} \\ d(2k+1) = 2d(k+1) - 1 \end{cases}$,for $k \geq 5$.

這裡要求 $k \geq 5$ 是由於為了實現遞迴時，我們必須先填下最左、最右、中間三個位置。

然而遞迴式並非普通的線性遞迴式，以下建表輔助。

n	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$d(n)$	3	4	3	4	5	6	7	8	5	6	7	8	9	10	11	12
n	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
$d(n)$	13	14	15	16	9	10	11	12	13	14	15	16	17	18	19	20

這個數列並非單調，然而，我們可以試著找出遞增的區間:會發現它在

$[2^k+2^{k-1}+1, 2^{k+1}+2^k]$ 間是遞增的，而且每次增加 1。

以下便以數學歸納法解出引理 14 的通式。

[引理 18]設 t 是滿足 $n > 2^t + 2^{t-1}$ 的最大 t 值，即 $2^{t+1}+2^t \geq n > 2^t+2^{t-1}$。則 $d(n) = n - 2^t$.

證明:我們對 t 進行歸納，$t=1$ 時直接觀察以遞迴式推出的表格。設 $t = p$ 時成立。

$t = p+1$ 時，由於 $2^{p+1} + 2^p \geq \left\lfloor \frac{n+1}{2} \right\rfloor > 2^p + 2^{p-1}$,

故由 $d(2k) = 2d(k) = 2(k - 2^p) = 2k - 2^{p+1}$;

$d(2k+1) = 2d(k+1) - 1 = 2(k+1-2^p) - 1 = 2k+1-2^{p+1}$ ，故得證。

[一般情況]

接著我們探究至多可以坐下多少人，符號記做 $h(n)$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$h(n)$	1	1	2	2	3	4	4	5	5	6	7	8	8	9	9	10	11
n	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
$h(n)$	10	11	12	13	14	15	16	16	17	17	18	19	18	19	20	21	22
n	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
$h(n)$	23	20	21	22	23	24	25	26	27	28	29	30	31	32	32	33	33

然而，這次並沒有明顯的規律。於是我先試著找出所有滿足 $h(n) > h(n+1)$ 的點，不妨稱作「極值點」。

在 $1 \leq n \leq 10000$ 下的極值點是:

17 29 35 53 59 71 101 107 119 143 197 203 215 239 287 389 395 407 431 479 575 773 779 791 815 863 959 1151 1541 1547 1559 1583 1631 1727 1919 2303 3077 3083 3095 3119 3167 3263 3455 3839 4607 6149 6155 6167 6191 6239 6335 6527 6911 7679 9215。

仍然非常複雜，然而，注意到 17 和 35 是兩倍差 1 的關係，又找到 $35 \times 2 + 1 = 71$，以及 $71 \times 2 + 1 = 143$ 在數列中。我們嘗試著做分類，整理出如下的表格。

17	35	71	143	287	575	1151	2303	4607	9215
29	59	119	239	479	959	1919	3839	7679	
53	107	215	431	863	1727	3455	6911		
101	203	407	815	1631	3263	6527			
197	395	791	1583	3167	6335				
389	779	1559	3119	6239					
773	1547	3095	6191						

1541	3083	6167							
3077	6155								
6149									

表中每一行的第 k 數 x_k 都滿足 $x_{k+1}=2x_k+1$，而這是由於遞迴式。

再引進一個名詞:一個數 n 被稱為「類似極值」，如果他滿足 $h(n)=h(n+1)$。

而在 $1\le n\le 10000$ 中，類似極值點有:

1 3 6 8 12 14 24 26 48 50 96 98 192 194 384 386 768 770 1536 1538 3072 3074 6144 6146

首先會注意到的是，他們可以兩兩分組，使的兩個數之間的差為 2。

(1, 3)（6, 8）(12, 14)（24, 26）(48, 50)（96, 98）(192, 194)（384, 386）(768, 770)（1536, 1538）(3072, 3074)（6144, 6146）

可以看出類似極值較小者正好是 2^k+2^{k+1} 的型式。

再者，可以猜想類似極值可能是極值的前身，只是變化較不明顯者，於是我試著回推第一列極值的前一項，如下。

極值第 1 列	17	29	53	101	197	389	773	1541	3077	6149
極值第 0 列	8	14	26	50	98	194	386	770	1538	3074

發現極值第 0 列正好是類似極值分組中的較大者。

綜合以上，我們可以猜測極值點為 $h^{(n)}(11000...010_{(2)})$ 的形式，其中 $h(x)=2x+1$，$h^{(n)}(x)$ 是 $h(x)$ 的 n 次迭代，即 $h^{(n)}(x)=h^{(n-1)}(h(x))$。事實上，$h(x)=2x+1$ 就相當於在 x 寫成二進位制後末位加上 1。所以，

$h^{(n)}(11000...010_{(2)})=11000...010111...1_{(2)}=(2^k+2^{k-1})+(2^t+2^{t-1})-1$；其中，$t=n+2$。

如同前面第一種情形所做的事，觀察 $h(n)-h(n-1)$ 還會注意到其值至多是 1.

[性質 19] $h(n)-h(n-1)\le 1$.

證明:這裡採強歸納法，奠基 $n=2$ 時查表，設 $n<k$ 時成立。

設 a 是當有 k 人時使坐下最多人時，第一個被坐下的位置，於是有

$h(k)=d(a)+d(k-a+1)-1$。當有 $k-1$ 人時，由 $h(x)$ 這個函數的最大性，有

$h(k-1)\geq d(a)+d(k-1-a+1)-1=d(a)+d(k-a)-1$，於是

$h(k)-h(k-1)\leq[d(a)+d(k-a+1)-1]-[d(a)+d(k-a)-1]=d(k-a+1)-d(k-a)\leq 1$

其中最後一個不等號是根據歸納假設($k-a<k$)，故得證。

接著我們仿前面第一種情形的步驟，找出第一個填入的人座標。由於在一般的情況下，第二種情形已不會滿足性質 1 的不等式(反例: $d(6)=4>\frac{6}{2}$)，因此我們獨立出一個符號$e(n)$表示在第二種情形下一般情況坐的下的人數。

同樣的，由於前面已宣告了類似引理 3, 4 的敘述對第二種情形也成立，故類似的，$e(n)$也會滿足遞迴式。

[引理 20]設第一個人坐在座標a的位置，那麼有

$$e(n)=\begin{cases}d(a)+d(n-a),\text{if } n\text{是偶數}\\ d(a)+d(n-a+1)-1,\text{if } n\text{是奇數}\end{cases}.$$

[定義 21]引進旗標變數 $f_n=n-2\cdot\left\lfloor\frac{n}{2}\right\rfloor$.

