Johannes Daniswara Pratama

5025221276

Lapoarn Soal Eolym “ Add All “

Permasalahan :

“The Cost Of Adding Two Numbers Equal to their sum “ . Jadi kita akan diberikan n angka yang mana harus dijumlahkan semuanya. Hasil dari penjumlahan angka dari dua angka akan dijumlahkan lagi dengan angka selanjutnya sampai habis.

Input :

- Baris pertama berisi positif integer n ($2 \leq \ n \leq \ 10^5$)
- Baris selanjutnya berisi n angka , masing masing tidak lebih dari 10^5

Output :

Outputkan minimum total cost of summation

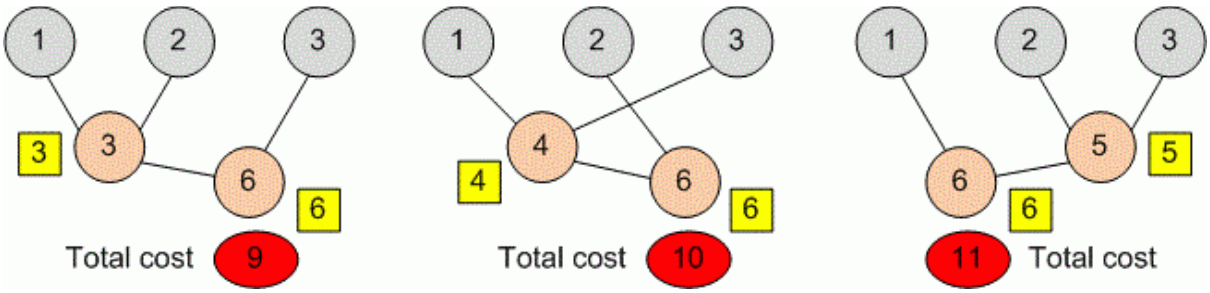
Penjelasan masalah :

Jadi akan diberikan integer sebanyak n buah . Dari kumpulan integer ini, kita harus menjumlahkan semuanya termasuk hasil jumlah agar hanya tersisa satu angka. Misal ada 3 angka, maka kita akan menjumlahkan angka pertama dan angka kedua. Lalu hasil penjumlahan tersebut ditambah dengan angka ketiga. Dan outputkan hasil total cost tersebut.

Penjelasan testcase :

```
3
1 2 3
```

Ada beberapa cara untuk menambahkan ketiga angka tersebut yaitu :



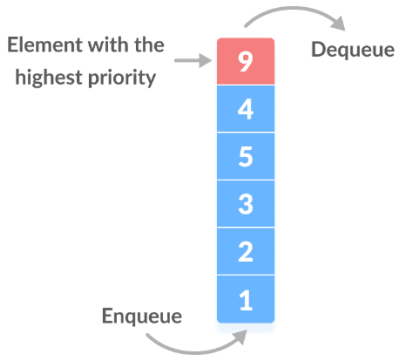
Namun , kita akan memilih yang pertama.

- 1 + 2 = 3 (cost = 3)
- 3 + 3 = 6 (cost = 6)
- 3 + 6 = 9

Jadi hasil dari penjumalahan angka, akan dijumlahkan dengan angka sebelumnya. Dan cost ini akan dijumlahkan juga sehingga hanya tersisa 1 angka

Kita dapat menggunakan STL class Priority Queue untuk menyelesaikan permasalahan ini .

Priority queue adalah jenis struktur data yang menyimpan elemen berdasarkan urutan prioritas nya (bisa secara ascending dan descending). Sifat dari priority queue adalah FIFO (First In First Out)



Priority Queue STL Method

Method	Definition
empty()	Returns whether the queue is empty.
size()	Returns the size of the queue.
top()	Returns a reference to the topmost element of the queue.
push()	Adds the element to the queue.
pop()	Deletes the first element of the queue.

