

Quiz PBO C – Soal 1

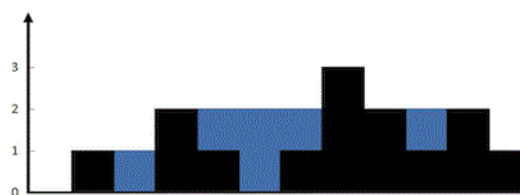
Brendan Timothy Mannuel

5025221177

Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1.

Compute how much water it can trap after raining.



Pada soal ini kita diperlukan mencari berapa banyak air (kotak) yang tertampung diantara kolom-kolom yang terbentuk dari kotak-kotak. Kita akan menggunakan implementasi Stack untuk mengerjakan soal ini.

class template

std::stack

<stack>

template <class T, class Container = deque<T> > class stack;

LIFO stack

Stacks are a type of container adaptor, specifically designed to operate in a LIFO context (last-in first-out), where elements are inserted and extracted only from one end of the container.

stacks are implemented as **container adaptors**, which are classes that use an encapsulated object of a specific container class as its **underlying container**, providing a specific set of member functions to access its elements. Elements are **pushed/popped** from the **"back"** of the specific container, which is known as the **top** of the stack.

The underlying container may be any of the standard container class templates or some other specifically designed container class. The container shall support the following operations:

empty
size
back
push_back
pop_back

The standard container classes [vector](#), [deque](#) and [list](#) fulfill these requirements. By default, if no container class is specified for a particular stack class instantiation, the standard container [deque](#) is used.

Template parameters

T

Type of the elements.
Aliased as member type stack::value_type.

Container

Type of the internal **underlying container** object where the elements are stored.
Its value_type shall be T.
Aliased as member type stack::container_type.

Setelah kita coba membaca soal lagi, kita dapat melihat agar air dapat terperangkap maka kita memerlukan 2 kolom yang terdapat minimal jarak 1 diantara 2 kolom tersebut, serta ketinggian dari air ditentukan oleh kolom yang lebih rendah. Kita akan menggunakan sample testcase 2 untuk menggambarannya. Kita akan menyimpannya di suatu array.

6

4 2 0 3 2 5

Untuk step pertama kita akan membuat suatu stack kosong. Kemudian kita akan memasukan index nya sebagai penunjuk ke tinggi agar nanti kita dapat lebih mudah mencari jarak diantara. Angka pertama akan langsung dimasukan karena stack masih kosong.

0					
4					

Kemudian untuk angka selanjutnya kita akan mengecek dengan parameter apakah stack tidak kosong dan ketinggian dari kolom di `stack.top()` itu lebih pendek dari ketinggian dari isi array selanjutnya. Ternyata masih belum memenuhi sehingga kita akan mengepush lagi.

0	1				
4	2				

Kemudian kita akan melakukan hal yang sama lagi dengan mengecek parameter, ternyata masih salah sehingga kita akan mengepush lagi.

0	1	2			
4	2	0			

Setelah itu kita akan mengecek lagi, dan ternyata ketinggian selanjutnya lebih tinggi maka kita akan menyimpan `stack.top()` sekarang dan kita akan mengepop nya.

Tinggiatas = 0

Index skrg = 3

0	1				
4	2				

Kemudian kita akan melihat apakah stack telah kosong, ternyata tidak. Selanjutnya kita akan mencari jarak diantara kedua kolom dengan mengurangi index nya

$\text{jarakantara} = i - \text{st.top()} - 1$

$3 - 1 - 1 = 1$ (jarak antara)

Kemudian kita akan mencari kolom yang lebih rendah diantara `st.top()` dan ketinggian dari index sekarang. Ternyata lebih rendah `st.top()` karena memiliki tinggi 2 sedangkan index sekarang memiliki ketinggian 3. Kemudian kita akan melakukan pengurangan dengan angka yang telah kita simpan (mengantisipasi bila didalam kolom tersebut ada balok yang dibawahnya) yaitu 0, $2 - 0 = 2$. Kemudian kita akan mengalikannya dengan jarak yaitu $2 \times 1 = 2$. Kemudian kita akan menyimpan hasil ini dan melanjutkan iterasi.

Ans = 2.