於是原式可改寫成$e(n)=d(a)+d(n-a+1)-f_n$，其中 f_n 是和 a 無關的。

[定義 22]記 $y(x)=2^x+2^{x-1}$.

於是，$d(n)=n-2^{\lfloor y^{-1}(n-1)\rfloor}$，

$$d(a)+d(n-a+f_n)=a-2^{\lfloor y^{-1}(a-1)\rfloor}+(n-a+f_n)-2^{\lfloor y^{-1}(n-a+f_n)\rfloor}=n+f_n-(2^{\lfloor y^{-1}(a-1)\rfloor}+2^{\lfloor y^{-1}(n-a+f_n)\rfloor})$$

而，所以為了使$d(a)+d(n-a+1)$最大，我們只要使$2^{\lfloor y^{-1}(a-1)\rfloor}+2^{\lfloor y^{-1}(n-a+f_n-1)\rfloor}$最小。

[定義 23]令 $s_i=d(i)-d(i-1)$；$t(a)=2^{\lfloor y^{-1}(a)\rfloor}$；$z(n,a)=t(a-1)+t(n-a+f_n-1)$.

我們的目標即是對每個n值求取一個a值使$z(n,a)$最小。類似的，我想執行調整法，則必須先宣告執行時所需要用到的引理，顯然t是一個不嚴格遞增的函數。

[引理 24]若$t(a-1)=t(a)$且a是最佳位置，則$a+1$也是最佳位置。

證明:

$$z(n,a)=t(a-1)+t(n-a)=t(a)+t(n-a)\geq t(a)+t(n-a-1)=z(n,a+1)\Rightarrow z(n,a+1)\leq z(n,a)$$

[推論 25]從而，2^k+2^{k-1} 會是座標最大的最佳位置，對於某個 k.

這是因為 $t(2^k+2^{k-1}-2)=t(2^k+2^{k-1}-1)<t(2^k+2^{k-1})$，故為 $(2^k+2^{k-1}-1)+1=2^k+2^{k-1}$

[性質 26]若 a 為最佳位置，則 $n+1-a$ 也是最佳位置。

這直接由對稱性 $z(n,a)=z(n,n+1-a)$ 即得。

因此，存在一個最佳位置 a_0 滿足 $a_0>\left\lfloor\frac{n+1}{2}\right\rfloor\geq\frac{n}{2}$，又恰有唯一一個 k 值滿足

$n\geq 2^k+2^{k-1}>\frac{n}{2}$，對於 $n\geq 3$，因此我們得到了一個最佳位置。

[結論 27] $h(n)=d(2^k+2^{k-1})+d(n-2^k+2^{k-1}+1)-1=2^k+d(n-2^k+2^{k-1}+1)-1$.

其中 $d(n)$ 的表達式如引理 18，且令 $d(0)=0$。

接著，我們要表達出然數隨著椅子的增加的變化，即探討 $\Delta h(n)=h(n+1)-h(n)$，於是我們便可以來驗證前面的猜想。

[引理 28] $\Delta h(n)=\begin{cases}2^{t-2}-2^t\ ,\text{if } n=(2^k+2^{k-1})+(2^t+2^{t-1})-1\ ,2\leq t<k\\ -1\ ,\text{if } n=(2^k+2^{k-1})+5\ ,2\leq t<k\\ 0\ ,\text{if } n=1\vee 3\vee(2^k+2^{k-1})\vee(2^k+2^{k-1})+2\ ,2\leq k\\ 1\ ,\text{if else.}\end{cases}$.

證明: 對於 $n\leq 5$ 可以直接驗證，以下考慮 $n\geq 6$ 的情況。

先考慮當 $n=2^k+2^{k-1}$ 時，

$h(n)=2^k+d(1)-1=2^k, h(n+1)=2^k+d(2)-1=2^k\Rightarrow h(n+1)-h(n)=0$.

對於其他的 n 則有 $\left\lfloor y^{-1}(n)\right\rfloor=\left\lfloor y^{-1}(n+1)\right\rfloor$，

因此 $h(n+1)-h(n)=d(n+1-2^k+2^{k-1}+1)-d(n-2^k+2^{k-1}+1)$，

令 $m=n-(2^k+2^{k-1})+1$，則 $h(n+1)-h(n)=d(m+1)-d(m)$.

而由引理 18 可以得到: $d(m+1)-d(m)=\begin{cases}1+2^{t-2}-2^t\ ,\text{if } m=2^t+2^{t-1}, t\geq 3\\ -1\ ,\text{if } m=6\\ 0\ ,\text{if } m=1\vee 3\\ 1\ ,\text{if } else\end{cases}$

將 $\Delta h(n)=d(m+1)-d(m)$ 和 $m=n-(2^k+2^{k-1})+1$ 代入整理即得。

接著我們考慮二維度平面的情形。

[**第二種情形**]考慮二維度 $I_2=[1,n]\times[1,m]$，即 $P(2,n,m)$.

[**角落情況**]

我們先試填:

3		4		2
	3		4	
4		3		4
	4		3	
1		4		3

3		4	4		2
	3		4	4	
4		3		4	4
4	4		3		4
	4	4		3	
1		4	4		3

會發現每次都是填滿一整斜排，於是我開始著手驗證這件事是否會恆成立，答案是肯定的。然而我們無法直接解決之，必須借助一些引理。事實上經由觀察會發現情況並沒有想像中的複雜，因為外圍的點對最小點距並不影響，這和前面的性質 2 是類似的。

[性質 29]推廣性質 2，若已填入的座位與邊界圍成閉區域，則此閉區域內任意點的最小點距僅與區域邊界上的點有關。

注意到 $x+y=c$ 和 $x+y=d$ 的圖形能與邊界構成閉區域。

[引理 30]對於所有滿足 $x+y=c$ 的點 (x,y) 必同時坐下，c 為常數。

證明:我們對讓一群人坐下的次數 n 採數學歸納法。初始狀況 $n=1$ 時坐下的是 $x+y=2$ 上的點 $(1,1)$。

假設在 $n=i$ 時，我們已填入了 k 條直線 $x+y=c_j$ $(1\le j\le k)$ 且 $2=c_1<c_2..<c_k=2n$。

當 $n=i+1$ 時，由於最小點距的計算只與前 i 次坐下的人有關，因此我們僅須證明對於任意一個在這次被坐下的位置 (a,b)，設 $a+b=s$，所有滿足 $x+y=s$ 的 (x,y) 都會被坐到。假設 $c_p<s<c_{p+1}$，以下證明 $(l,s-l)$ 也會被填入。由三角不等式知，

