Tugas PBO C – Resume

Brendan Timothy Mannuel

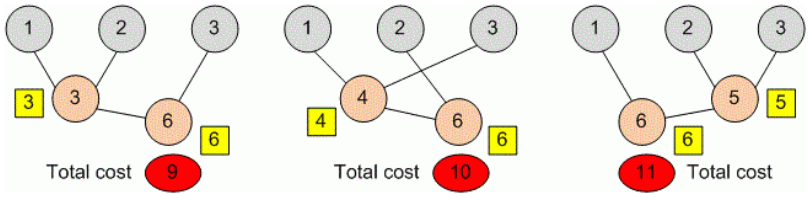
5025221177

Add All

The cost of adding two numbers equals to their sum. For example to add **1** and **10** costs **11**. The cost of addition **1** and **2** is **3**. We can add numbers in several ways:

- **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**

We hope you understood the task. You must add all numbers so that the total cost of summation will be the smallest.



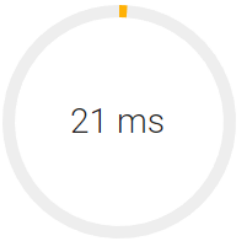
Pada soal ini kita akan mencari total cost terkecil yang bisa dibuat ketika kita menambahkan semua angka yang diberikan. Setelah kita mencoba untuk melihat soal nya Kembali dapat kita lihat bahwa sebenarnya agar hasil cost nye menjadi paling kecil maka kita perlu untuk menambahkan 2 angka terkecil terlebih dahulu agar kita bisa menghemat cost nya. Oleh karena itu kita memerlukan suatu cara sorting yang efisien. Kita dapat menggunkana Priority Queue yang bisa melakukan sorting, tetapi karena kita memerlukan nya dari angka yang paling kecil terlebih dahulu maka kita dapat menggantinya dengan menjadikannya memili sifat Ascending

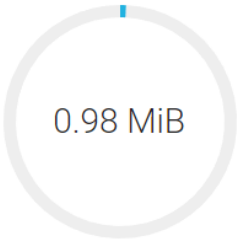
```
priority_queue<long, vector<long>, greater<long> > pq;
```

Setelah itu kita akan memasukan semua angka yang ingin dimasukan cost nya, kemudian kita akan mengambil 2 angka paling atas karena pastinya dua angka tersebut adalah 2 angka yang terkecil sehingga kita dapat menambahkannya dan memasukannya Kembali kedalam priority queue, kita akan terus mengulangi hal ini sampai isi priority queue hanya tersisa 1, kemudain kita akan mengoutputnya.

```
while (pq.size() > 1) {
    ll a = pq.top();
    pq.pop();
    ll b = pq.top();
    pq.pop();
    pq.push(a + b);
    res += (a + b);
}
```

Problem	Submitted	Programming Language	Author
Add All	yesterday	C++ 20 (gnu 10.2)	Brendan_C177





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

```cpp
#include <stdio.h>
#include <queue>
#include <functional>
using namespace std;
#define gc getchar_unlocked
#define ll long long

template <typename T> void getNum(T &val){
    char ch; bool bo=0; val=0;
    for(ch=gc(); ch<'0'||'9'<ch;ch=gc()) if(ch=='-') bo =1;
    for(;'0'<=ch && ch<='9'; val=(val<<3)+(val<<1)+ch-48, ch=gc());
    if(bo) val=-val;
}

priority_queue<long, vector<long>, greater<long> > pq;

int main() {
    ll testcase, number, res, i;

    getNum<ll>(testcase);

    for (res = i =0; i < testcase; i++) {
        getNum<ll>(number);
        pq.push(number);
    }

    while (pq.size() > 1) {
        ll a = pq.top();
        pq.pop();
        ll b = pq.top();
        pq.pop();
        pq.push(a + b);
        res += (a + b);
    }

    printf("%lld\n", res);
}
```

Tugas PBO C – Soal 1

Brendan Timothy Mannuel

5025221177

Pick the Candies

> Many children went to a sweet shop. There were n candy varieties and each variety is kept in a separate bowl. The sweetness of each variety is written on the bowl. All the children wanted the candy with highest sweetness value. As there are only limited candies in each variety, the shop keeper makes a rule. According to the rule, the shopkeeper will show selectively chosen k varieties to every children. The children can pick any one of those varieties and move away. To make it easy for him, the shop keeper shows the
>
> varieties 1,2,...,k to children1,
>
> varieties 2,3,...,k+1 to children2,
>
> varieties 3,4,...,k+2 to children3 and so on..
>
> All the children are good at math. Find what variety each child will choose.