Perancangan dan Analisis Algoritma :

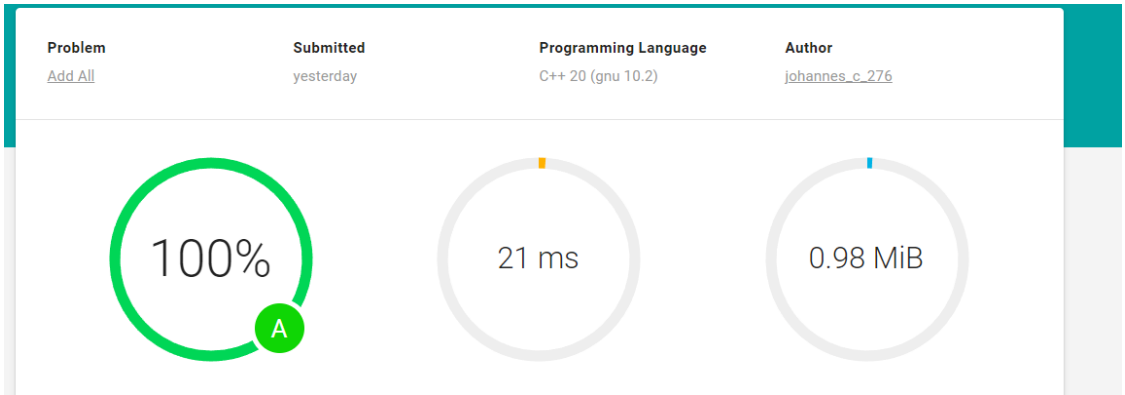
1) Inisialisasi variable long long n, num, res , i
2) Inisialisasi priority queue dengan greater<> untuk ascending order
3) Lakukan perulangan sebanyak n kali , ambil input num dan push ke prioriy queue
4) Selama size dari priority queue > 1 :
   - Ambil top dari queue , dan masukan ke ll a . pop elemen
   - Ambil top lagi dari queue, dan masukan ke ll b. pop elemen
   - Push a + b kedalam priority queue
   - Tambahkan res dengan a + b
5) Kemudian print res

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

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

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

int main() {
    ll n, num, res, i;
    getnum<ll>(n);

