Advanced Competitive Programming

國立成功大學ACM-ICPC程式競賽培訓隊
nckuacm@imslab.org

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

終極密碼

終極密碼

各位應該都聽過終極密碼

不管有沒有聽過，總之規則如下:

終極密碼

- 兩人以上的遊戲
- 其中一人 P， 0 ~ N (N>=1) 中選一個數字(**目標**)，別告訴其他人
- 其他人要想辦法**猜出**這個數字
- P 會告訴猜測者，目前猜的數字**大於**還是**小於**目標
- **等於**時遊戲結束

二分搜尋

- 一開始區間設定為 [0, N]

- 每次猜區間 [L, R] 內的中間值 M

- 如果目標 小於 M
- 區間改為 [L, M-1]，反之則改為 [M+1, R]

終極密碼: 二分搜尋

- 例如 [0, 99]，目標值為 42
- 猜 50，區間改 [0, 49]
- 猜 25，區間改 [26, 49]
- 猜 37，區間改 [38, 49]
- 猜 43，區間改 [37, 42]
- 猜 40，區間改 [41, 42]
- 猜 41，區間改 [42, 42]
- 猜 42

終極密碼: 二分搜尋

- 共 $\log_2 100 = 6.6438... \leqq 7$ 次猜測 (100 = 99+1)

- 這樣的猜法 複雜度為 O(log N)

lower & upper bound

推廣一下

如果現在有個**遞增數列**，目標出現**一個以上**

至少要有**兩個位置**(index)，以表達區間內都是目標

這兩個位置分別叫做:

- Lower bound
- Upper bound

區間

例如長度為 8， **起始位置**(index) 為 0
數列 1, 2, 2, 2, 2, 2, 3, 9
目標值為 2

輸出區間 [1, 5] 或是 [1, 6)
又或是 (0, 5]、(0, 6)

都能表達這個區間內的目標值 2

左閉右開

普遍的實作，
會採用 [1, 6) 這樣的**左閉右開**區間

左閉右開的好處，參考第四週的教材

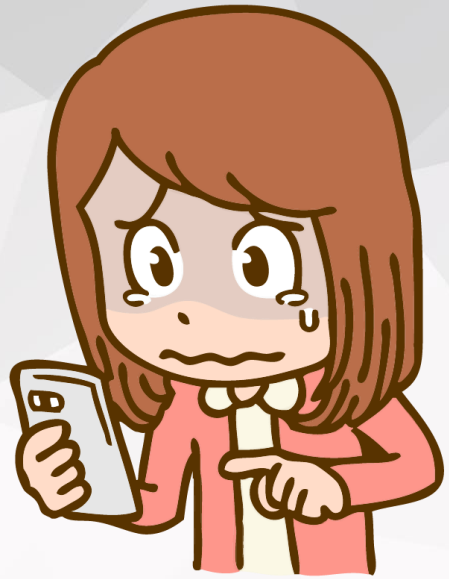
bound

數列 1, 2, 2, 2, 2, 2, 3, 9，目標值為 2

普遍實作中，

- lower bound 為 1
- upper bound 為 6

有個問題



目標值不在數列中怎麼辦?

例如數列 1, 2, 2, 2, 2, 2, 3, 9
當目標值為 42, -42, 5

當目標值不在數列中

數列 1, 2, 2, 2, 2, 2, 3, 9
目標值為 42

- lower bound 為 8
- upper bound 為 8

因為此時 42 若位於 index 8 的位置
1, 2, 2, 2, 2, 2, 3, 9, 42 這樣的數列依然保持遞增(**最適合**)

當目標值不在數列中

數列 1, 2, 2, 2, 2, 2, 3, 9
目標值為 -42

- lower bound 為 0
- upper bound 為 0

因為此時 -42 若位於 index 0 的位置
-42, 1, 2, 2, 2, 2, 2, 3, 9 這樣的數列依然保持遞增(**最適合**)

當目標值不在數列中

數列 1, 2, 2, 2, 2, 2, 3, 9
目標值為 5

- lower bound 為 7
- upper bound 為 7

因為此時 5 若位於 index 7 的位置
1, 2, 2, 2, 2, 2, 3, 5, 9 這樣的數列依然保持遞增(**最適合**)

一分搜尋 (Linear Search)

要如何找到 lower & upper bound 呢?

