CP Techniques

redleaf23477

Outline

- Bitwise Operation
- Modular Operation
- Prefix Sum
- Simple Sweeping Line
- Two Pointer
- Simple Greedy

Bitwise Operation

- bitwise and &
  - 7 & 2
- bitwise or |
  - 7 | 2
- bitwise xor ^
  - 7 ^ 2
- bitwise not ~
  - ~7
- shift left <<
  - 7 << 2
- shift right >>
  - 7 >> 2

$7 = (000111)_2$

$2 = (000010)_2$

Sets

使用整數 binary representation 表示 subset

第 i 個 bit = 1 ⇐⇒ S[i] 包含在子集裡面

S = {0, 1, 2, 3, 4, 5}

A = {0, 2, 4, 5} ⊆ S

a = $(110101)_2$ = 53

S	5	4	3	2	1	0
a	1	1	0	1	0	1
	2^5	2^4	2^3	2^2	2^1	2^0

Set Operations with Bitwise Operations

```
// membership query
bool in_set(int i, int s) { return (s >> i) & 1; }
// insert
int ins(int i, int s) { s ^= (1 << i); }
// remove
int rm(int i, int s) { s ^= (1 << i); }
// intersection
int inter (int s1 , int s2) { return s1 & s2; }
// union
int uni(int s1 , int s2) { return s1 | s2; }
// complement
int comp(int s) { return ~s; }
```

std::bitset

https://en.cppreference.com/w/cpp/utility/bitset

```
bitset<5> b1;             // 00000
bitset<5> b2(4);          // 00100
bitest<5> b3("01010");    // 01010

if (b1[2] == true) {
    cout << "Meow" << endl;
}
```

CSES 1745 - Money Sums

You have n coins with certain values. Your task is to find all money sums you can create using these coins.

Input

The first input line has an integer n: the number of coins.

The next line has n integers $x_1, x_2, \ldots, x_n$: the values of the coins.

Output

First print an integer k: the number of distinct money sums. After this, print all possible sums in increasing order.

Constraints

- $1 \leq n \leq 100$
- $1 \leq x_i \leq 1000$

Example

Input:
```
4
4 2 5 2
```

Output:
```
9
2 4 5 6 7 8 9 11 13
```

CSES 1745 - Money Sums

`dp[n][k] = true` ⇐⇒ 考慮前 n 個硬幣，面額 k 可以被創造出來

`dp[0][0] = true` base case

`dp[n][k] = dp[n-1][k] or dp[n-1][k-c[n]]`

面額 k 本來就可被前 n-1 個硬幣湊出

面額 k - c[n] 可被前 n-1 個硬幣湊出再補一個 n 號硬幣湊出面額 k

- 狀態數 O(NK), K = c[1] + c[2] + … + c[n]
- 狀態轉移 O(1)
- ⇒ 時間複雜度 O(NK)

CSES 1745 - Money Sums

`dp[n][k] = true` ⇐⇒ 考慮前 n 個硬幣，面額 k 可以被創造出來

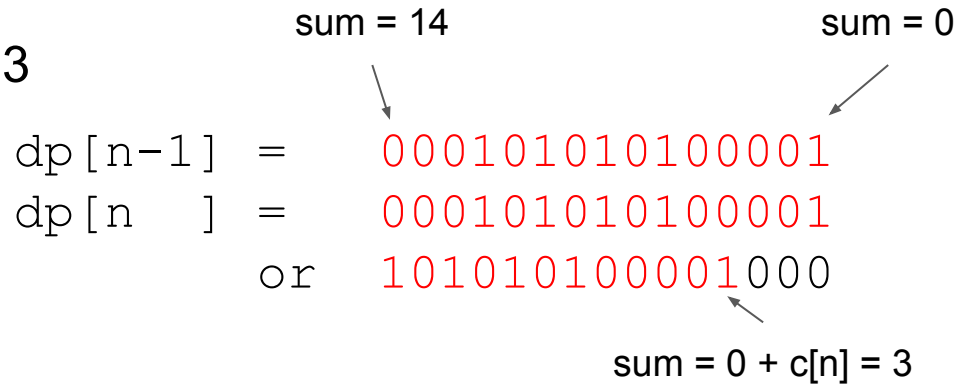
`dp[0][0] = true` base case

`dp[n][k] = dp[n-1][k] or dp[n-1][k-c[n]]`

面額 k 本來就可被前 n-1 個硬幣湊出

面額 k - c[n] 可被前 n-1 個硬幣湊出再補一個 n 號硬幣湊出面額 k

e.g. c[n] = 3



CSES 1745 - Money Sums

`dp[n][k] = true` ⇐⇒ 考慮前 n 個硬幣，面額 k 可以被創造出來

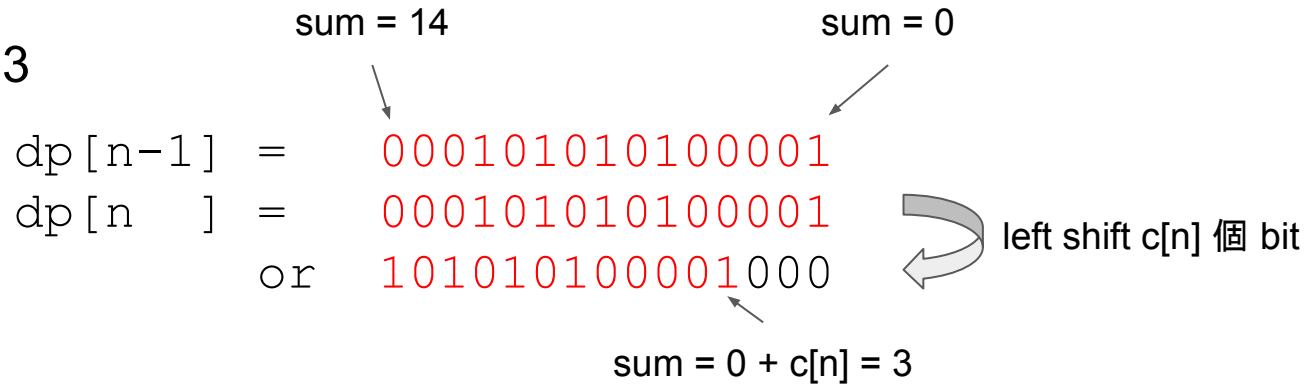
`dp[0] = bitset<100>(1)` base case (i.e. dp[0][0] = true)