Pada soal ini kita disuruh mencari angka terbesar dari setiap pilihan yang diberikan oleh pemilik toko. Banyak pilihan yang diberikan ditentukan sebanyak K dengan kombinasi yang selalu bergeser sebanyak 1. Kita dapat menggambarkan persoalan ini seperti sebuah “window” dimana kita harus dapat memilih angka terbesar dari setiap window sebesar K. Kita dapat menggunakan double ended queue atau biasa disebut deque karena sifat nya yang bisa diakses dari front atau back

std::deque

Defined in header <deque>

```
template<
    class T,
    class Allocator = std::allocator<T>
> class deque;                                                    (1)

namespace pmr {
    template< class T >
    using deque = std::deque<T, std::pmr::polymorphic_allocator<T>>;   (2)  (since C++17)
}
```

`std::deque` (double-ended queue) is an indexed sequence container that allows fast insertion and deletion at both its beginning and its end. In addition, insertion and deletion at either end of a deque never invalidates pointers or references to the rest of the elements.

As opposed to `std::vector`, the elements of a deque are not stored contiguously: typical implementations use a sequence of individually allocated fixed-size arrays, with additional bookkeeping, which means indexed access to deque must perform two pointer dereferences, compared to vector's indexed access which performs only one.

The storage of a deque is automatically expanded and contracted as needed. Expansion of a deque is cheaper than the expansion of a `std::vector` because it does not involve copying of the existing elements to a new memory location. On the other hand, deques typically have large minimal memory cost; a deque holding just one element has to allocate its full internal array (e.g. 8 times the object size on 64-bit libstdc++; 16 times the object size or 4096 bytes, whichever is larger, on 64-bit libc++).

The complexity (efficiency) of common operations on deques is as follows:

- Random access - constant *O(1)*.
- Insertion or removal of elements at the end or beginning - constant *O(1)*.
- Insertion or removal of elements - linear *O(n)*.

`std::deque` meets the requirements of *Container*, *AllocatorAwareContainer*, *SequenceContainer* and *ReversibleContainer*.

Kita akan memanipulasi deque tersebut dengan cara setiap saat kita menginputkan angka maka kita akan terus melakukan pengecheckan.

```
for (int l = 0; l < n; l++)
{
    while (!dq.empty() && dq.front() < l - k + 1)
        dq.pop_front();

    while (!dq.empty() && arr[dq.back()] < arr[l])
        dq.pop_back();

    dq.push_back(l);

    if (l >= k - 1)
        printf("%d ", arr[dq.front()]);
}
```

1. Kita akan mengechek apakah isi deque tidak kosong serta kita kan mengecheck apakah sekarang besar window sudah lebih besar dari K, jika sudah maka kita akan mengepop front dari deque.
2. Kita akan mengechek apakah isi deque tidak kosong serta kita akan mengecheck apakah nilai dari angka paling belakang di dalam window kita masih lebih besar, jika ternyata lebih kecil maka kita akan melakukan pop back, hal ini dilakukan untuk memastikan angka terdepan di dalam window pasti menjadi yang paling besar tetapi juga masih menyimpan angka sebanyak K.
3. Kemudian kita akan mengepush indeks dari array yang mengandung angka kedalam window
4. Kemudian kita akan mengecheck apakah sekarang telah sebesar K, jika iya kita akan mengoutput angka terdepan didalam window karena angka tersebut angka yang paling besar.

Kita akan mencoba menguunakan test case

```
5 3
1 2 3 4 5
```

Kita akan membuat window sebesar 3, dan memsaukan 5 kali inputan

Input “1”

1		

Input “2”

2		

1 di pop karena 2 lebih besar

Input “3”

3		

2 di pop karena 3 lebih besar, kemudian kita akan mengouput 3 karena index sekarang telah sama dengan K

Input “4”

4		

3 di pop karena 4 lebih besar, kemudian kita akan mengouput 4 karena index sekarang telah lebih besar dari K

Input “5”

5		

4 di pop karena 5 lebih besar, kemudian kita akan mengouput 5 karena index sekarang telah lebih besar dari K

Sehingga hasil output adalah **3 4 5**

```cpp
#include <stdio.h>
#include <queue>
#include <functional>
using namespace std;
#define gc getchar_unlocked
#define ll long long

template <typename T> void getNum(T &val){
    char ch; bool bo=0; val=0;
    for(ch=gc(); ch<'0'||'9'<ch;ch=gc()) if(ch=='-') bo =1;
    for(;'0'<=ch && ch<='9'; val=(val<<3)+(val<<1)+ch-48, ch=gc());
    if(bo) val=-val;
}

int main(){
    int t, n, k;

    getNum<int>(t);
    int y = 0;
    for(int i = 0; i < t; i++){
        getNum<int>(n);
        getNum<int>(k);
        deque <int> dq;
        int arr[n];

        for (int j = 0; j < n; j++){
            int x;
            getNum<int>(x);
            arr[j] = x;
        }

        for (int l = 0; l < n; l++)
        {
            while (!dq.empty() && dq.front() < l - k + 1)
                dq.pop_front();

            while (!dq.empty() && arr[dq.back()] < arr[l])
                dq.pop_back();

            dq.push_back(l);

            if (l >= k - 1)
                printf("%d ", arr[dq.front()]);
        }
        printf("\n");
    }
}
```

