

Eolymp | 657 | Playing with coins

Nama : Timothy Hosia Budianto

NRP : 50525211098

Permasalahan dan pendekatan :

Playing with coins

Coins in denominations of one penny decomposed into stacks (a stack may be a different number of coins), and the stack placed on the table in a row from left to right. Two opponents take turns making moves. Progress lies in the fact that one player takes on the left several piles in a row, no less than one nor more than before that he took his opponent. The first player takes his first move no more than K stacks. The game ends when the cups are left. Required to find the maximum number of coins that can get the first party after the game, if the second player also tries to walk, so to get as many coins.

Input

The first line is a first number of stacks N , follow him: for N numbers giving the number of coins in the stacks from left to right, then the number of K . All the numbers in the row are separated by spaces.

$1 \leq N \leq 180$, $1 \leq K \leq 80$, number of coins in the pile - at least 1 and not more than 20 000.


Output

Derive a single number - the maximum number of coins that can certainly get the first player.


 Time limit 1 second

 Memory limit 64 MiB

Di soal ini digambarkan ada 2 orang yang sedang bermain koin. Akan diberikan beberapa stack koin sejumlah n . kemudian secara bergantian dimulai dari player satu akan mengambil koin sebanyak maksimal k stack dari kiri ke kanan. Syarat lainnya adalah pemain tidak diperbolehkan untuk mengambil stack lebih dari pemain sebelumnya. Soal ini menanyakan berapa koin paling banyak yang bisa diambil oleh pemain pertama. Soal ini memiliki time limit 1 detik dan memory 64 mb.

Input example #1 

```
3 4 9 1 3
```

Output example #1 

```
14
```

Pada sample test case pertama dilihat bahwa pemain pertama memiliki batas mengambil 3 stack koin, sehingga ia mengambil 3 stack pertama yaitu $4 + 9 + 1 = 14$.

Abstraksi :

Untuk lebih memahami kode kita akan menggunakan ilustrasi untuk menjelaskan soal. Misal dengan input seperti dibawah. N merupakan banyak stack koin, table hijau berisi berapa koin di tiap tiap stack, dan nilai k adalah jumlah stack maksimal yang bisa diambil oleh pemain 1 di Langkah pertama.

Input	N						K
	5	1	2	3	4	5	3

Berikut adalah Langkah yang bisa diambil, dengan warna biru adalah pemain 1, dan merah adalah pemain 2.

kemungkinan 1	1	2	3	4	5	sum = 6
kemungkinan 2	1	2	3	4	5	sum = 8
kemungkinan 3	1	2	3	4	5	sum = 9

Bisa dilihat pada kemungkinan 1, pemain 1 mengambil 3 stack (maximal) dan pemain berikutnya mengabil 2 stack.

Pada kemungkinan 2, pemain 1 mengambil 2 stack, sehingga pada Langkah berikutnya pemain 2 hanya bisa mengambil stack ≤ 2 . Tetapi pemain 2 bertujuan untuk juga mengambil koin terbanyak yang mengakibatkan ia mengambil 2 stack bernilai 7 dan pemain 1 mengambil stack terakhir yang mengakibatkan pemain 1 mengambil 8 koin.

Pada kemungkinan 3, pemain 1 mengambil 1 stack saja yang mengakibatkan stack yang diambil berseling- selingan, mengakibatkan pemain 1 dapat koin paling banyak dibanding metode lain yaitu 9 koin.

Untuk menyelesaikan persoalan diatas tidak mungkin dilakukan secara brute force mencoba semua kemungkinan karena akan membutuhkan waktu yang terlalu lama. Oleh karena itu kita akan menggunakan metode **dynamic programming**. Pada umumnya, pendekatan dynamic programming selalu bergerak dari indeks kecil ke indeks besar atau secara incremental. Namun

beberapa kasus soal, pendekatan incremental tersebut bisa jadi kurang efisien, maka dari itu, dapat juga digunakan pendekatan decremental contohnya pada soal ini.