`dp[n] = dp[n-1] or (dp[n-1] << c[n])`

面額 k 本來就可被前 n-1 個硬幣湊出

面額 k - c[n] 可被前 n-1 個硬幣湊出再補一個 n 號硬幣湊出面額 k

e.g. c[n] = 3



CSES 1745 - Money Sums

`dp[n][k] = true` ⇐⇒ 考慮前 n 個硬幣，面額 k 可以被創造出來

```
dp[0]    = bitset<100>(1)    base case (i.e. dp[0][0] = true)

dp[n]    = dp[n-1]     or (dp[n-1] << c[n])
```

面額 k 本來就可被前 n-1 個硬幣湊出

面額 k - c[n] 可被前 n-1 個硬幣湊出再補一個 n 號硬幣湊出面額 k

- 狀態數 O(NK), K = c[1] + c[2] + … + c[n]
- 狀態轉移 O(1)
  - 由於 bitset 優化，多個狀態可以一起轉移
- ⇒ 時間複雜度 O(NK)

Modular Operation

Output

Print the number of different ways to distribute the prizes modulo $10^9 + 7$.

Output

Print the number of different directed paths of length K in G, modulo $10^9 + 7$.

Output

Print one integer — the number of different strings which can be obta
once. Since the answer can be large, output it modulo 998244353.

Modular Operation

$a = q \cdot d + r$

$a \equiv r \bmod d$

- q: quotient (商)
- d: divisor (除數)
- r: remainder (餘數)

同餘性質:

1. 若 $a \equiv b \pmod m$, $b \equiv c \pmod m$
   則 $a \equiv c \pmod m$
2. 若 $a \equiv b \pmod m$, $c \equiv d \pmod m$
   則 $a + c \equiv b + d \pmod m$,
3. 若 $a \equiv b \pmod m$, $c \equiv d \pmod m$
   則 $ac \equiv bd \pmod m$

Modular Operation in C++

```cpp
// suppose 0 <= lhs, rhs < mod
int mod_add(int lhs, int rhs, int mod) {
  return (lhs + rhs) % mod;
}
int mod_sub(int lhs, int rhs, int mod) {
  return ((lhs - rhs) % mod + mod) % mod;
}
// be careful overflow
int mod_mul(int lhs, int rhs, int mod) {
  return lhs * rhs % mod;
}
```

Modular Operation in C++

Mod 運算 (%) 常數很大，有時會使用 if-else 以減少 mod 運算

(以下寫法只適用加減法，務必特別注意 lhs, rhs 的值域)

```
// suppose 0 <= lhs, rhs < mod
int mod_add(int lhs, int rhs, int mod) {
  lhs += rhs;
  return (lhs >= mod? lhs - mod : lhs);
}
int mod_sub(int lhs, int rhs, int mod) {
  lhs -= rhs;
  return (lhs < 0? lhs + mod : lhs);
}
```

Prefix Sum (前綴和)

idx	0	1	2	3	4	5
A		4	8	7	6	3
s0	0					
s1		4				
s2			12			
s3				19		
s4					25	
s5						28

Prefix Sum (前綴和)

給定一個序列 $A = (a_1, a_2, \ldots, a_n)$

有 m 筆詢問，詢問包含 L, R，請輸出 $\text{sum}(A[L, R]) = a_L + a_{L+1} + \ldots + a_R$

- $1 \le n, m \le 10^6$

TLE Solution:

- 對每一筆詢問：`for (int i = L; i <= R; i++) sum += A[i];`
- O(nm) 複雜度炸裂

Prefix Sum (前綴和)

高中數學:

- $S_1 = a_1$
- $S_i = S_{i-1} + a_i$
- $a_L + \ldots + a_R = S_R - S_{L-1}$

套用到程式:

- O(n) 存 Prefix Sum
- O(1) 算 sum(A[L, R])
- 時間複雜度:O(n + m)

```
vector<int> S(n+1, 0);
for (int i = 1; i <= n; i++) {
  S[i] = S[i-1] + A[i];
}

int sum = S[R] - S[L-1];
```

CSES 1661 Subarray Sums II

Given an array of n integers, your task is to count the number of subarrays having sum x.

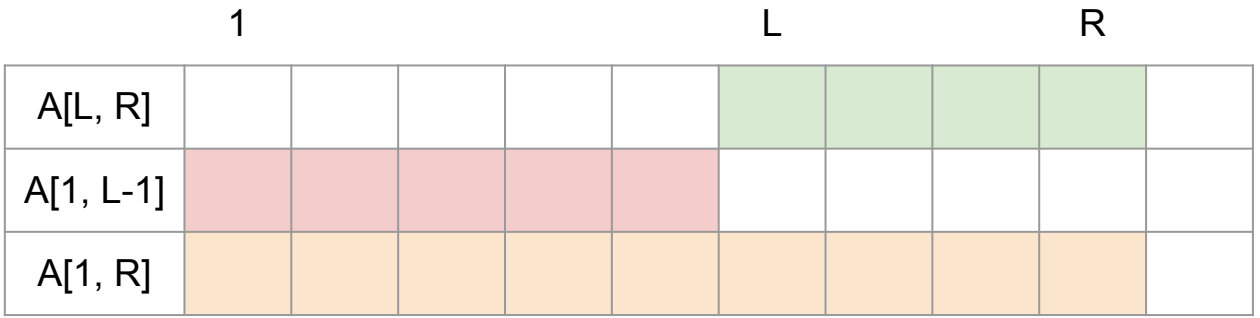
- $1 \le n \le 2 \cdot 10^5$
- $-10^9 \le x, a_i \le 10^9$

TLE Solution:

- 枚舉所有區間, 計算區間和
- $O(n^3)$ 或 $O(n^2)$, 總之都 TLE

CSES 1661 Subarray Sums II

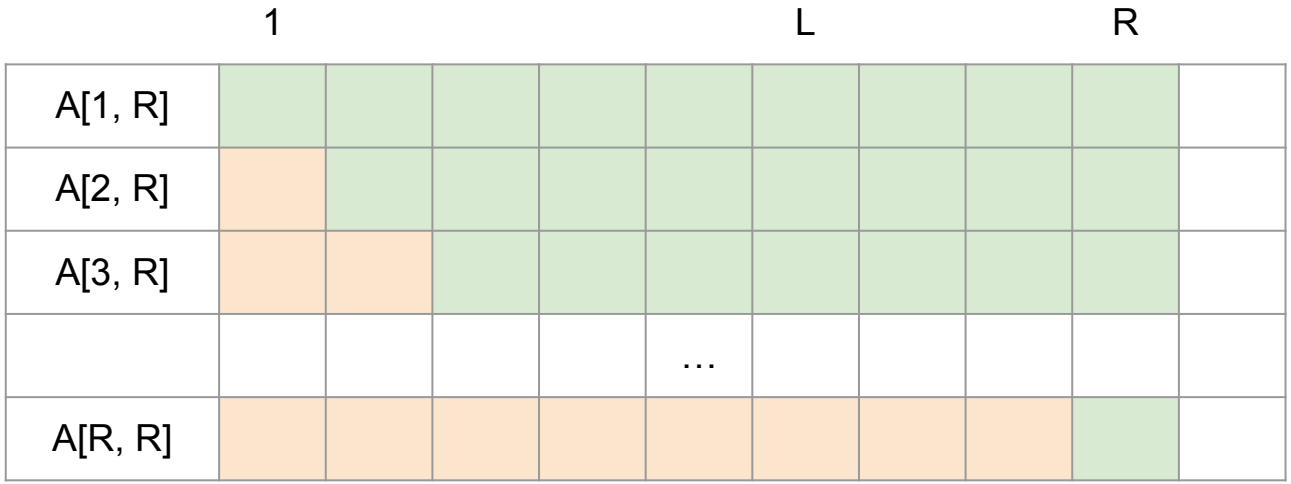
回憶一下區間和:



CSES 1661 Subarray Sums II

觀察所有結尾在 R 的區間:

他們的區間和恰為 $(S_R - S_0)$, $(S_R - S_1)$, $(S_R - S_2)$, …, $(S_R - S_{R-1})$



CSES 1661 Subarray Sums II

觀察所有結尾在 R 的區間:

他們的區間和恰為 $(S_R - S_0)$, $(S_R - S_2)$, $(S_R - S_3)$, …, $(S_R - S_{R-1})$

找有多少結尾在 R 的區間, 滿足區間和 = x

⇒ 有多少前綴和 S_i 滿足 $S_i = S_R - x$

⇒ 用 std::map 維護！

CSES 1661 Subarray Sums II

變數說明:

- `n, x` 同題目敘述
- `arr` 存輸入序列
- `mp[s]` = #前綴和為 s 的區間
- `pre` 存當前的前綴和

時間複雜度:O(n lg n)

- 迴圈跑 O(n) 次
- 每次計算花 O(lg n)

```
LL ans = 0, pre = 0;
map<LL, int> mp; mp[0] = 1;
for (auto a : arr) {
  pre += a;
  if (mp.count(pre - a) != 0) {
    ans += mp[pre - x];
  }
  mp[pre] += 1;
}
cout << ans << endl;
```

Atcoder DP Educational Contest, M - Candies

Problem Statement

There are N children, numbered $1, 2, \ldots, N$.

They have decided to share K candies among themselves. Here, for each i ($1 \leq i \leq N$), Child i must receive between 0 and a_i candies (inclusive). Also, no candies should be left over.

Find the number of ways for them to share candies, modulo $10^9 + 7$. Here, two ways are said to be different when there exists a child who receives a different number of candies.

Constraints

- All values in input are integers.
- $1 \leq N \leq 100$
- $0 \leq K \leq 10^5$
- $0 \leq a_i \leq K$

Atcoder DP Educational Contest, M - Candies

`dp[n][k]` = 考慮將 `k` 個糖果分給前 `n` 個人的方法數

`dp[0][0] = 1` base case

`dp[n][k] = dp[n-1][k-0] + dp[n-1][k-1] + … + dp[n-1][k-a[n]]`

分 0 個給 n 號人　　分 1 個給 n 號人　　分 a[n] 個給 n 號人

- 狀態數 O(NK)
- 狀態轉移 O(K)
- ⇒ 時間複雜度 $O(NK^2)$
- 怎麼優化?

Constraints

- All values in input are integers.
- $1 \leq N \leq 100$
- $0 \leq K \leq 10^5$
- $0 \leq a_i \leq K$

Atcoder DP Educational Contest, M - Candies

Notes: MOD Operations

AC, 122 ms ⇒

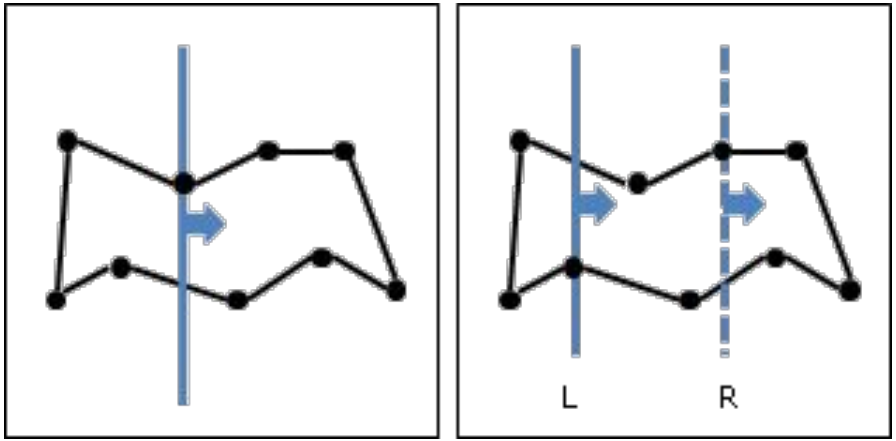
```
// suppose 0 <= lhs, rhs < mod
int mod_add(int lhs, int rhs, int mod) {
  return (lhs + rhs) % mod;
}
int mod_sub(int lhs, int rhs, int mod) {
  return ((lhs - rhs) % mod + mod) % mod;
}
```