$|x-l|+|y-(s-l)|\ge|x+y-s|$，其中，我們僅需考慮兩種情況 $x+y=c_p\vee c_{p+1}$

。當取 $x=l$ 時，$y=c_p-l\vee c_{p+1}-l$，無論哪種都會使 $(x-l)[y-(s-l)]=0\geq 0$，從而三角不等式的等號成立，說明這個最小值是會被達到的。因此，$(l,s-l)$ 到其他點的最小點距為 $\min(|c_p-s|,|c_{p+1}-s|)$，這個值是對 l 無關的。意即，$l=a$ 的最小點距與任意 l 的最小點距相同，因此若 (a,b) 在此次被坐下 $\Leftrightarrow$ $(l,s-l)$ 被坐下，對於所有 l 在範圍中。從而由歸納法得證。

於是我們可以把填入格子化歸為填入線束：$x+y=i$ $(2\leq i\leq 2n)$，而且觀察發現(在推論 32 中說明)：每次皆為在 $x+y=p$ 及 $x+y=q$ 之間填入 $x+y=\left\lfloor\frac{p+q}{2}\right\rfloor$ 和 $x+y=\left\lceil\frac{p+q}{2}\right\rceil$. 這和一維的情況是類似的，事實上，我們可以證明這個問題是一維的加權。

[定義 31]定義線束 $x+y=c$ 的最小點距為線上任意一點的最小點距。而在引理 27 的證明中以證明它在這種填法下，線上每點最小點距都相同。

而在引理 30 後半部利用三角不等式推導最小點距時即得到 $x+y=a$ 上任一點對 $x+y=b$ 的最短距離為 $|b-a|$.

[推論 32]每次填入相當於在 $x+y=p$ 及 $x+y=q$ 之間填入 $x+y=\left\lfloor\frac{p+q}{2}\right\rfloor$ 和 $x+y=\left\lceil\frac{p+q}{2}\right\rceil$.

接著要讓它能適用當初一維的推導過程，進而導出遞迴式。

[性質 33]性質 16 在二維情況下也會對，將點視為線束即可。

於是，我們確實可以將二維的角落情況視為一維的加權，在 $n\times m$ $(n\geq m)$ 的方格中，相當於有 $m+n-1$ 條線束，且權數如下。

1	2	3	…	…	$m-1$	m	m	…	…	m	m	$m-1$	…	…	3	2	1

值為 l 代表著填入那一格時，實際上坐下了 l 人。

第一個坐下的為最左，其次為最右，接著是中間，於是把長條分為左右兩半，其中最左和最右已被填入。

1	2	3	…	…	$m-1$	m	m	…	…	m

我們等價於要探討上表填入的結果，我們要計算所有被填入的格子之數字加總。

[定義 34]假設上表末尾有 k 個 m，總共有 n 個數，我們記其結果為 $a(n,k)$，且在 $n\times m$ 的方格中坐下的人數記為 $d(n,m)$.

[性質 35] k,m,n 三者的關係為 $k+m-1=n$.

[引理 36]當 $n\geq m$ 時，$d(n,m)=\begin{cases}2a(\frac{n+m-1}{2},n-m+1)\ ,\text{if } 2\nmid n+m\\ 2a(\frac{n+m}{2},n-m+1)-m\ ,\text{if } 2\mid n+m\end{cases}$.

證明:

$d(n,m)=\begin{cases}2a(\frac{n+m-1}{2},k)\ ,\text{if } 2\mid n+m-1\\ 2a(\frac{(n+m-1)+1}{2},k)-m\ ,\text{if } 2\nmid n+m\text{-}1\end{cases}$，$k=n-m+1$ 帶入整理即得證。

然而在實現遞迴時，會遇到兩種情況：

(1)分成以下兩條，其中一條皆為 m 。

1	2	3	…	…	$m-1$	m	m	…	…	m

m	m	…	…	m

(2)分成以下兩條，其中一條為嚴格遞增的數列。

1	2	3	…	…	k

$k+1$	$k+2$	$k+3$	…	…	$m-1$	m	m	…	…	m

對於(1)出現皆為 m 的情況，我們可以直接援引 $d(n)$ 的值再乘以 m 倍。

[引理 37-1]其遞迴式為:

$$\begin{cases} \text{if } k>\dfrac{n}{2}\begin{cases} \text{if } 2\mid n,\ a(n,k)=a(\dfrac{n}{2},k-\dfrac{n}{2})+m\times d(\dfrac{n}{2}) \\ \text{if } 2\nmid n,\ a(n,k)=a(\dfrac{n+1}{2},k-\dfrac{n-1}{2})+m\times d(\dfrac{n+1}{2})-m \end{cases} \\ \text{if } k\le\dfrac{n}{2}\begin{cases} \text{if } 2\mid n,\ a(n,k)=[a(\dfrac{n}{2},k)+\dfrac{n}{2}\times d(\dfrac{n}{2})]+a(\dfrac{n}{2},1) \\ \text{if } 2\nmid n,\ a(n,k)=[a(\dfrac{n+1}{2},k)+\dfrac{n-1}{2}\times d(\dfrac{n+1}{2})]+a(\dfrac{n+1}{2},1)-\dfrac{n+1}{2} \end{cases} \end{cases}.$$

然後再利用高斯符號稍作化簡合併，改成:

[引理 37-2]這是引理 36-1 的另一種形式

$$a(n,k)=\begin{cases} a(\left\lfloor\dfrac{n+1}{2}\right\rfloor,\left\lfloor k-\dfrac{n}{2}\right\rfloor+m\times d(\left\lfloor\dfrac{n+1}{2}\right\rfloor)-m\cdot f_n, & \text{if } k>\dfrac{n}{2} \\ [a(\left\lfloor\dfrac{n+1}{2}\right\rfloor,k)+\left\lfloor\dfrac{n}{2}\right\rfloor\times d(\left\lfloor\dfrac{n+1}{2}\right\rfloor)]+a(\left\lfloor\dfrac{n+1}{2}\right\rfloor,1)-\left\lfloor\dfrac{n+1}{2}\right\rfloor f_n, & \text{if } k\le\dfrac{n}{2} \end{cases}.$$

其中在這裡 f_n 是因為 n 為奇數時會被重複算的修正項，定義如定義 21.

我們可能會想先計算出 $a(n,1)$ 的值，因為如此我們便能保證在更少的操作次數下求出 $a(n,k)$。(事實上如此我們能在不大於 $O(\lceil\log_2 n\rceil^2)$ 次操作下求得 $a(n,k)$。若在不給定 n 的範圍下，要計算 $\lfloor y^{-1}(n)\rfloor$ 的時間為 $O(\lceil\log_2 n\rceil)$ 次操作。)這裡符號 $O(\cdot)$ 為 big O.