Pseudocode:

```
n ← (scan value)
for i= 0 until n – 1
    a[i] ← (scan value)
for i = n – 1 until i >= 0
    sum[i] := sum[i + 1] + a[i]
    i--
k ← (scan value)
for i = 1 until i = k
    for j = n – i until j >=0
        dp[j] := max(sum[j] – dp[j + i], dp[j])
    j--
print dp[0]
```

Implementasi Code

```
1  #include <stdio.h>
2  #include <algorithm>
3  int dp[185], sum[190], n, a[185], k;
4
5  int main(){
6      scanf("%d", &n);
7      for(int i=0; i<n; i++) scanf("%d", &a[i]);
8      for(int i=n-1; i>=0; i--)
9          sum[i] = sum[i+1] + a[i];
10     scanf("%d", &k);
11     for(int i=1; i<=k; i++){
12         for(int j=n-i; j>=0; j--){
13             dp[j] = std::max(sum[j] - dp[j+i], dp[j]);
14         }
15     }
16     printf("%d\n", dp[0]);
17     return 0;
18 }
```

Dengan Test Case

Input example #1

3 4 9 1 3

Output example #1

14

Line 1-3 adalah library dan variable yang akan digunakan

Line 6 input n

Line 7 input jumlah koin tiap stack

Line 8-9 kan menggunakan looping untuk menghitung dan disimpan di array sum. Dengan test case diatas akan menghasilkan

`sum[2] = 1`

`sum[1] = 10`

`sum[0] = 14`

Selanjutnya line 11-14 akan mengscan value k. Kemudian dilakukan looping sebanyak K untuk menghitung kemungkinan maksimum jumlah stack yang dapat diambil oleh pemain pertama.

Nilai tiap dp yang dihasilkan adalah sebagai berikut

`dp[2] = 1`

`dp[1] = 9`

`dp[0] = 5`

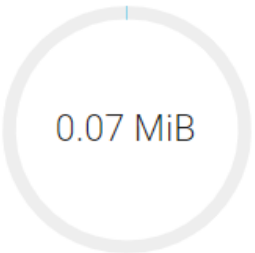
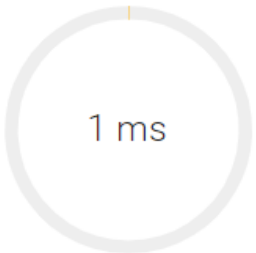
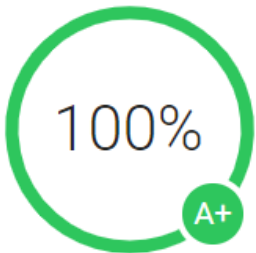
`dp[1] = 10`

`dp[0] = 13`

`dp[0] = 14`

Nilai dari tiap kemungkinan akan coba dihitung. Nilai maksimal stack koin yang dapat diambil oleh pemain 1 terdapat pada dp[0]. Hal tersebut terjadi karena kita melakukan perhitungan tiap tiap kemungkinan dari indeks paling akhir (backward) dan tiap iterasi rekurensnya, dicari nilai maksimum tiap langkah yang memungkinkan. Dan nilai maksimum pemain pertama ada di indeks ke-0 karena ia selalu melangkah lebih dulu. Lalu di print nilai yaitu 14

Problem	Submitted	Programming Language	Author
Playing with coins	7 hours ago	C++ 17 (gnu 10.2)	TimothyHosia_5025211098



Your submission was graded A+, which means it passed all tests and used LESS resources than 99% of the submissions on the website.

SPOJ | AU7_2 - SERVERS

Permasalahan dan pendekatan :

AU7_2 - SERVERS

no tags

There are N ($1 \leq N \leq 100000$) servers and each has a fixed serving time T_i irrespective for any single task.

There are M ($1 \leq M \leq 1000000000$) tasks which needs to processed in sequential order one after another.

The tasks need not be processed immediately, if the servers are free. They can wait and assigned to a faster server if necessary.