AC, 91 ms ⇒

```
// suppose 0 <= lhs, rhs < mod
int mod_add(int lhs, int rhs, int mod) {
  lhs += rhs;
  return (lhs >= mod? lhs - mod : lhs);
}
int mod_sub(int lhs, int rhs, int mod) {
  lhs -= rhs;
  return (lhs < 0? lhs + mod : lhs);
}
```

Sweeping Line

2D 平面上，一條直線沿著某方向掃過平面上所有點或線段



CSES 1619 Restaurant Customers

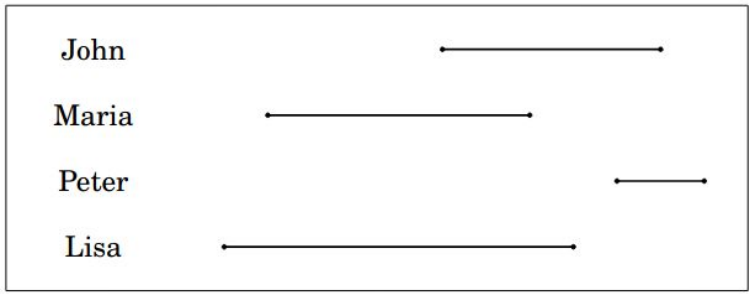
You are given the arrival a and leaving b times of n customers in a restaurant.

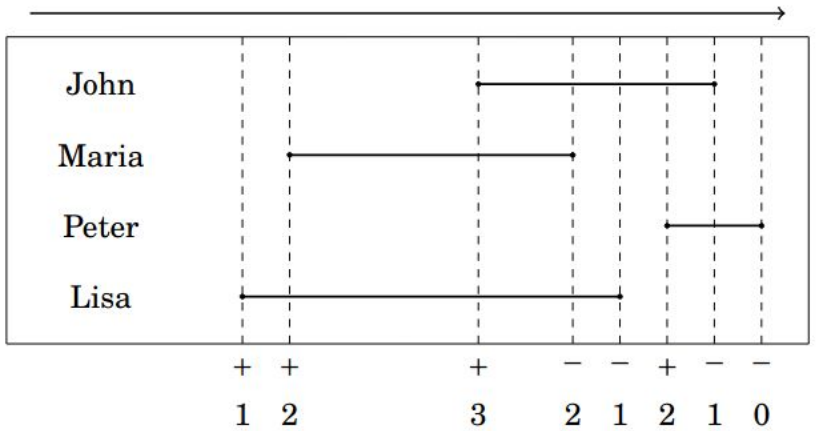
What was the maximum number of customers in the restaurant at any time?

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq a < b \leq 10^9$

CSES 1619 Restaurant Customers

person	arrival time	leaving time
John	10	15
Maria	6	12
Peter	14	16
Lisa	5	13





CSES 1619 Restaurant Customers

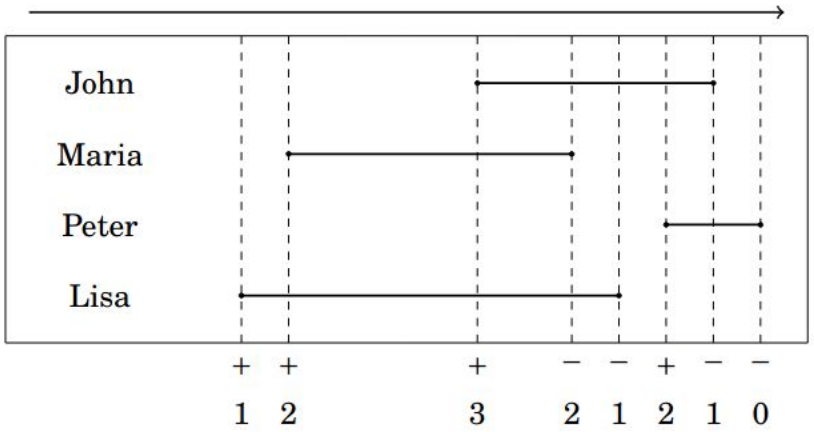
```
vector<int> timeline(N, 0);
for (int i = 0; i < n; i++) {
  timeline[a[i]]++;
  timeline[b[i]+1]--;
}

int ans = 0, pre = 0;
for (int i = 0; i < N; i++) {
  pre += timeline[i];
  ans = max(ans, timeline[i]);
}
```

person	arrival time	leaving time
John	10	15
Maria	6	12
Peter	14	16
Lisa	5	13

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
					+1	+1				+1			-1	+0		-1	-1
Σ	0	0	0	0	1	2	2	2	2	3	3	3	2	2	2	1	0

CSES 1619 Restaurant Customers



person	arrival time	leaving time
John	10	15
Maria	6	12
Peter	14	16
Lisa	5	13

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
					+1	+1				+1			-1	+0		-1	-1
Σ	0	0	0	0	1	2	2	2	2	3	3	3	2	2	2	1	0

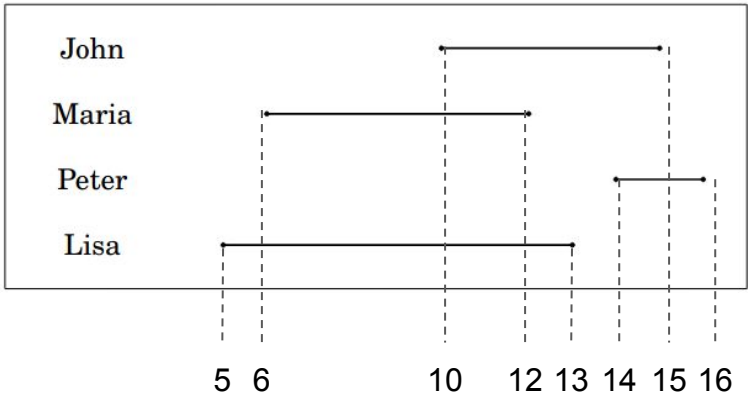
CSES 1619 Restaurant Customers

```
vector<int> timeline(N, 0);
for (int i = 0; i < n; i++) {
  timeline[a[i]]++;
  timeline[b[i]+1]--;
}

int ans = 0, pre = 0;
for (int i = 0; i < N; i++) {
  pre += timeline[i];
  ans = max(ans, timeline[i]);
}
```

timeline 的 N 要開多大？

- $1 \le n \le 2 \cdot 10^5$
- $1 \le a < b \le 10^9$



Relabel 離散化

給定一個數列:

- 數值大小不重要
- 元素之間大小關係重要
- 將序列中的數值改為在序列中的名次
- 縮小值域
  - $n \le 10^6, a_i \le 10^9 \rightarrow a_i' \le n$

$$A = (4, 8, 7, 6, 6, 6, 3) \rightarrow A' = (1, 4, 3, 2, 2, 2, 0)$$

Relabel with STL

```
void relabel(vector<int> &arr) {
  set<int> s(arr.begin(), arr.end());
  map<int,int> mp;
  int idx = 0;
  for (auto x : s) mp[x] = idx++;
  for (auto &x : arr) x = mp[x];
}
```

Relabel with Binary Search

```
void relabel(vector<int> &arr) {
  vector<int> tmp(arr);
  sort(tmp.begin(), tmp.end());
  auto it = unique(tmp.begin(), tmp.end());
  tmp.erase(it, tmp.end());
  for (auto &x : arr) {
    x = lower_bound(tmp.begin(), tmp.end(), x) - tmp.begin();
  }
}
```

CSES 1619 Restaurant Customers

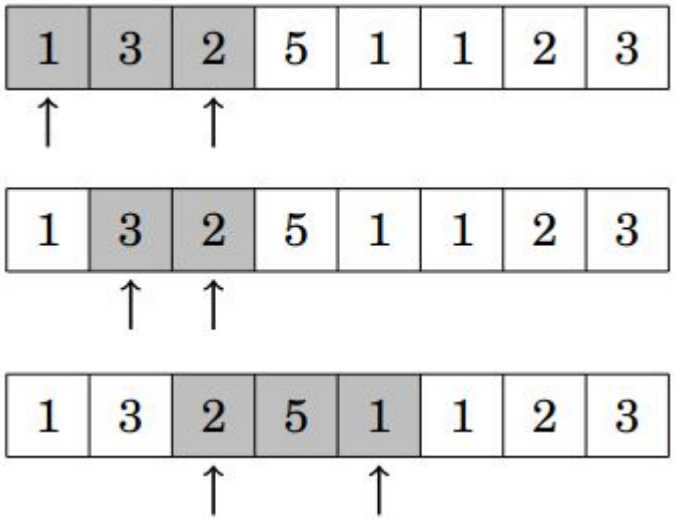
解題架構:

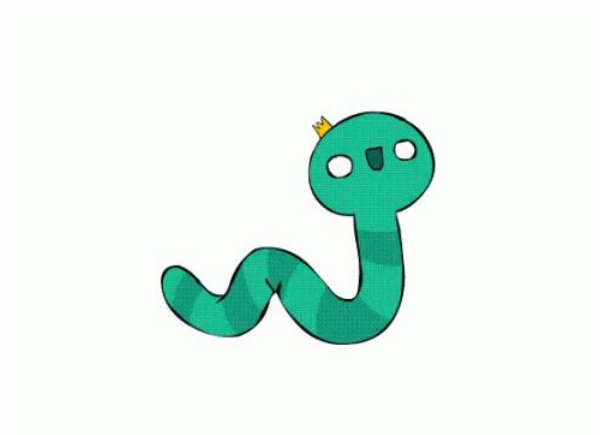
1. 對 arrival time 和 leave time 離散化 … O(n lg n)
2. 套用 Sweeping line … O(n)

時間複雜度:O(n lg n + n) = O(n lg n)

Two Pointer (爬行法)

用兩個 pointer 走過整個陣列，過程中兩個 pointer 只能 ++





CSES 1660 - Subarray Sums I

Given an array of n positive integers, your task is to count the number of subarrays having sum x.

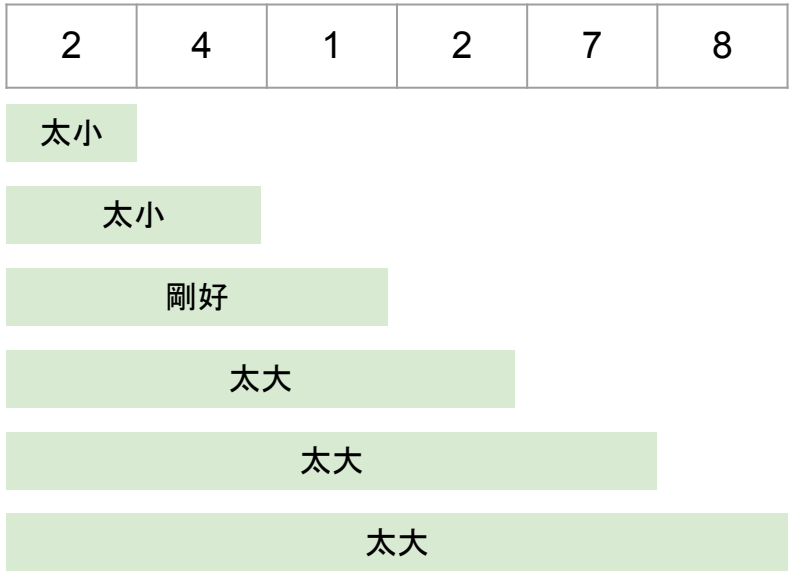
- $1 \le n \le 2 \cdot 10^5$
- $1 \le x, a_i \le 10^9$

CSES 1660 - Subarray Sums I

暴力枚舉所有區間 $O(n^2)$ … TLE

觀察一下 :

e.g. x = 7, arr = (2, 4, 1, 2, 7, 8)