Kemudian kita akan melakukan pengecekan apakah `st.top()` lebih kecil daripada index sekarang ternyata benar sehingga kita akan melakukan iterasi lagi

Tinggiatas = 2

Index skrg = 3

0					
4					

Kemudian kita akan melihat apakah stack telah kosong, ternyata tidak. Selanjutnya kita akan mencari jarak diantara kedua kolom dengan mengurangi index nya

$\text{jarakantara} = i - \text{st.top()} - 1$

$3 - 0 - 1 = 2$ (jarak antara)

Kemudian kita akan mencari kolom yang lebih rendah diantara st.top() dan ketinggian dari index sekarang. Ternyata lebih rendah index sekarang karena memiliki tinggi 3 sedangkan st.top() memiliki ketinggian 4. Kemudian kita akan melakukan pengurangan dengan angka yang telah kita simpan (mengantisipasi bila didalam kolom tersebut ada balok yang dibawahnya) yaitu 0, $3 - 2 = 1$. Kemudian kita akan mengalikan nya dengan jarak yaitu $1 \times 2 = 2$. Kemudian kita akan menyimpan hasil ini dan melanjutkan iterasi.

$\text{Ans} = 2 + 2 = 4$.

Kita akan melakukan pengecekan untuk index selanjutnya. Ternyata lebih rendah dari stack.top() maka kita akan mengepush nya.

Kemudian kita akan melakukan pengecekan apakah st.top() lebih kecil daripada index sekarang ternyata salah sehingga kita akan mengepush index sekarang.

0	3				
4	3				

Kemudian untuk angka selanjutnya kita akan mengecek dengan parameter apakah stack tidak kosong dan ketinggian dari kolom di stack.top() itu lebih pendek dari ketinggian dari isi array selanjutnya. Ternyata masih belum memenuhi sehingga kita akan mengepush lagi.

0	3	4			
4	3	2			

Setelah itu kita akan mengecek lagi, dan ternyata ketinggian selanjutnya lebih tinggi maka kita akan menyimpan stack.top() sekarang dan kita akan mengepop nya.

Tinggiatas = 2

Index skrg = 5

0	3				
4	3				

Kemudian kita akan melihat apakah stack telah kosong, ternyata tidak. Selanjutnya kita akan mencari jarak diantara kedua kolom dengan mengurangi index nya

$\text{jarakantara} = i - \text{st.top()} - 1$

$5 - 3 - 1 = 1$ (jarak antara)

Kemudian kita akan mencari kolom yang lebih rendah diantara `st.top()` dan ketinggian dari index sekarang. Ternyata lebih rendah `st.top()` karena memiliki tinggi 3 sedangkan index sekarang memiliki ketinggian 5. Kemudian kita akan melakukan pengurangan dengan angka yang telah kita simpan (mengantisipasi bila didalam kolom tersebut ada balok yang dibawahnya) yaitu $0, 3 - 2 = 1$. Kemudian kita akan mengalikannya dengan jarak yaitu $1 \times 1 = 1$. Kemudian kita akan menyimpan hasil ini dan melanjutkan iterasi.

$$\text{Ans} = 4 + 1 = 5.$$

Kemudian kita akan melakukan pengecekan apakah `st.top()` lebih kecil daripada index sekarang ternyata benar sehingga kita akan melakukan iterasi lagi

$$\text{Tinggiatas} = 3$$

$$\text{Index skrg} = 5$$

0					
4					

Kemudian kita akan melihat apakah stack telah kosong, ternyata tidak. Selanjutnya kita akan mencari jarak diantara kedua kolom dengan mengurangi index nya

$$\text{jarakantara} = i - \text{st.top()} - 1$$

$$5 - 0 - 1 = 4 \text{ (jarak antara)}$$

Kemudian kita akan mencari kolom yang lebih rendah diantara `st.top()` dan ketinggian dari index sekarang. Ternyata lebih rendah `st.top()` karena memiliki tinggi 4 sedangkan index sekarang memiliki ketinggian 5. Kemudian kita akan melakukan pengurangan dengan angka yang telah kita simpan (mengantisipasi bila didalam kolom tersebut ada balok yang dibawahnya) yaitu $0, 4 - 3 = 1$. Kemudian kita akan mengalikannya dengan jarak yaitu $1 \times 4 = 4$. Kemudian kita akan menyimpan hasil ini dan melanjutkan iterasi.