接著我們討論二維度 $I_2=[1,n]\times[1,n]$，即 $P(2,n,n)$，這等價要探討 $a(n,1)$ 的情況。

[定義 38]簡記 $a(n,1)=b(n)$.

於是，可以利用引理 36-2 表示 $b(n)$，

$$b(n)=[b(\left\lfloor\frac{n+1}{2}\right\rfloor)+\left\lfloor\frac{n}{2}\right\rfloor\times d(\left\lfloor\frac{n+1}{2}\right\rfloor)]+b(\left\lfloor\frac{n+1}{2}\right\rfloor)-q_n=2b(\left\lfloor\frac{n+1}{2}\right\rfloor)+\left\lfloor\frac{n}{2}\right\rfloor\times d(\left\lfloor\frac{n+1}{2}\right\rfloor)-q_n.$$

[推論 39] $b(n)=2b(\left\lfloor\dfrac{n+1}{2}\right\rfloor)+\left\lfloor\dfrac{n}{2}\right\rfloor\times d(\left\lfloor\dfrac{n+1}{2}\right\rfloor)-q_n$.

先建表觀察規律:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$b(n)$	1	1	4	5	9	14	12	18	25	33	42	52	35	45	56
n	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$b(n)$	68	81	95	110	126	143	161	180	200	117	135	154	174	195	217

發現它只有在幾個特殊的點滿足 $b(n)>b(n+1)$.

我們收集 $1\le n\le 10000$ 中，滿足 $b(n)>b(n+1)$ 的 n 值及 $b(n+1)$ 值。

n	6	12	24	48	96	192	384	768	1536	3072	6144
$b(n+1)$	12	35	117	425	1617	6305	24897	98945	394497	1575425	6296577

明顯地，n 的規則是能表示為 $2^{k+1}+2^k$ 的形式。於是我們要探討的是 $b(2^{k+1}+2^k+1)$.

我們利用先猜後證的手法，先試圖找規律。由於數字有點大，因此我先將其因數分解成兩個相近的數字相乘：

$12=3\times 4;35=5\times 7;117=9\times 13;425=17\times 25;1617=33\times 49;6305=63\times 97$ 。

會發先假設第 i 項為 $c_i\times d_i$ ，則有 $c_{k+1}=2c_k-1;d_{k+1}=2d_k-1$ ，於是

$c_{k+1}d_{k+1}=(2c_k-1)(2d_k-1)=4c_kd_k-2(c_k+d_k)+1$，其中 c_k,d_k 是有遞迴關係的數列。因此我們現在構造一個數列意圖複製這個規則，假設有一個數列 $\langle x_i\rangle$ ，滿足 $x_1=12$ ，且令 $x_{k+1}=4x_k-y_k$ ，其中數列 $\langle y_i\rangle$ 是待定的。

n	6	12	24	48	96	192	384	768	1536	3072	6144
x_n	12	35	117	425	1617	6305	24897	98945	394497	1575425	6296577
y_n	13	23	43	83	163	323	643	1283	2563	5123	10243

容易觀察到數列 $\langle y_i\rangle$ 的規則是滿足 $y_{k+1}=2y_k-3$.

於是我們要來解這兩個遞迴數列。

[引理 40]遞迴數列 $\langle x_i\rangle$ 和 $\langle y_i\rangle$ 的解分別是：$x_k=6\cdot 4^{k-1}+5\cdot 2^{k-1}+1$， $y_k=5\cdot 2^k+3$.

證明：

$y_{k+1}=2y_k-3\Rightarrow y_{k+1}-3=2(y_k-3)\Rightarrow y_{k+1}-3=2^k(y_1-3)\Rightarrow y_k-3=2^{k-1}\cdot 10\Rightarrow y_k=5\cdot 2^k+3$

$\therefore x_{k+1}=4x_k-(5\cdot 2^k+3)$，這是一階非齊次線性遞迴方程。設它的特解 $x_k^p=a\cdot 2^k+b$，

$a\cdot 2^{k+1}+b=4(a\cdot 2^k+b)-(5\cdot 2^k+3)\Rightarrow(4a-2a-5)\cdot 2^k+(4b-b-3)=0$，

得 $a=\frac{5}{2},b=1$，故 $x_k^p=\frac{5}{2}\cdot 2^k+1=5\cdot 2^{k-1}+1$。又假設它的齊解為 $x_k^h=c\cdot 4^k$，將其代回

$x_k=x_k^h+x_k^p\Rightarrow x_k=c\cdot 4^k+5\cdot 2^{k-1}+1$. 取 $k=1$， $x_1=4c+5+1\Rightarrow 12=4c+6\Rightarrow c=\frac{3}{2}$.

$\therefore x_k=\frac{3}{2}\cdot 4^k+5\cdot 2^{k-1}+1=6\cdot 4^{k-1}+5\cdot 2^{k-1}+1$，得證。

[引理 41] $x_k=b(2^{k+1}+2^k+1)$.

證明: 記 $z_k=b(2^{k+1}+2^k+1)$.

採歸納法， $z_1=x_1$ 確實成立，假設 $z_j=x_j$，我們須證明 $z_{j+1}=x_{j+1}$。

$$z_{j+1}=b(2^{k+2}+2^{k+1}+1)=2b(2^{k+1}+2^k+1)+(2^{k+1}+2^k)\times d(2^{k+1}+2^k+1)-q_{2^{k+2}+2^{k+1}+1}$$
$$=2\cdot(6\cdot 4^{k-1}+5\cdot 2^{k-1}+1)+(2^{k+1}+2^k)\times(2^k+1)-(2^{k+1}+2^k+1)$$
$$=3\cdot 4^k+5\cdot 2^k+2+(2^{k+1}+2^k)\times 2^k-1$$
$$=3\cdot 4^k+5\cdot 2^k+2+3\cdot 4^k-1$$
$$=6\cdot 4^k+5\cdot 2^k+1=x_{j+1}$$

故得證。

接著要找全體數列的通式，然而規律並不明顯。有了之前的經驗，我們退而求其次的觀察它的差分 $\Delta b(n)=b(n+1)-b(n)$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\Delta b(n)$	0	3	1	4	5	−2	6	7	8	9	10	−17	10	11	12
n	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$\Delta b(n)$	13	14	15	16	17	18	19	20	−83	18	19	20	21	22	23

我們發現它的差分在 $n\in(2^k+2^{k-1},2^{k+1}+2^k)$ 中是以斜率 1 成長的，於是以下列舉出 $n\le 10000, n=2^{k+1}+2^k+1$ 的 $\Delta b(n)$。

n	7	13	25	49	97	193	385	769	1537	3073	6145
$\Delta b(n)$	6	10	18	34	66	130	258	514	1026	2050	4098

規則看似是 $\Delta b(2^{k+1}+2^k+1)=2^{k+1}+2$ ，於是我們可以列出通式，以下給予證明:

[引理 42] $\Delta b(2^{k+1}+2^k+m)=2^{k+1}+m+1,\ \forall k\in\mathbb{N},1\le m<2^{k+1}+2^k$.