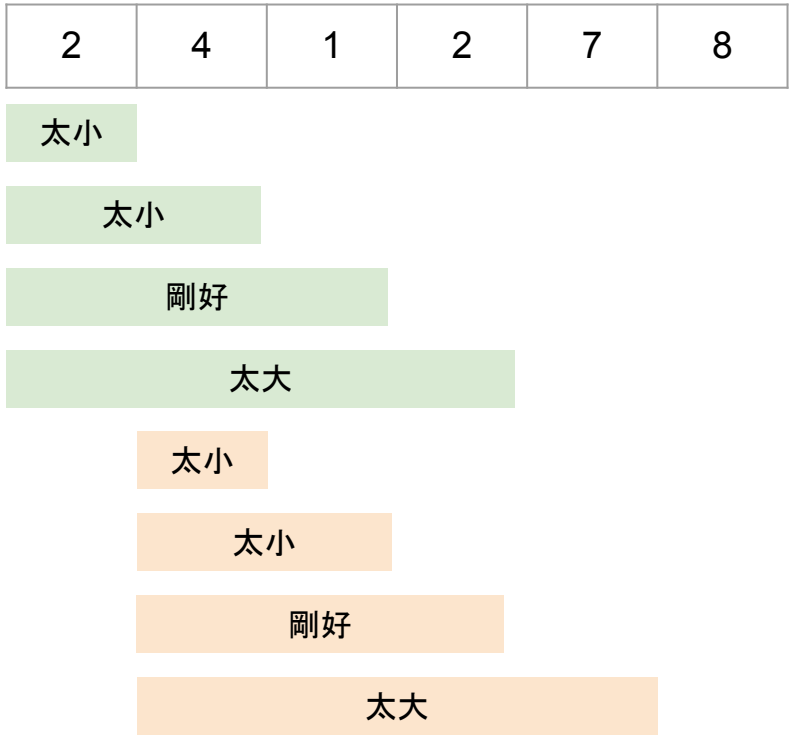


CSES 1660 - Subarray Sums I

暴力枚舉所有區間 $O(n^2)$ … TLE

觀察一下 :

e.g. x = 7, arr = (2, 4, 1, 2, 7, 8)

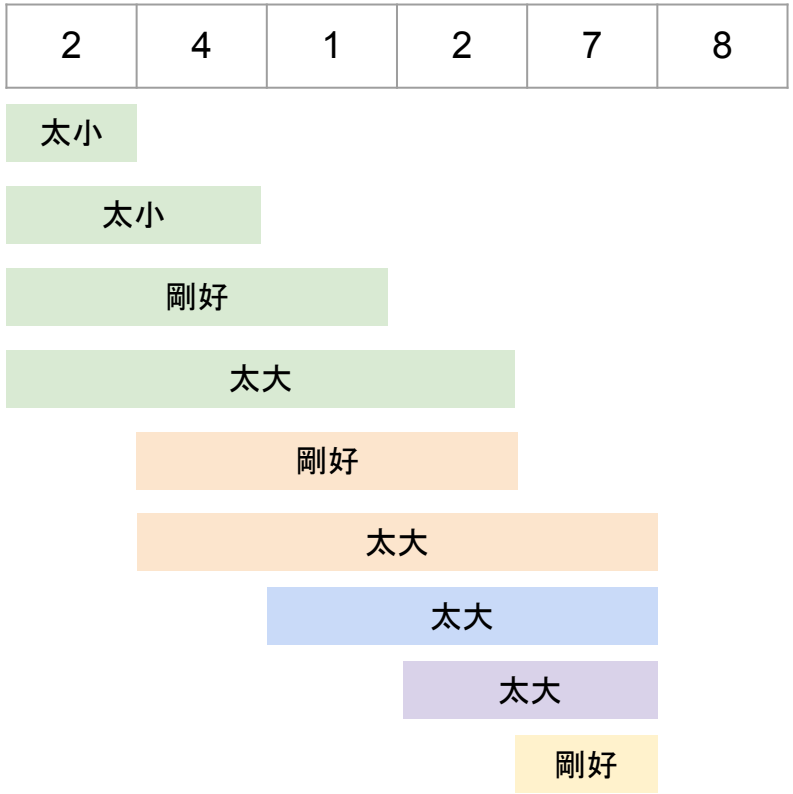


CSES 1660 - Subarray Sums I

暴力枚舉所有區間 $O(n^2)$ … TLE

觀察一下 :

e.g. x = 7, arr = (2, 4, 1, 2, 7, 8)



CSES 1660 - Subarray Sums I

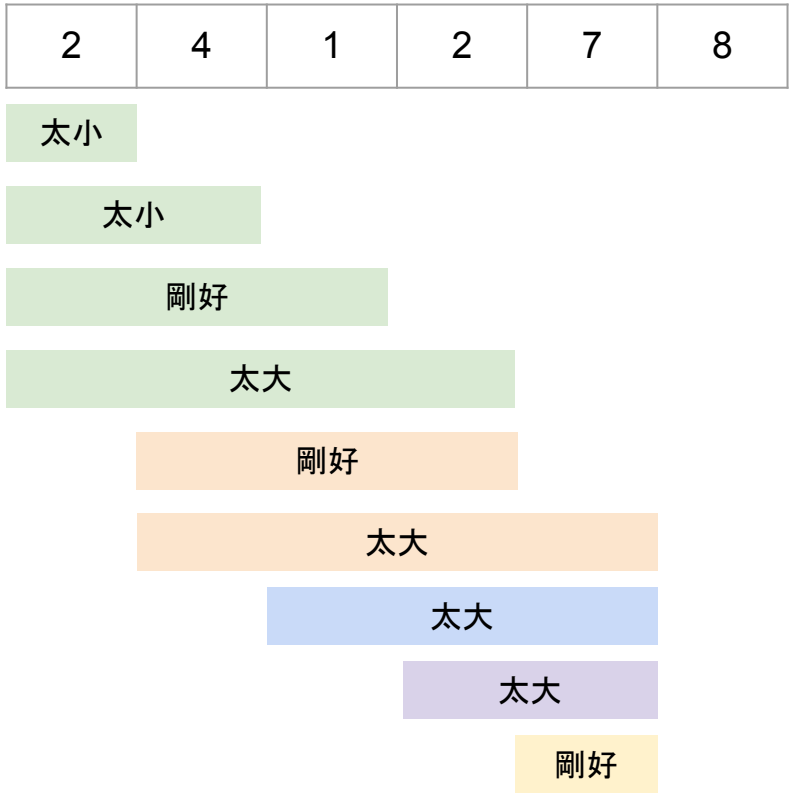
暴力枚舉所有區間 $O(n^2)$ … TLE

觀察一下：

e.g. x = 7, arr = (2, 4, 1, 2, 7, 8)

動左指標 L 和右指標 R 的規則：

- 維護 [L, R] 區間和
- 區間和太小：R++
- 區間和太大：L++



CSES 1660 - Subarray Sums I

時間複雜度: 均攤 O(n)

- L 只會從 1 加到 n
- R 只會從 1 加到 n
- 整個過程只會動約 2n 次
- [L, R] 區間和可 O(1) 維護

```
LL ans = 0, sum = arr[0];
for (int L = 0, R = 0; L < n; L++) {
  while (R+1 < n && sum < s) {
    sum += arr[R+1];
    R++;
  }
  if (sum == s) ans += 1;
  sum -= arr[L];
}
cout << ans << "\n";
```

CSES 1141 - Playlist

Given n songs where the id of the songs are k_1, k_2, …, k_n

What is the longest sequence of successive songs where each song is unique?