$$\text{Ans} = 5 + 4 = 9.$$

Kemudian kita akan melakukan pengecekan apakah `st.top()` lebih kecil daripada index sekarang ternyata benar sehingga kita akan melakukan iterasi lagi

$$\text{Tinggiatas} = 4$$

$$\text{Index skrg} = 5$$

Kemudian kita akan melihat apakah stack telah kosong, ternyata iya sehingga kita akan melakukan pemutusan pada loop dan mengepush index sekarang

5					
5					

Karena `i` sudah memiliki nilai yang sama dengan `m` maka kita tinggal melakukan output `ans` yaitu 9.

```

1  #include <stdio.h>
2  #include <stack>
3  using namespace std;
4
5  int main()
6  {
7      int n;
8      stack <int> st;
9      scanf("%d", &n);
10     int array[n];
11     for(int i = 0; i < n; i++){
12         scanf("%d", &array[i]);
13     }
14     int m = sizeof(array) / sizeof(array[0]);
15
16     int ans = 0;
17
18     for(int i = 0; i < m; i++){
19         while(!st.empty() && (array[st.top()] < array[i])){
20
21             int tinggiatas = array[st.top()];
22             st.pop();
23
24             if (st.empty()) break;
25
26             int jarakantara = i - st.top() - 1;
27
28             int tinggi = min(array[st.top()], array[i]) - tinggiatas;
29
30             ans += tinggi * jarakantara;
31         }
32         st.push(i);
33     }
34
35     printf("%d", ans);
36
37     return 0;
38 }
39

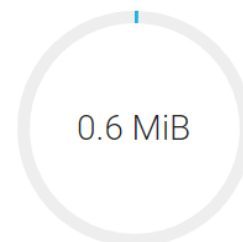
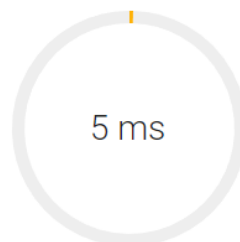
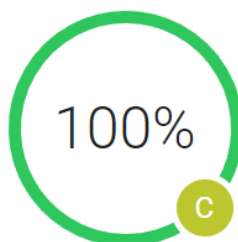
```

Problem
[Trapping Rain Water](#)

Submitted
1 hour ago

Programming Language
C++ 20 (gnu 10.2)

Author
[Brendan_C177](#)



Your submission was graded C, which means it passed all tests and used LESS resources than 50% of the submissions on the website.

Quiz PBO C – Soal 2

Brendan Timothy Mannuel

5025221177

Arrange

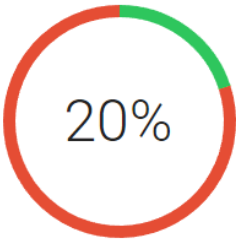
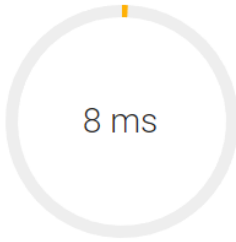
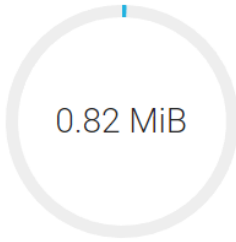
There are m students in a classroom, out of which n of them are girls. The teacher, Mr X wants to arrange all the students in a line. X thinks that his class girls are very talkative. So he doesn't want any two girls to be together. Mr X wants to know the number of ways he can arrange these m students. Help him out.

Pada soal ini kita diperlukan mencari berapa banyak kombinasi yang dapat dibentuk dari suatu urutan tempat duduk antara cewek dan cowok dimana tidak boleh ada 2 cewek yang duduk bersebelahan. Untuk menyelesaikan permasalahan ini kita dapat menggunakan rumus kombinasi pada umumnya, kita misalkan jumlah laki laki adalah variable L, yaitu untuk laki laki maka banyak cara untuk laki laki adalah factorial dari jumlah laki laki (L!) sedangkan untuk Perempuan dapat duduk sebanyak laki-laki + 1 faktorial (L+1!) dengan kombinasi sebanyak ${}^{L+1}P_n$. Di soal juga dijelaskan bahwa kita harus memodulo jawaban dengan $10^9 + 7$.

```
1  #include <iostream>
2  using namespace std;
3  const long long int MOD = 1000000007;
4
5  long long int faktorial(long long int n) {
6      long long int ans = 1;
7      for(int i=1; i<=n; i++){
8          ans = ans*i;
9      }
10
11     return ans % MOD;
12 }
13
14 int main() {
15     int itr;
16     scanf("%d", &itr);
17
18     while (itr--) {
19         int m, n;
20         scanf("%d %d", &m, &n);
21
22         int l = m - n;
23
24         long long int hasil = ((faktorial(l) * faktorial (l + 1)) / (faktorial (l-n+1))) % MOD;
25
26         printf("%lld\n", hasil);
27     }
28     return 0;
29 }
```