證明: 對 k 應用歸納法， $k=1$ 時， $\Delta b(7)=6$ ，成立。假設 $k=j$ 時成立。

$k=j+1$ ，由 $b(n)=2b(\left\lfloor\frac{n+1}{2}\right\rfloor)+\left\lfloor\frac{n}{2}\right\rfloor\times d(\left\lfloor\frac{n+1}{2}\right\rfloor)-q_n$ ，

當 $n=2^{k+1}+2^k+m,1\le m\le 2^{k+1}+2^k$ 時，有 $2^k+2^{k-1}+1\le\left\lfloor\frac{n+1}{2}\right\rfloor\le 2^{k+1}+2^k$ 。

因此對於 $n=2^{j+2}+2^{j+1}+m,1\le m<2^{j+2}+2^{j+1}$ ，有 $d(\left\lfloor\frac{n+1}{2}\right\rfloor+1)-d(\left\lfloor\frac{n+1}{2}\right\rfloor)=1$ 。

由推論 39，可以列出 $\Delta b(n)$ 的遞迴式， $\Delta b(n)=\begin{cases}d(\left\lfloor\frac{n+1}{2}\right\rfloor)+q_n\ ,\text{if } 2\nmid n\\ 2\Delta b(\left\lfloor\frac{n+1}{2}\right\rfloor)+\left\lfloor\frac{n}{2}\right\rfloor-q_{n+1}\ ,\text{if } 2\mid n\end{cases}$

(i) 當 $2\nmid n$ 時，

$\Delta b(n)=d(\left\lfloor\frac{n+1}{2}\right\rfloor)+q_n=d(\frac{n+1}{2})+(\frac{n+1}{2})=(\frac{n+1}{2}-2^{j+1})+(\frac{n+1}{2})=n+1-2^{j+1}=2^{j+2}+m+1$

(ii) 當 $2\mid n$ 時，

$\Delta b(n)=2\Delta b(\left\lfloor\frac{n+1}{2}\right\rfloor)+\left\lfloor\frac{n}{2}\right\rfloor-q_{n+1}=2\Delta b(\frac{n}{2})+\frac{n}{2}-\left\lfloor\frac{n+2}{2}\right\rfloor$

$=2(\frac{n}{2}-2^j+1)-1=n-2^{j+1}+1=2^{j+2}+m+1$

因而由歸納法得證。

於是，我們便能導出 $b(n)$ 的一般式。

[結論 43] 設 $n=2^{k+1}+2^k+m,1\le m\le 2^{k+1}+2^k$ ，

則 $b(n)=6\cdot 4^{k-1}+(4m+1)\cdot 2^{k-1}+\frac{m(m+1)}{2}$.

證明: 當 $m>1$ 時，

$$b(n)=b(2^{k+1}+2^k+1)+\sum_{i=1}^{m-1}\Delta b(2^{k+1}+2^k+i)=(6\cdot 4^{k-1}+5\cdot 2^{k-1}+1)+\sum_{i=1}^{m-1}(2^{k+1}+i+1)$$

$$=(6\cdot 4^{k-1}+5\cdot 2^{k-1}+1)+(m-1)2^{k+1}+\frac{m(m-1)}{2}+(m-1)$$

$$=6\cdot 4^{k-1}+(4m+1)\cdot 2^{k-1}+\frac{m(m+1)}{2}$$

故得證。

綜合引理 36 即結論 43，我們可以得到 $d(n,n)$ 的表達式。

[結論 44] 設 $n=2^{k+1}+2^k+m, 1\le m\le 2^{k+1}+2^k$，則 $d(n,n)=6\cdot 4^{k-1}+(m-1)\cdot 2^{k+1}+\frac{m(m-1)}{2}$.

證明:

$$d(n,n)=2d(n,1)-n=2b(n)-n=6\cdot 4^{k-1}+(4m+1)\cdot 2^{k-1}+\frac{m(m+1)}{2}-(5\cdot 2^{k-1}+m)$$

$$=6\cdot 4^{k-1}+(m-1)\cdot 2^{k+1}+\frac{m(m-1)}{2}.$$

[關於更高維度的情況]

先以同樣的方法給出基本定理:

[結論 45] $\left\lceil\frac{N}{2}\right\rceil\ge f_n(N)\ge\left\lceil\frac{N}{2n+1}\right\rceil$，在 n 維度共計 N 個點的情況下。

證明:和性質一是類似的，在第二種情形下僅不等號右邊成立。

若我們能找出表達式，則我們必能找出遞迴式。而前面提供了有效求遞迴式的方法。

首先，如同「線束」一樣，要先把一個 n 維空間寫成多個形如 $x_1+x_2+...+x_n=k$ 的 $n-1$ 維空間組，二維如下。

1	2	3	…	…	$m-1$	m	m	…	…	m	m	$m-1$	…	…	3	2	1

在一個 n 維長度為 M 計算其權數相當於求 $x_1+x_2+...+x_n=k$，$1\le x_i\le M$ 的正整數解數。

[結論 46] 對於一個正 n 維的情況，當 $k-n=i\in(nm,n(m+1)]$，其解數為

$$H_0^nH_i^n-H_1^nH_{i-n}^n+H_2^nH_{i-2n}^n....+H_m^nH_{i-mn}^n.$$

證明:由排容原理公式得到。

到這裡為止，如果直接模擬，在n維下每邊k格之時間複雜度已是多項式時間的$O(n^2k^2)$，如果把每一項H_j^n有出現的提出來計算，由於這個值的計算是一個向量空間，把它拆成m個基底是由H_i^n構成的矩陣，乘上一個係數積H_m^n，意即分為多個盤面計算後加總，雖然仍然是$O(n^2k^2)$，但是因為少算了很多重複的次數，所以也有優化到時間。

根據上面的想法，如果我們能把同一個盤面的形狀做的簡單，那一切就會比較好算，現在是H_{i-jn}^n或 0，然而由一個事實$C_b^a=0,\text{if } a<b$，我們可以假設權是每個H_j^n項都有的，即$H_0^nH_i^n-H_1^nH_{i-n}^n+H_2^nH_{i-2n}^n....+H_k^nH_{i-kn}^n$，至此每個盤面的每個值都是相同的$H_{i-jn}^n$.