Find a schedule such that the total time needed for processing all tasks is minimized.

Example: $N=2$, $M=6$ $T_1 = 7$ $T_2 = 10$

Optimal Schedule:

server1 processes task1 and server2 processes task2.

after 7 seconds, server1 is free and task3 is assigned to server1.

after 10 seconds, server2 is free and task4 is assigned to server2.

after 14 seconds, server1 is free and task5 is assigned to server1.

after 20 seconds, server2 is free and task6 is not assigned to server2. It waits for 1 second.

then after 21 seconds server1 is free and task6 is assigned to server1.

After 28 seconds all tasks are completed.

On other hand, if task6 was immediately assigned to server2 without waiting the total time would have been 30seconds.

Input:

The first line contains T ($T \leq 10$) number of testcases. Followed by description of each test case. An integer N number of servers and M number of tasks. The next N lines contain T_k for each server.

Output:

The minimum time(use longlong data type) for completing all tasks.

Sample Input:

```
1
2 6
7
10
```

Sample Output:

```
28
```

Soal ini digambarkan sebagai sebuah production line. Soal ini meminta untuk mencari berapa waktu tercepat yang diperlukan untuk memproduksi sebuah barang. Soal akan pertama menginputkan berapa testcase yang akan dilakukan dilambangkan dengan huruf T. kemudian soal akan memberikan berapa jumlah mesin (N) dengan batas 100.000 dan jumlah task yang harus dilakukan (M) dengan batas 1 miliar. Kemudian sebanyak n akan diberikan

waktu yang diperlukan dari tiap mesin untuk memproses task tersebut. Soal ini memiliki time limit 1 detik.

Abstraksi :

Jika kita menggunakan pendekatan brute force mencoba semua kemungkinan testcase dengan test case 1 miliar akan membutuhkan waktu yang sangat lama. Selain itu sebenarnya memungkinkan untuk mencari solusi dengan metode priority queue yang akan menghasilkan time complexity $O(M \log N)$ tetapi waktu tersebut masih tidak cukup karena nilai M bisa mencapai 1 miliar.

Oleh karena itu kita akan melakukan pendekatan **divide-and-conquer** yaitu khususnya **binary search** yang akan menghasilkan waktu $O(N \log M)$. Konsep utama binary search adalah

1. Selalu membagi menjadi 2 bagian
2. Tidak saling berinterseksi
3. Salah satu bagian akan dieksplorasi sampai mendapatkan solusi
4. Bilangan harus **monotone**, yaitu bilangan disusun secara seragam (contoh semua decreasing atau semua increasing).

Langkah – langkahnya adalah sebagai berikut

1. Mulailah dengan elemen tengah dari seluruh range sebagai kunci pencarian.
2. Jika nilai kunci pencarian sama dengan item, maka kembalikan indeks kunci pencarian.
3. Atau jika nilai kunci pencarian kurang dari item di tengah interval, persempit interval ke bagian bawah. Jika tidak, persempit ke bagian atas.
4. Periksa berulang kali dari poin kedua hingga nilainya ditemukan atau intervalnya kosong.

Contoh ilustrasi nya adalah sebagai berikut



<https://assets.digitalocean.com/articles/alligator/js/linear-vs-binary-search/binary-search.png>

Jika sample testcase diatas dicari waktu yang paling ideal akan menghasilkan ilustrasi sebagai berikut

waktu	mesin 1	mesin 2
1	task 1	task 2
2		
3		
4		
5		
6		
7		
8	task 3	task 4
9		
10		
11		
12		
13		
14		
15	task 5	task 4
16		
17		
18		
19		
20		
21		
22	task 6	
23		
24		
25		
26		
27		
28		

Untuk mencari waktu diatas akan meng Implementasi **binary search** yaitu kita akan menggunakan binary search untuk mencari waktu yang cukup untuk mesin melakukan iterasi yaitu 28. Range dari soal ini akan didapat dari mengkalikan waktu mesin pertama dikalikan dengan jumlah task yang harus dilakukan. Kita akan membahas algoritma dalam implementasi test case.