Ternyata setelah kita coba untuk submit jawaban, ternyata masih Run Time Error

Problem	Submitted	Programming Language	Author
Arrange	7 days ago	C++ 20 (gnu 10.2)	Brendan_C177

Test #	Status	Score	Duration	CPU	Memory
❗ Test suite #1	Runtime error	20 / 100	8 ms	7 ms	840 KiB
✅ Test #1	Accepted	20 / 20	1 ms	1 ms	840 KiB
❗ Test #2	Runtime error	0 / 20	7 ms	7 ms	772 KiB
❗ Test #3	Runtime error	0 / 20	8 ms	7 ms	768 KiB
❗ Test #4	Runtime error	0 / 20	7 ms	7 ms	772 KiB
❗ Test #5	Runtime error	0 / 20	8 ms	7 ms	776 KiB
		20 / 100	8 ms	7 ms	840 KiB

Setelah kita coba untuk mencari tahu kesalahan, terdapat beberapa kemungkinan kesalahan yang bisa terjadi yaitu operasi factorial memakan waktu yang terlalu lama, serta ada kondisi pada percabangan yang kurang benar.

Oleh karena itu kita dapat mencoba mengubah dengan cara menyimpan operasi factorial sekali saja saat kita menjalankan program sehingga akan mempercepat perhitungan.

```

24 factorial[0] = 1;
25 for (long long i=1; i<1000009; i++) {
26     factorial[i] = (factorial[i-1] * i) % MOD;
27 }
```

Selain itu kita juga akan menambahkan 1 kasus percabangan Ketika jumlah laki – berjumlah lebih sedikit dan memiliki selisih lebih dari 2 dimana pada kasus ini dipastikan ada 2 cewek yang akan selalu duduk bersebelahan sehingga kita dapat langsung mengasumsikan jawaban = 0

```

37
38 if (l+1 < n) {
39     printf("0\n");
40     continue;
41 }
```

Untuk mempercepat perhitungan factorial kita dapat menggunakan Fermat Little Theorem untuk mempercepat mencari kombinasi dengan memanfaatkan Modular Multiplicative Inverse, dan juga menghilangkan operasi pembagian yang juga memakan banyak cost

```

5 long long fast_pow(long long base, long long n, long long M) {
6     if(n==0)
7         return 1;
8     if(n==1)
9         return base;
10    long long halfn=fast_pow(base,n/2,M);
11    if(n%2==0)
12        return ( halfn * halfn ) % M;
13    else
14        return ( ( halfn * halfn ) % M ) * base ) % M;
15 }
16
17 long long findMMI_fermat(long long n,long long M) {
18     return fast_pow(n,M-2,M);
19 }

45 long long y = findMMI_fermat(factorial[l + 1 - n], MOD);
46 y = (factorial[l + 1] * y) % MOD;
47
48 long long hasil = x * y % MOD;
49
50 printf("%lld\n", hasil);
51 }

```

Dengan memperbaiki tiga hal diatas maka kita dapat lebih mengoptimisasi kode sehingga tidak terjadi RTE.

Problem	Submitted	Programming Language	Author
Arrange	2 hours ago	C++ 20 (gnu 10.2)	Brendan_C177

100%
A+

66 ms

8.45 MiB

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

Test #	Status	Score	Duration	CPU	Memory
✓ Test suite #1	Accepted	100 / 100	66 ms	65 ms	8,652 KiB
✓ Test #1	Accepted	20 / 20	6 ms	6 ms	8,644 KiB
✓ Test #2	Accepted	20 / 20	27 ms	27 ms	8,652 KiB
✓ Test #3	Accepted	20 / 20	66 ms	65 ms	8,648 KiB
✓ Test #4	Accepted	20 / 20	39 ms	39 ms	8,648 KiB
✓ Test #5	Accepted	20 / 20	66 ms	65 ms	8,648 KiB
		100 / 100	66 ms	65 ms	8,652 KiB

Tugas PBO C – Resume

Brendan Timothy Mannuel

5025221177

Pada pertemuan kali ini kita membahas tentang bagaimana cara menyelesaikan suatu soal dengan focus pada object oriented programming. Berbeda dengan structure oriented dimana kita harus memikirkan tentang fungsi yang harus digunakan, pada object oriented kita harus mencari tahu dulu class. Tetapi tetap terdapat persamaan seperti keduanya kita harus melakukan abstraksi.