在引用之前的想法，我們現在開始要做降維，然而二維時的作法是引用一維的值，一維的值卻是具體的表達式，如果我們要用這種方法的話，則需進行遞迴。

對於單一一項$H_j^nH_{i-jn}^n$，當j固定時，有$H_j^nH_{i-jn}^n=H_j^nC_{i-jn}^{i-jn+n-1}=H_j^nC_{n-1}^{i-jn+n-1}$，其中唯一的變數為$i$，展開後可以得到以為變數的單元多項式。於是我們在計算時分類，對每個H_j^n求和，則每個位置權數相當於H_{i-jn}^n。遞迴的是多項式的次數，可以先預處理好i^k的情況，之後計算時只需要取值乘上係數即可，這種算法是$O(nk)$的。

而至於預處理的部分，我們要記錄的是權數為i^a，長度為b的情況，其中$1\le a\le n,1\le b\le nk$，記做$P(a,b)$. 同樣是利用對稱性遞迴，x和$b-x$會同時被取到，而$x^a+(b-x)^a=cx^a+Q(x),\deg Q<a,c=0\vee 2$。所以$Q(x)$可借助前面$P(i,b)$的結果，$1\le i<a$；$cx^a$可借助前面$P(a,\left\lfloor\frac{b}{2}\right\rfloor)$的結果。如此算法每個要算$O(n)$次，要算$O(n^2k)$次，一共是$O(n^3k)$.

我們利用程式(附件)計算得出下表，縱列為n，橫列為k，內部為所求能填下的空格數。

n \ k	1	2	3	4	5	6	7
1	1	1	2	2	3	4	3
2	1	2	5	6	13	22	17
3	1	2	9	38	41	106	189
4	1	8	41	86	313	778	801

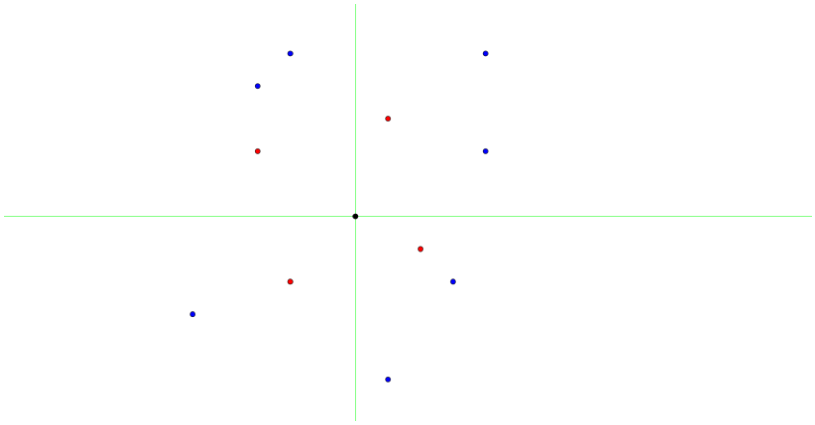
5	1	22	143	512	1875	3122	7663
6	1	22	243	2208	5209	21106	65357
7	1	72	871	10672	33465	155598	501473

回頭解決第一種情形，記 N 為總人數，$N=k^n$. 如果使用最樸素的方法模擬，每次要兩兩計算距離是 $O(N^2)$，然後 $O(N)$ 找出最小值，這樣一共要重複 $O(N)$ 次，是 $O(N^3)$.

如果每次計算距離時把結果記錄下來，每次更新時只要對新增的點更新一次距離，這樣每次只需 $O(N)$，共重複 $O(N)$ 次，是 $O(N^2)$.

然後還可以進行優化，像一開始一維時那樣，我們只考慮在一個兩端被包圍的區間運作，而在 n 維時，我們在更新時也只需要考慮包圍某個點的那些點。如一維時，我們只考慮其值比較大的和比較小的兩種情況中，分別取距離最近的記錄下來。而在二維時有兩個分量，可以分成(大，大)、(大，小)、(小，大)、(小，小)，共四個情況分別記錄結果。同理在 n 維時就有 2^n 種組合方式，每次更新只需要對這 2^n 個點更新即可，而每次取最小值可以用特殊的資料結構如 set 保存，其複雜度只要 $O(\log N)$ 可忽略，同樣要執行 $O(N)$ 次，故為 $O(2^n N)$.

如下圖，二維空間分為四個區塊，只記錄每區距離最近的點(紅點)。



陸、研究結論

(一)在一維$\mathbb{R}^1$數線上，$I_1=[1,n]$中的n個格子點為座位，則這群人選擇座位的方式:

在第一種情形中:

（1）當第一個人坐在角落(座標$A(1)$)時:能坐下的人數為

$$c(x)=\begin{cases}(2^{k-1}+1),\text{if } x=2^k+m\ (k\geq 2,1\leq m\leq 2^{k-1})\\(2^{k-1}+m),\text{if } x=2^k+2^{k-1}+m\ (k\geq 2,1\leq m\leq 2^{k-1})\end{cases}\quad \text{對於 } x\geq 4.$$

（2）當第一個人任意決定位置時，選擇的座標為$2^{\lfloor \log_2(n-1)\rfloor}+1$時能坐下最多人。

（3）承上，此時能坐下的人數為$g(n)=(\frac{x+1}{2})+c(n-x+1)$，其中$x=2^{\lfloor \log_2(n-1)\rfloor}+1$.

（4）承上，每增加一張座位時，能坐下的人數最大值之變化函數如下:

$$g(n)-g(n-1)=\begin{cases}0,\text{if } x=2^k+m\ (k\geq 2,1<m\leq 2^{k-1}+1)\\1,\text{if } x=2^k+2^{k-1}+m\ (k\geq 2,1<m\leq 2^{k-1}+1)\end{cases}.$$

在一維$\mathbb{R}^1$數線上，$I_1=[1,n]$中的n個格子點為座位，則這群人選擇座位的方式:

在第二種情形中:

（1）當第一個人坐在角落(座標$A(1)$)時，能坐下的人數如下:

$d(n)=n-2^t$，其中t滿足$2^{t+1}+2^t\geq n>2^t+2^{t-1}$.

（2）當第一個人任意決定位置時，選擇的座標為2^k+2^{k-1}時能坐下最多人，其中

$n\geq 2^k+2^{k-1}>\frac{n}{2}$.

（3）承上，此時能坐下的人數為$h(n)=2^k+d(n-2^k+2^{k-1}+1)-1$，其中$n\geq 2^k+2^{k-1}>\frac{n}{2}$.