lower bound : 一分搜尋

數列 A: 1, 2, 2, 2, 2, 2, 3, 9

目標為 4

lower bound : 一分搜尋

數列 A: 1, 2, 2, 2, 2, 2, 3, 9
目標為 4

從最左邊開始找，找著找著碰到 9

lower bound : 一分搜尋

數列 A: 1, 2, 2, 2, 2, 2, 3, 9
目標為 4

從最左邊開始找，找著找著碰到 9
則 lower bound: 7

二分搜尋 (Binary Search)

高速枚舉的方法

lower bound : 二分搜尋

回到二等分の搜尋

跟終極密碼一樣，每次只找一半的區間

```
int m = (l+r) / 2; // m := middle
```

lower bound : 二分搜尋

數列 A: 1, 2, 2, 2, 2, 2, 3, 9

目標為 4

lower bound: 7

若 A[m] >= 4 則 r = m;　　// m 保留

若 A[m] < 4 則 l = m + 1; // m 捨去

lower bound : 二分搜尋

數列 A: 1, 2, 2, 2, 2, 2, 3, 9

目標為 4

lower bound: 7

若 `A[m] >= 4` 則 `r = m;` `// m 保留`

若 `A[m] < 4` 則 `l = m + 1;` `// m 捨去`

[l, m), [m+1, r)

lower bound : 二分搜尋

有個細節:

```
int m = (l+r) / 2;
```

這個 m 每次都會落在區間內
不會在 r 上，因為它是**開**的

原因是 int 除法會**無條件捨去小數位**

lower bound : 二分搜尋

```
while (l != r) {
   int m = (l+r) / 2;
   if (A[m] >= target) r = m;
   else l = m + 1;
}

return l;
```

TIOJ 1432 骨牌遊戲

觀察問題

直接的，先試試一分搜尋(枚舉)吧

從求值問題轉為判定問題

提出作法

```
int l = *max_element(s, s+n), r = maxn*maxn;
for (int i = l; i <= r; i++)
  if (check(i)) return i;
```

提出作法

接著將 check 函數寫出來

目的是要把**可行**的“最大傷害強度”挑出來

提出作法

```
bool check(int strength) {
  int cost = 0, cnt = 0;
  for (int i = 0; i < n; i++) {
    cost += s[i];
    if (cost > strength) cost = s[i], cnt++;
  }

  return cnt <= w;
}
```

觀察問題

可是這樣複雜度是 $O(N^3)$ 頗慢的

觀察問題

可是這樣複雜度是 $O(N^3)$ 頗慢的

注意到，只要給定的最大傷害強度越大
check 的回傳值越可能是 true

觀察問題

可是這樣複雜度是 $O(N^3)$ 頗慢的

注意到，只要給定的最大傷害強度越大
check 的回傳值越可能是 true

甚至，check 就是一個單調函數!
枚舉只要到第一次遇到 true 就能回傳答案了

提出作法

```
while (l != r) {
  int m = (l+r)/2;
  if (check(m)) r = m;
  else l = m+1;
}
return l;
```

TIOJ 1337 隕石

觀察問題 1

- 有**哪些**隕石需要射爆?

- 到底需要設**多少**防護罩?

兩個問題**似乎是依賴**的，
設越多防護罩，越不需要去射爆隕石
能射爆許多隕石，就不用設太多防護罩

觀察問題 1

- 到底需要設多少防護罩?

可以**枚舉**找出確切的值

- 有哪些隕石需要射爆?

在**判定**確切的值時，一同考慮進去

到底需要設**多少**防護罩?

設**足夠多**的防護罩，肯定能防止世界被破壞

也就是說，這個數量序列是**單調**的

有**哪些**隕石需要射爆?

影響範圍越大的隕石破壞的防護罩點位越多

位置防護罩夠多，就不需射爆破壞該位置的隕石

提出作法 1

用二分搜 搜出至少需要幾個防護罩

每次左至右將每個點被砸的次數算出來
看看防護罩夠不夠用，不夠用就射爆隕石
如果該射但不能再射，表示防護罩當初是不夠的

觀察問題 2

L_i, R_i 的**範圍頗大**

離散化!