- 如何套用 Two Pointer？
- [L, R] 夾的區間需要維護什麼性質？
- 動 L, R 的規則？

CSES 1640 / 1641 / 1642 - Sum of 2 / 3 / 4 Values

Try to use two pointers to AC them

Greedy

有 n 桶冰淇淋，第 i 種口味有 w_i 公克，滿足度 v_i 每公克

已知甜筒最多能裝 m 公克，最多能裝的滿足度為多少？

(每種口味可以裝任意重量的冰淇淋，w_i, v_i, m 都是整數)

⇒ Ans: 從**最好吃**的開始裝，裝完換次好吃的 …

Greedy

有 n 條薯條，第 i 條重量 w_i 公克，滿足度 v_i

已知紙盒最多能裝 m 公克，最多能裝多少滿意度的薯條？

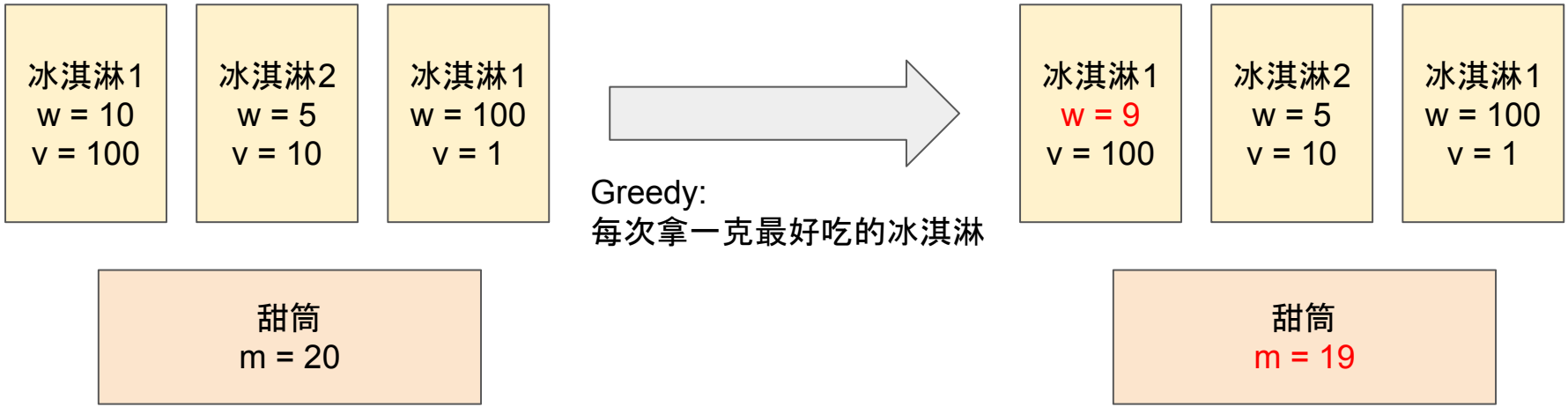
(每次要裝完整的薯條, w_i, v_i, m 都是整數)

⇒ Ans: 每次挑**最大 (v_i / w_i) 值**的開始裝 … ???

Greedy

最佳化問題中，每一步有多個選擇，貪心演算法會挑**當前看起來最好**的選擇

(做完一次貪心選擇後，得到一個一模一樣但是比較小的子問題)



Greedy

最佳化問題中，每一步有多個選擇，貪心演算法會挑**當前看起來最好**的選擇

(做完一次貪心選擇後，得到一個一模一樣但是比較小的子問題)

Greedy 會不會得到最佳解？ 如何證明?

- 為什麼貪心選擇最好? … (反證法)
  - Suppose not: 我不挑最好吃的冰淇淋 … Contradiction
  - ⇒ 我一定可以挑最好吃的冰淇淋
- 一直做貪心選擇可以得到最佳解? … (數學歸納法)
  - Base case: 當我的甜筒大小只有 m = 1 時，我挑最好吃的可以得到最佳解
  - Inductive case: 當我的甜筒大小有 m = k (k > 1) 時，假設 Greedy 可以得到 m < k 時的最佳解
  - … ⇒ Greedy 也可以得到 m = k 時的最佳解

CSES 1629 - Movie Festival

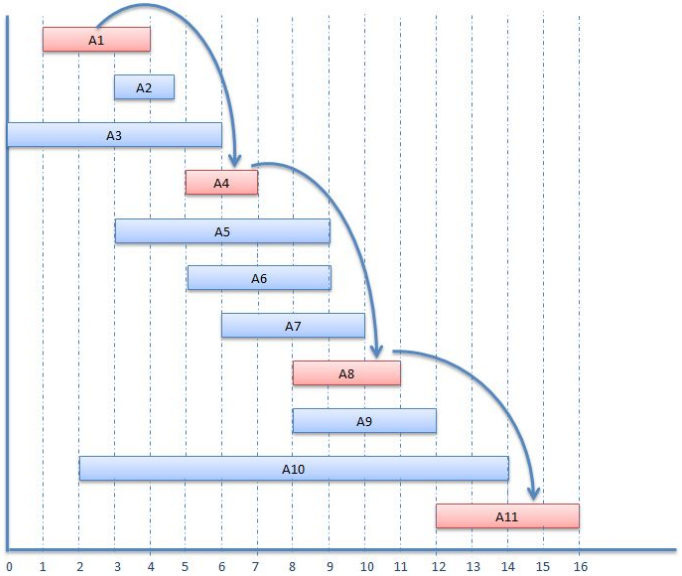
In a movie festival n movies will be shown. You know the starting and ending time of each movie. What is the maximum number of movies you can watch entirely?

- 每次挑最短的?
- 每次挑最早開始的?
- 每次挑最早結束的?