31972470		2023-10-07 07:00:06	Pick the candies	accepted edit ideone it	0.40	5.4M	CPP14

Tugas PBO C – Soal 2

Brendan Timothy Mannuel

5025221177

Appreciating Guards in ISM 375 Points

Guards in ISM work really hard to provide the top class security to students ;-) ISM Administration also knows this. So they decided to give some money to the guards. They gave some amount of money to all the guards, but some guards became really angry after this since they got less money than others. So Administration thought to distribute the money to the guards such that each guard gets equal amount of money. They used a particular algorithm to do this. At every step, if guard A has maximum amount of money (say "X") and guard B has minimum amount of money (say "Y"), they transfer LIF((X-Y)/2) amount of money from A to B. LIF(x) returns the least integer that is not less than x. They repeat this until everyone gets equal amount of money. Can you tell ISM Administration in how many steps they will be able to distribute this money equally? Print "-1" without quotes if it is not possible to distribute money equally by this algorithm.

Pada soal ini kita diperlukan untuk mencari cara agar bisa membagikan gaji kepada N petugas secara rata dengan cara mengambil gaji seseorang dan memberikannya kepada seseorang, kita harus mengoutput berapa banyak langkah yang diperlukan jika kita menggunakna rumus LIF(x-y/2) diman x adalah jumlah gaji terbanyak dan y adalah jumlah gaji tersedikit. Kita akan menggunakan implementasi Priority Queue Kembali karena kita memerlukan cara mencari angka terkecil serta angka terbesar, kita dapat menggunakan 2 priority queue dimana priority queue pertama bersifat descending sedangkan yang kedua bersifat ascending.

```
priority_queue<int> pq1;
priority_queue<int, vector<int>, greater<int> > pq2;
```

Untuk mengetahui apakah nanti semua dari petugas telah memiliki gaji yang sama, maka kita dapat mencari rata rata nya terlebih dahulu, dan sekaligus kita akan mengechek jika total dari semua angka ketika dimodulo dengan total petugas tidak 0 maka dapat dihasilkan hasil yang seimbang.

```
for(int j = 0; j < n; j++){
    int number;
    getNum<int>(number);
    total += number;
    temp[j] = number;
}

if (total % n == 0){
    average = total/n;
} else {
    printf("-1\n");
    continue;
}
```

Setelah itu kita akan mengepush semua angka tadi mengikuti aturan, jika dibawah average maka kita kan memasukan kedalam priority queue yang bersifat ascending dan sebaliknya jika diatas average maka akan dimasukan kedalam priority queue yang bersifat descending. Kita melakukan hal ini agar kita dapat lebih mudah mengakses angka terbesar dan angka terkecil nya.

```
for(int j = 0; j < n; j++){
    if (temp[j] < average){
        pq2.push(temp[j]);
    } else if (temp[j] > average){
        pq1.push(temp[j]);
    }
}
```

Kemudian akan mengambil top dari kedua priority queue karena pastinya kedua angka tersebut adalah yang terbesar dan yang terkecil, kemudian kita akan memasukannya kedalam rumus yaitu Z = LIF((X-Y) / 2) dimana X adalah gaji terbesar dan Y adalah gaji terkecil. Kemudian X akan dikurangi dengan Z sedangkan Y akan ditambah dengan Z agar seimbang, kemudian kita akan melakukan pengecekan lagi seperti jika dibawah average maka kita kan memasukan kedalam priority queue yang bersifat ascending dan sebaliknya jika diatas average maka akan dimasukan kedalam priority queue yang bersifat descending. Kita akan melakukan loop sampai salah satu dari priority queue telah habis sekaligus kita akan menyimpan jumlah iterasi dan akan diotput sebagai jawaban.

```
int itr = 0;
while (!pq1.empty() && !pq2.empty()){
x = pq1.top();
pq1.pop();
y = pq2.top();
pq2.pop();
z = ceil((x - y) / 2);
x -= z;
if (x < average) pq2.push(x);
else if (x > average) pq1.push(x);
y += z;
if (y < average) pq2.push(y);
else if (y > average) pq1.push(y);
itr++;
}
printf("%d\n", itr);
```

31979095	■	2023-10-08 10:21:49	Appreciating Guards in ISM 375 Points	**accepted** edit Ideone it	0.10	5.4M	CPP14