最多也就 2 * N 個點位 (根據隕石範圍的左右界)

所以將**大範圍**映**射**到**小範圍**

提出作法 2

大範圍例如 [1, 5, 7, 10, 11]
將之映射為 [0, 1, 2, 3, 4, 5]

所以範圍 -1e9 ~ 1e9 將映射為小於 2e5 的範圍

三分搜尋 (Ternary Search)

三分搜尋

要怎麼找出**凹向上**二次函數的極值?

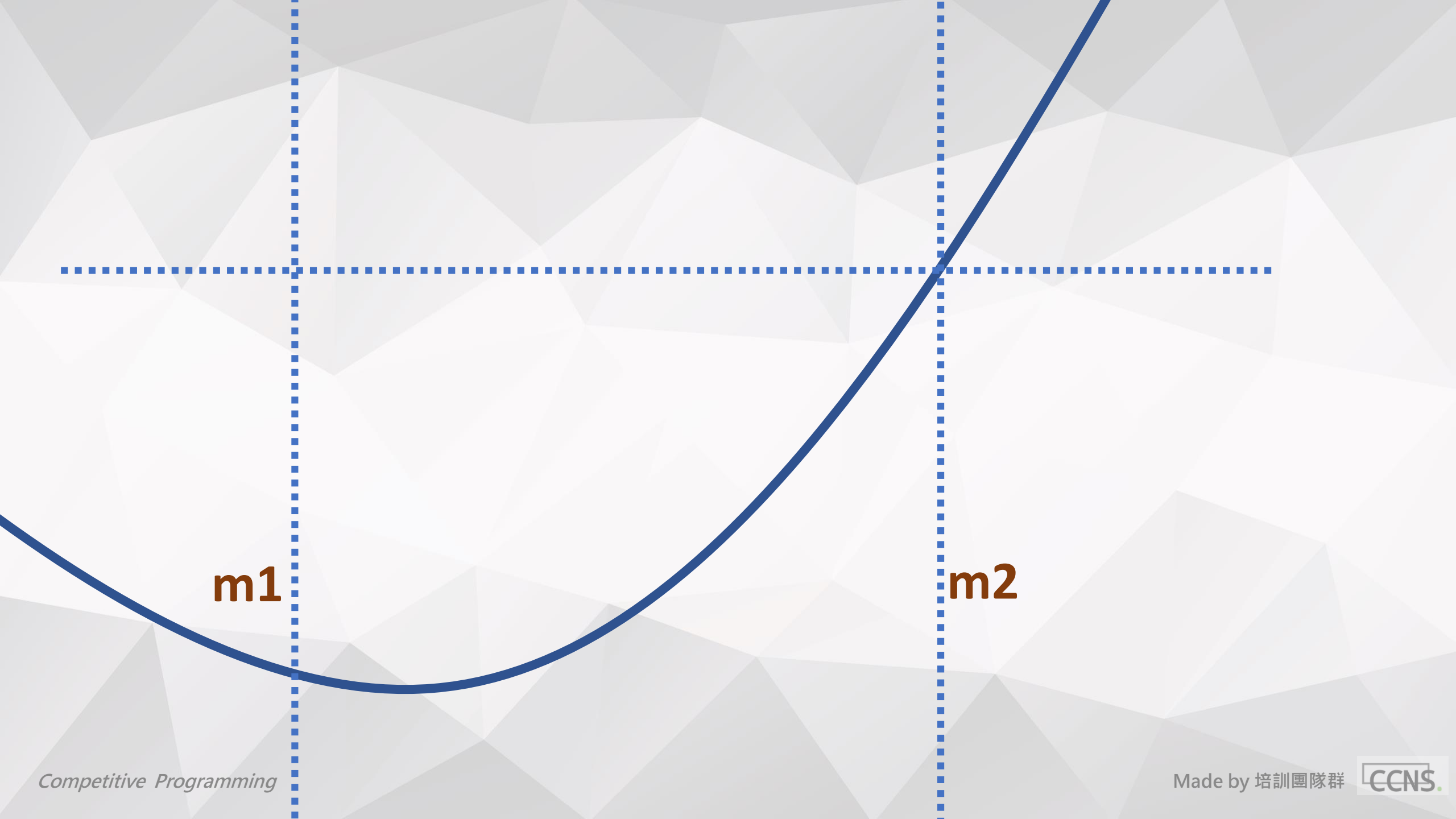
三分搜尋

枚舉，若遞減遞減到某點突然開始遞增，
那麼那個轉折點就是極點

函數有**一部分**的單調性


m1
m2


m1
m2


m1
m2

三分搜尋

```
// 凹向上
while(r-l < eps) {
   double m1 = (l+r)*(1/3);
   double m2 = (l+r)*(2/3);

   if(f(m1) < f(m2)) r = m2;
   else l = m1;
}
```

Longest Increasing Subsequence (LIS)

試著用二分搜優化吧!