Implementasi Code

```
1  #include <stdio>
2  typedef long long LL;
3  #define MAXN 100002
4  int n,m;
5  int t[MAXN];
6  bool enough(LL time){
7      LL cnt = 0;
8      for (int i=0;i<n;++i){
9          LL here = time / t[i];
10         if(here >= m || here + cnt >= m)
11             return true;
12         cnt += here;
13     }
14     return false;
15 }
17 int main(){
18     int T;
19     scanf("%d", &T);
20     while(T--){
21         scanf("%d%d", &n, &m);
22         for(int i=0;i<n;++i)
23             scanf("%d", &t[i]);
24         LL low = 0, high = (LL)t[0]*m;
25         while (high - low > 1){
26             LL mid = (low + high)/2;
27             (enough(mid) ? high : low) = mid;
28         }
29         printf("%lld\n", high);
30     }
31     return 0;
32 }
33
```

Missal dengan testcase

Sample Input:

```
1
2 6
7
10
```

Sample Output:

Line 1-5 adalah library dan variable yang akan digunakan.

Line 19 pertama akan menginputkan nilai t yaitu jumlah queries

Line 20 akan mengulang sebanyak queries dalam while

Line 21 input jumlah mesin dan jumlah task

Line 22-23 akan menginput waktu yang dibutuhkan tiap mesin untuk memproses

Line 24 akan mengassign nilai low = 0 dan nilai high adalah nilai mesin pertama dikali jumlah task, artinya $7 \times 6 = 42$ akan menghasilkan ilustrasi sebagai berikut.

0	1	2	21	40	41	42
Low								High

Kemudian pada line 25-27 menggunakan while hingga range cuman ada 2 nilai, akan mencari nilai mid yaitu low + high dibagi 2, dalam testcase pertama berarti $0 + 42 / 2 = 21$. Sehingga menghasilkan range sebagai berikut.

0	1	2	21	40	41	42
Low				Mid				High

Line 27 akan mempassing nilai mid (21) pada fungsi enough. Jika return true nilai nilai high akan menjadi mid (pencarian akan ke kiri) jika false pencarian ke kanan.

Pada line 6-15 kita akan mengecek dengan passing nilai mid. Menggunakan perulangan for sebanyak jumlah mesin, variable here akan menyimpan nilai waktu dibagi waktu dibutuhkan tiap mesin. Jika penjumlahan waktu dibagi waktu tiap mesin = task yang ada maka nilai yang dicari sudah ditemukan. Jika tidak range pencarian akan di update sesuai ketentuan diatas.

Contoh iterasi fungsi enough

Mid = 21

Cnt = 0

Iterasi 1 here = $21 / 7 = 3$

$3 \geq 6$ || $3 + 0 \geq 6$

Cnt = 3

Iterasi 2 here = $21/10 = 2$

$2 \geq 6 \ || \ 2 + 3 \geq 6$

Maka return false, sehingga pencarian akan di geser kekanan (nilai low menjadi mid). Tiap ilustrasi akan menjadi ilustrasi seperti dibawah

Iterasi 1 :

21	31	42
Low		Mid		High

Iterasi 2 :

21	26	31
Low		Mid		High

Iterasi 3 :

26	28	31
Low		Mid		High

Iterasi 4 :

26	27	28
Low	Mid	High

Iterasi 5:

27	28
Low	High

High-Low ≤ 1 . Print nilai high yaitu 28.

Bukti accept

TimothyHosia_5025211098: submissions

SERVERS

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
30429255		2022-11-15 15:25:00	SERVERS	accepted edit ideone it	0.25	5.3M	CPP14
30426235		2022-11-15 07:35:54	SERVERS	accepted edit ideone it	0.15	5.3M	CPP14
30426163		2022-11-15 07:29:06	SERVERS	accepted edit ideone it	0.25	5.4M	CPP14