（4）承上，每增加一張座位時，能坐下的人數最大值之變化函數如下:

$$\Delta h(n)=\begin{cases}2^{t-2}-2^t\ ,\text{if } n=(2^k+2^{k-1})+(2^t+2^{t-1})-1\ ,2\leq t<k\\-1\ ,\text{if } n=(2^k+2^{k-1})+5\ ,2\leq t<k\\0\ ,\text{if } n=1\vee 3\vee(2^k+2^{k-1})\vee(2^k+2^{k-1})+2\ ,2\leq k\\1\ ,\text{if else.}\end{cases}.$$

(二)在二維 $\mathbb{R}^2$ 平面上，I_2 中的格子點為座位，則這群人選擇座位的方式，只考慮第二種情形:

(1)當 $I_2=[1,n]\times[1,n]$ 時，當第一個人做角落時(座標 $A(1,1)$)，能坐下的人數為:

$$d(n,n)=6\cdot 4^{k-1}+(m-1)\cdot 2^{k+1}+\frac{m(m-1)}{2}.$$

(2)當 $I_2=[1,n]\times[1,m]$ 時($n\geq m$)，能坐下的人數滿足

$$d(n,m)=\begin{cases}2a(\frac{n+m-1}{2},n-m+1)\text{ ,if } 2\nmid n+m\\ 2a(\frac{n+m}{2},n-m+1)-m\text{ ,if } 2\mid n+m\end{cases}.$$

其中，

$$a(n,k)=\begin{cases}a(\left\lfloor\frac{n+1}{2}\right\rfloor,\left\lfloor k-\frac{n}{2}\right\rfloor+m\times d(\left\lfloor\frac{n+1}{2}\right\rfloor)-m\cdot f_n\text{ ,if } k>\frac{n}{2}\\ [a(\left\lfloor\frac{n+1}{2}\right\rfloor,k)+\left\lfloor\frac{n}{2}\right\rfloor\times d(\left\lfloor\frac{n+1}{2}\right\rfloor)]+a(\left\lfloor\frac{n+1}{2}\right\rfloor,1)-\left\lfloor\frac{n+1}{2}\right\rfloor\cdot f_n\text{ ,if } k\leq\frac{n}{2}\end{cases},$$

$$f_n=n-2\cdot\left\lfloor\frac{n}{2}\right\rfloor.$$

且當 $n=2^{k+1}+2^k+m, 1\leq m\leq 2^{k+1}+2^k$ 時，$a(n,1)=6\cdot 4^{k-1}+(4m+1)\cdot 2^{k-1}+\frac{m(m+1)}{2}$.

(三)在 n 維 $\mathbb{R}^n$ 空間中，I_n 中的格子點為座位，則這群人選擇座位的方式，考慮第二種情形:

(1) 基本定理： $f_n(N)\geq\left\lceil\frac{N}{2n+1}\right\rceil$.

(2) 計算時間複雜度：假設每次基本運算(加減乘除)時間為 $O(1)$，n 維下每邊 k 格。

- 直接模擬預期複雜度：每次計算 $O(k^n)$，預期計算 $O(k^n)$ 次➔ $O(k^{2n})$.
- 直接模擬證明後的複雜度：實際上可證明只需計算 $O(kn)$ 次➔ $O(k^{n+1}n)$.
- 利用類似權值和線束的想法：計算權值 $O(n^2k^2)$，模擬 $O(nk)$ ➔ $O(n^2k^2)$.
- 利用算兩次方法合併同類項：要算 $O(k)$ 組，每組 $O(nk)$，每個要計算 $O(n)$ 次
  ➔ $O(n^2k^2)$（相對於前者僅常數優化).
- 利用前述的算兩次和以多項式分類及遞迴預處理的複雜度：
  預處理 $O(n^3k)$ 後，每次計算時展開多項式並計算是 $O(n)$ 的，要計算 $O(k)$ 組相加
  ➔每次 $O(nk)$.

(四)在 n 維 $\mathbb{R}^n$ 空間中，I_n 中的格子點為座位，則這群人選擇座位的方式，考慮第一種情形:

記 N 為總人數，$N=k^n$，

- 直接模擬預期複雜度：每次計算 $O(N^2)$，預期計算 $O(N)$ 次➔ $O(N^3)$.
- 記錄當前結果的複雜度：每次更新和找最小值皆 $O(N)$，共 $O(N)$ 次➔ $O(N^2)$.
- 分區，只記錄各區最接近的更新：每次 $O(2^n)$，共 $O(N)$ 次➔ $O(2^n N)$.

柒、未來展望

在較高維度 $n\geq 2$ 的情況下，數據資料會變得很雜亂無序，甚至可能從中無法有一般規則表達式。但未來我們可以努力方向是針對更高維度的情況給出一個較為優化的上界與下界。在經過討論之後，我們發現另一個研究方向是優化計算它的時間複雜度，也會嘗試著去改善其演算法。

捌、參考資料

1. http://www.artofproblemsolving.com/community

2. 冯志刚 數學奧林匹亞小叢書高中卷 6 第二版-數列與數學歸納法 九章出版社

3. 张垚 數學奧林匹亞小叢書高中卷 11 第二版-組合數學 九章出版社

4. 楊鈞任 大數模板

玖、附錄

以下為本篇報告中，利用程式演算法計算結果，其程式語法如下，計算程式檔實作，

大數模板 bignum.cpp：

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
class Bignum
{
    public:
    int num[100];
    int digit;
    int negative;
    Bignum()
    {
        digit = 1;
        for( int i = 0 ; i < 100 ; i++ )
            num[i] = 0;
        negative = 0;
    }
    Bignum( string s )
    {
        digit = s.length();
        for( int i = 0 ; i < s.length() ; i++ )
            num[i] = s[s.length()-i-1] - '0';
        for( int i = s.length() ; i < 100 ; i++ )
            num[i] = 0;
        negative = 0;
    }
    Bignum operator+( Bignum x )
    {
        Bignum ans;
        if(x.negative)for(int i = 0 ;i < x.digit ; i++)x.num[i]*=-1;
        int max_digit = max( digit, x.digit );
        for( int i = 0 ; i < max_digit ; i++ )
        {
            ans.num[i] += num[i] + x.num[i];
            ans.num[i+1] += ans.num[i] / 10;
            ans.num[i] %= 10;
```