Abstraksi adalah cara kita menyelesaikan suatu masalah dengan membangkitkan intuisi kita. Kita dapat melakukan nya karena sebenarnya di hidup kita terdapat banyak real world problems yang pastinya memerlukan solusi sehingga kita harus melakukan mapping masalah tersebut dan menggunakan berbagai macam teori yang sudah ada untuk menyelesaikan masalah tersebut, bagaimana kita bisa tahu mana teori yang terbaik pastinya dengan sering membaca dan juga sering menyelesaikan masalah.

Kita akan coba untuk menyelesaikan masalah berikut ini

LOTTERY - Tickets lottery

Byteland organizes this year's Soccer World Cup. Because of the mismanagement of the organizing team almost all tickets are sold out. But one radio station still has some tickets which will be raffled. Specifically, the radio station has announced a telephone game, where participants can choose a number between 1 and 1.000.000.000, and after each day the person who has chosen the k^{th} smallest number will win one ticket. A special rule is in place which disallows people to choose a number which was already chosen by someone else (in this case the person is asked to choose another number).

Martin, a fanatic soccer fan without tickets, bribed Robert H., an employee of the radio station, by promising small gifts for telling him a current winning number: "A fine basket with specialities from the black forest, including some really good sausages, ham and - hold on to your seat - a wonderful KuKuClock! And a beer mug, too! Do I leave you any choice???"

Now Robert is in trouble and asks you if you can write a program which will tell him the k^{th} smallest number at any time of the game.

Setelah kita membaca soal kita tahu bahwa objektif kita adalah mencari angka k^{th} terkecil dari suatu Kumpulan no telepon yang diberi. Kita bisa saja dengan mudah melakukan sorting pada setiap saat kita melakukan input dengan melakukan bubble sort atau insertion sort, tetapi waktu yang akan dipakai sangat besar karena time complexity nya adalah $O(n)$. Kita perlu mencari cara agar bisa sorting dan menggunakan waktu yg paling sedikit, oleh karena itu kita bisa menggunakan Priority Queue yang memiliki kemampuan untuk melakukan sorting serta memiliki time complexity $O(\log n)$. Implementasi Dibawah

```

int main(){
    int testcase, phoneCalls, number, result, kth;
    testcase = getNum();
    for (int i = 0; i < testcase; i++){
        priority_queue<int> numList;
        phoneCalls = getNum();
        kth = getNum();
        for (int j = 0; j < phoneCalls; j++){
            number = getNum();
            if (number == 0){
                if (kth > numList.size()) result = -1;
                else result = numList.top();
                printf("%d\n", result);
            }
            else {
                if (numList.size() < kth) numList.push(number);
                else if (numList.size() == kth && number < numList.top()){
                    numList.pop();
                    numList.push(number);
                }
            }
        }
    }
    return 0;
}

```

Kita akan memasukan angka kedalam suatu priority queue dan akan memanfaatkan angka yang diminta oleh soal sebagai suatu batas. Ketika angka yang diminta oleh soal adalah ke 2 terkecil maka kita hanya akan menyimpan 2 angka didalam priority queue, karena sifat dari priority queue yaitu descending sehingga kita hanya perlu untuk menyimpan sebanyak k angka terkecil saja dan kita hanya perlu mengoutput top nya. Selanjutnya agar lebih cepat kita menggunakan fungsi Getnum agar lebih cepat dalam masalah IO

```

6 int getNum(){
7     int res = 0;
8     char c;
9     int b = 0;
10    while (1){
11        c = getchar_unlocked();
12        if (c == '-') b=1;
13        if (c == ' ' || c == '\n') continue;
14        else break;
15    }
16    if (c != '-') res = c - '0';
17    while(1){
18        c = getchar_unlocked();
19        if (c >= '0' && c <= '9') res = 10 * res + c - '0';
20        else break;
21    }
22    if (b == 1) res*= -1;
23    return res;
24 }

```

11966109		2023-10-06 05:26:30	Tickets lottery	accepted edit delete it	0.05	5.4M	CPP14
----------	---	------------------------	-----------------	--	------	------	-------