Longest Increasing Subsequence (LIS)

在給定 N 長度序列 a，找到一個子序列，
為**嚴格遞增**且長度**最長**

例如 a = (1, 4, 2, 3, 8, 3, 4, 1, 9)
則 LIS 為 (1, 2, 3, 4, 9) 或 (1, 2, 3, 8, 9)

觀察問題

貪心的看，**當前** LIS 末項數字越小，
那麼**越有可能**使得 LIS 繼續變長

例如 (1,4,2) 有 LIS (1,4),(1,2) ，
但之後欲接 3，則只有 (1,2) 能接得 (1,2,3)

提出作法

嘗試紀錄 S(l) 為長度為 l 時 LIS 的最小末項

提出作法

嘗試紀錄 S(l) 為長度為 l 時 LIS 的最小末項

1, 4, 2, 3, 8, 3, 4, 1, 9

提出作法

嘗試紀錄 S(l) 為長度為 l 時 LIS 的最小末項

1, 4, 2, 3, 8, 3, 4, 1, 9

S(3) = 3

提出作法

嘗試紀錄 S(l) 為長度為 l 時 LIS 的最小末項

觀察問題

問題是如何給定 I?

觀察問題

問題是如何給定 l?

是因為有 a，使得 a > S(l-1)
才能得到 S(l) = a

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)
i = 2, S(2) = 4 (1, 4)

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)
i = 2, S(2) = 4 (1, 4)
i = 3, S(2) = 3 (1, 3)

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)
i = 2, S(2) = 4 (1, 4)
i = 3, S(2) = 3 (1, 3)

如果 a_i < S(l) => S(l) = a_i

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)
i = 2, S(2) = 4 (1, 4)
i = 3, S(2) = 3 (1, 3)
i = 4, S(3) = 4 (1, 3, 4)

觀察問題

1, 4, 3, 4, 2, 3, 4, 1, 9

i = 1, S(1) = 1 (1)
i = 2, S(2) = 4 (1, 4)
i = 3, S(2) = 3 (1, 3)
i = 4, S(3) = 4 (1, 3, 4)
i = 5, S(2) = 2 (1, 2)

提出作法

有這個末項 $S(l) = a_i$ 能知道上個末項是何者

提出作法

有這個末項 $S(l) = a_i$ 能知道上個末項是何者
於是能靠 $f(i) = p(l-1)$ 紀錄 LIS 順序

這裡 $p(l-1)$ 表示 $S(l-1)$ 的位置 $p(l) = i$

提出作法

有這個末項 $S(l) = a_i$ 能知道上個末項是何者
於是能靠 $f(i) = p(l-1)$ 紀錄 LIS 順序

這裡 $p(l-1)$ 表示 $S(l-1)$ 的位置 $p(l) = l$

l 長的最小末項表示在之前還有 $a_{p(l-1)}$, $a_{p(l-2)}$, ... 的接著

觀察問題

已解的 S 序列明顯是 單調的!!

提出作法

```
for(int i = 1; i <= n; i++) {
  int l = lower_bound(S+1, S+1+L, a[i]) - S;
  if(l > L) L++;

  p[l] = i;
  if(l > 1) f[i] = p[l-1];
  else f[i] = i;

  S[l] = a[i];
}
```

Longest Increasing Subsequence (LIS)

例如 a = (1, 4, 2, 3, 8, 3, 4, 1, 9)
得出 f = (1, 1, 1, 3, 4, 3, 4, 8, 5)

假設 $a_9 = 9$ 為末項的 LIS 為例：

$f_9 = 5 \rightarrow a_5 = 8$
$f_5 = 4 \rightarrow a_4 = 3$
$f_4 = 3 \rightarrow a_3 = 2$
$f_3 = 1 \rightarrow a_1 = 1$
$f_1 = 1$

印出 LIS

```
print_LIS(p[L]);
```

結語