```
    }
    ans.digit = max_digit+1;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator+( int x )
{
    char str[100];
    Bignum ans;sprintf(str,"%d",x);int len=strlen(str);
    int max_digit = max( digit, len );
    for( int i = 0 ; i < max_digit ; i++ )
    {
        ans.num[i] += num[i] + str[i]-'0';
        ans.num[i+1] += ans.num[i] / 10;
        ans.num[i] %= 10;
    }
    ans.digit = max_digit+1;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator-( Bignum x )
{
    Bignum ans;
    int max_digit = max( digit, x.digit );
    if( *this >= x )
    {
        for( int i = 0 ; i < max_digit ; i++ )
        {
            ans.num[i] = num[i] - x.num[i];
        }
```

```
    }
    else
    {
        for( int i = 0 ; i < max_digit ; i++ )
        {
            ans.num[i] = x.num[i] - num[i];
        }
        ans.negative = 1;
    }
    for( int i = 0 ; i < max_digit ; i++ )
    {
        if( ans.num[i] < 0 )
        {
            ans.num[i+1]--;
            ans.num[i] += 10;
        }
    }
    ans.digit = max_digit;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator-( int x )
{
    char str[100];
    sprintf(str,"%d",x);
    int len=strlen(str);
    Bignum ans;
    int max_digit = max( digit, len );
    if( *this >= x )
    {
        for( int i = 0 ; i < max_digit ; i++ )
        {
            ans.num[i] = num[i] - str[i]-'0';
        }
```

```
    }
    else
    {
        for( int i = 0 ; i < max_digit ; i++ )
        {
            ans.num[i] = str[i]-'0' - num[i];
        }
        ans.negative = 1;
    }
    for( int i = 0 ; i < max_digit ; i++ )
    {
        if( ans.num[i] < 0 )
        {
            ans.num[i+1]--;
            ans.num[i] += 10;
        }
    }
    ans.digit = max_digit;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator*( Bignum x )
{
    Bignum ans;
    for( int i = 0 ; i < digit ; i++ )
        for( int j = 0 ; j < x.digit ; j++ )
        ans.num[i+j] += num[i] * x.num[j];

    for( int i = 0 ; i < digit+x.digit+1 ; i++ )
    {
        ans.num[i+1] += ans.num[i] / 10;
        ans.num[i] %= 10;
    }
    ans.digit = digit+x.digit+1;
```

```
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator*( int x )
{
    Bignum ans;
    for( int i = 0 ; i < digit ; i++ )
        ans.num[i] += num[i] * x;
    for( int i = 0 ; i < 98 ; i++ )
    {
        ans.num[i+1] += ans.num[i] / 10;
        ans.num[i] %= 10;
    }
    ans.digit = 99;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
bool operator>=( Bignum x )
{
    if( digit > x.digit )
        return true;
    if( digit < x.digit )
        return false;
    for( int i = digit ; i >= 0 ; i-- )
        if( num[i] > x.num[i] )
            return true;
        else if( num[i] < x.num[i] )
            return false;
    return true;
}
```

```
bool operator>=( int x )
{
    char str[100];sprintf(str,"%d",x);int len=strlen(str);
    if( digit > len)
        return true;
    if( digit < len )
        return false;
    for( int i = digit ; i >= 0 ; i-- )
        if( num[i] > str[i]-'0' )
            return true;
        else if( num[i] < str[i]-'0' )
            return false;
    return true;
}
Bignum operator/( Bignum x )
{
    Bignum temp("0");
    Bignum ans;
    for( int i = digit-1 ; i >= 0 ; i-- )
    {
        string tempstr = "";
        tempstr += (char)(num[i]+'0');
        temp = temp * 10 + Bignum(tempstr);
        for( int j = 9 ; j > 0 ; j-- )
        {
            if( temp >= x*j )
            {
                temp = temp - x*j;
                ans.num[i] = j;
                break;
            }
        }
    }
    ans.digit = digit;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
```

```
    return ans;
}
Bignum operator/(int x)
{
    Bignum temp("0");
    Bignum ans;
    for( int i = digit-1 ; i >= 0 ; i-- )
    {
        string tempstr = "";
        tempstr += (char)(num[i]+'0');
        temp = temp * 10 + Bignum(tempstr);
        for( int j = 9 ; j > 0 ; j-- )
        {
            if( temp >= x*j )
            {
                temp = temp - x*j;
                ans.num[i] = j;
                break;
            }
        }
    }
    ans.digit = digit;
    while( !ans.num[ans.digit] && ans.digit >= 0 )
        ans.digit--;
    if( ans.digit < 1 )
        ans.digit = 1;
    else
        ans.digit++;
    return ans;
}
Bignum operator=(string s)
{
    Bignum x(s);
    return x;
}
void print()
{
    if( negative )
        cout << '-';
    for( int i = digit-1 ; i >= 0 ; i-- )
```

```
            cout << num[i];
    }
};
Bignum operator%(Bignum x,Bignum mod)
{
    Bignum k=x/mod;
    Bignum now=k*mod;
    return x-now;
}
Bignum operator%(Bignum x,int mod)
{
    Bignum k=x/mod;
    Bignum now=k*mod;
    return x-now;
}
```

主要程式 main.cpp：

```
#include<iostream>
#include<stdio.h>
#include"bignum.cpp"
using namespace std;
string str0="0",str1="1";
Bignum nul=str0,idn=str1,ans=nul,a[1010];
inline Bignum C(Bignum a,Bignum b)
{
    Bignum ans=idn,cnt=idn;
    while(b>=cnt)
    {
        ans=ans*(a-cnt+idn)/cnt;
        cnt=cnt+idn;
    }
    return ans;
}
inline Bignum H(Bignum a,Bignum b)
{
    return C(a+b-idn,b);
}
inline void bs(int l,int r)
{
    if(l+3>=r)return;
    int m=(l+r)/2;
```

```
    if((l+r)%2==0)
    {
        ans=ans+a[m];
        bs(l,m);
        bs(m,r);
    }
    else
    {
        ans=ans+a[m]+a[m+1];
        bs(l,m);
        bs(m+1,r);
    }
}
int main()
{
    string strn,strk;
    cin>>strn;cin>>strk;
    Bignum n=strn,k=strk,x=nul;
    int i;
    for(i=0;;i++,x=x+idn)
    {
        a[i]=nul;
        if((k*n-n>=x)==0)break;
        Bignum l=nul,t=x;
        while(true)
        {
            if(l%(idn+idn)>=idn)a[i]=a[i]-C(n,l)*H(n,t);
            else a[i]=C(n,l)*H(n,t)+a[i];
            if((t>=k)==0)break;
            t=t-k;
            l=l+idn;
        }
    }
    ans=nul;
    int la=0,r=i-1;
    ans=ans+a[la]+a[r];
    if(la>=r-1)ans=a[r];
    bs(la,r);
    ans.print();
return 0;}
```