CSES 1629 - Movie Festival

Greedy Algorithm:

1. 按照結束時間排序
2. 先挑最早結束的電影
   再挑剩下不重疊的電影中最早結束的



為什麼正確？

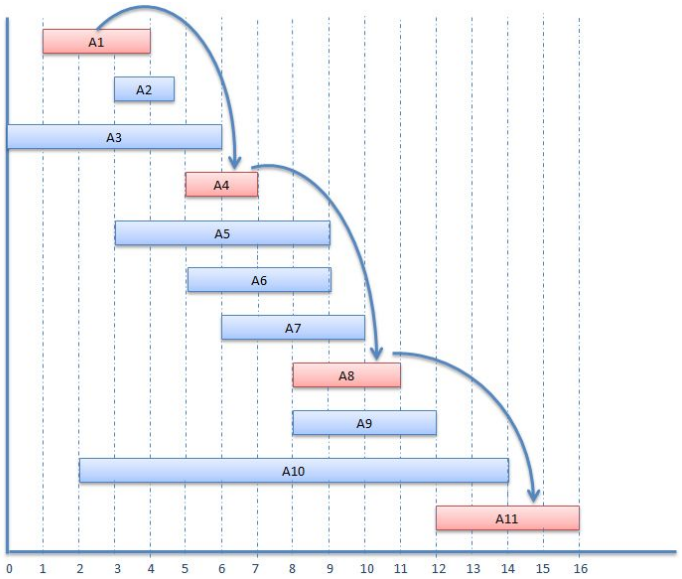
⇒ 去看 CLRS Ch 16.1

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：

1. a_m 必定包含在某個最佳解當中
2. sol(S) = 1 + sol(S - {a_m} - {重疊 a_m 的電影})



CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：1. a_m 必定包含在某個最佳解當中

- 反證法：假設所有最佳解皆不包含 a_m

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：1. a_m 必定包含在某個最佳解當中

- 反證法：假設所有最佳解皆不包含 a_m
- 考慮任一最佳解所包含的電影為 A，其中最早結束的電影為 a_i

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：1. a_m 必定包含在某個最佳解當中

- 反證法：假設所有最佳解皆不包含 a_m
- 考慮任一最佳解所包含的電影為 A，其中最早結束的電影為 a_i
- 令 A' = A - $\{a_i\}$ ∪ $\{a_m\}$，A' 中的電影互不重疊，A' 是一個包含 a_m 的合法解

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：1. a_m 必定包含在某個最佳解當中

- 反證法：假設所有最佳解皆不包含 a_m
- 考慮任一最佳解所包含的電影為 A，其中最早結束的電影為 a_i
- 令 A' = A - $\{a_i\}$ ∪ $\{a_m\}$，A' 中的電影互不重疊，A' 是一個包含 a_m 的合法解
- |A'| = |A| - 1 + 1 = |A|（矛盾）

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：1. a_m 必定包含在某個最佳解當中

- 反證法：假設所有最佳解皆不包含 a_m
- 考慮任一最佳解所包含的電影為 A，其中最早結束的電影為 a_i
- 令 A’ = A - $\{a_i\}$ ∪ $\{a_m\}$，A’ 中的電影互不重疊，A’ 是一個包含 a_m 的合法解
- |A’| = |A| - 1 + 1 = |A|（矛盾）
- 得證：存在最佳解包含 a_m

CSES 1629 - Movie Festival

令當前電影集合 S，其中最早結束的電影為 a_m

證明：2. sol(S) = 1 + sol(S - {a_m} - {重疊 a_m 的電影})

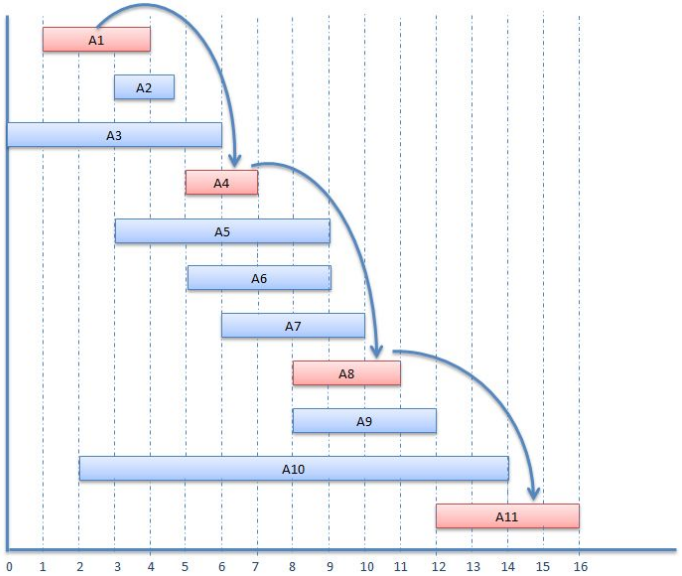
CSES 1629 - Movie Festival

Greedy Algorithm:

1. 按照結束時間排序
2. 先挑最早結束的電影
   再挑剩下不重疊的電影中最早結束的

證明:

1. a_m 必定包含在某個最佳解當中
2. sol(S) = 1 + sol(S - {a_m} - {重疊 a_m 的電影})



UVa 10954 - Add all

給定 n 個數字，每次將兩個數字 a, b 合併為 (a+b)，並須付出成本 (a+b)

求合併 n 個數字所需最低成本

$1 + 2 = 3$, cost $= 3$ $3 + 3 = 6$, cost $= 6$ Total $= 9$	$1 + 3 = 4$, cost $= 4$ $2 + 4 = 6$, cost $= 6$ Total $= 10$	$2 + 3 = 5$, cost $= 5$ $1 + 5 = 6$, cost $= 6$ Total $= 11$

- Greedy: 每次挑最小的兩個數字合併
- 證明: 令當前數字集合 S
  - 存在最佳解在其中一步合併 S 中最小的兩個數字
  - sol(S) = (a+b) + sol(S - {a, b} + {a+b})

同場加映：Greedy?

有 n 顆蘋果，第 i 顆重量 w_i 公克，滿足度 v_i

你想從中挑恰好 k 棵蘋果 (你一定要挑整顆，不能切，不能榨汁)

使得最後 CP 值 (= 滿足度加總 / 重量加總) 最大

⇒ Ans: ???