    for (res = i = 0; i < n; i++) {
        getnum<ll>(num), pq.push(num);
    }
    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);
    return 0;
}
```

Time complexity : O (n log n)

Bukti verdict AC : https://www.eolymp.com/en/submissions/14587191



Laporan Soal SPOJ " ACMCEG2C - Pick the candies "

Permasalahan :

Terdapat n variant permen yang disimpan dalam piring terpisah. Tingkat kemanisan tiap variant terdapat di piring. Anak anak yang pergi ke toko permen mau permen dengan tingkat kemanisan yang paling tinggi. Karena stok permen ini terbatas, maka penjual toko akan menujukan k pilihan variant permen kepada anak anak. Anak-anak dapat memilih salah satu dan harus pergi.

Aturan:

Variant 1,2,…, k untuk anak 1

Variant 2,3,…,k+1 untuk anak 2

Variant 3,4,..,k+2 untuk anak 3 dan seterusnya

Input

- Baris pertama adalah integer t $(1 \leq t \leq 1000)$, yaitu testcase . Tiap testcase memiliki input sebanyak 2 baris
- Baris selanjutnya adalah dua integer , yaitu n $(1 \leq n \leq 1000)$ dan k $(1 \leq k \leq n)$
- Baris selanjutnya adalah integer sejumlah n yang berisi tingkat kemanisan $((0 \leq kemanisan \leq 10000)$

Output

Untuk setiap test case, print variant permen yg dipilih oleh anak – anak

Penjelasan Masalah :

Jadi akan diberikan sejumlah n variant permen dan k . k ini merupakan banyaknya pilihan permen yg akan ditunjukan oleh penjual kepada anak anak. Masing masing n permen ini memiliki tingkat kemanisan yg berbeda. Kita disuruh untuk memilih tingkat kemanisan yg paling tinggi diantara k pilihan yg ditunjukan oleh penjual.

Tiap anak hanya boleh memilih salah satu permen. Jika sudah memilih, penjual akan menunjukan variant mulai dari variant k+1 untuk anak ke 2, k+2 untuk anak 3 , dan seterusnya. Dari pola ini kita dapat mengetahui jumlah anak yg dapat memilih yaitu

Jumlah anak yg dapat memilih : $n - k + 1$

Penjelasan testcase 1:

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

Pilihan anak 1 : 1 , 2 , 3 maka yg dipilih : 3

Pilihan anak 2 : 2 , 3 , 4 maka yg dipilih : 4

Pilihan anak 3 : 3, 4, 5 maka dipilih : 5

Penjelasan testcase 2 :

4 2

7 1 6 8

Pilihan anak 1 : 7,1 maka yg dipilih : 7

Pilihan anak 2 : 1,6 maka yg dipilih : 6

Pilihan anak 3 : 6,8 maka dipilih : 8

Penjelasan testcase 3 :

9 5

7 14 3 0 2 2 2 2 2

Pilihan anak 1 : 7,14,3,0,2 maka yg dipilih adalah 14

Pilihan anak 2 : 14,3,0,2,2 maka yg dipilih adalah 14

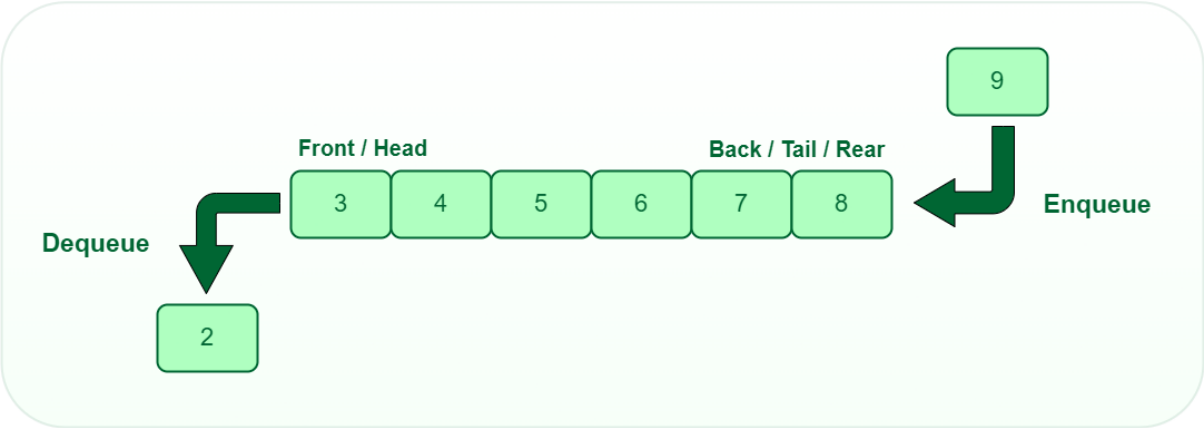
Pilihan anak 3: 3,0,2,2,2 maka yg dipilih adalah 3

Pilihan anak 4 : 0,2,2,2,2 maka yg dipilih adalah 2

Pilihan anak 5 : 2,2,2,2,2 maka yg dipilih adalah 2

Perancangan dan Analisis Algoritma

Kita dapat menggunakan STL class deque untuk menyelesaikan problem ini. Mengapa deque ? Karena merupakan struktur data yg bersifat FIFO (*First In First Out*) dan dapat dimasukan dari top maupun back .



Method yg dimiliki deque :

1) front : accesses the first element
2) back : access the last element

3) Empty : check whether the container of stack is empty
4) Size : return the number of element
5) Push_back : insert element to the end
6) Push_front : insert the element to the begginning
7) Pop_back : remove last element
8) Pop_front: remove the first element

Algoritma #1 – Deque with Max Element Range Query

Langkah algoritma :

1. Inisialisasi variable cases,n,k,x . Ambil input cases
2. Untuk setiap cases :
   - Ambil input n dan k
   - Inisialisasi deque<int> q;

   Untuk setiap n :

   - Ambil input dari x , dan push_back ke deque
3. Setelah semua diinput, lakukan perulangan dari o sampai $n - k + 1$. Operasi ini untuk menghitung range maximum number yg ada dari iterator deque front dan front+k .
4. Pop_front deque . lalu print new line

```
#include <stdio.h>
#include <queue>
#include <algorithm>
#define gc getchar_unlocked
using namespace std;

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

int main() {
    int cases;
    getnum(cases);
    int n, k,x;

    while (cases--) {
```

```
        getnum<int>(n);
        getnum<int>(k);
        deque<int> v;

        for (int i = 0; i < n; ++i) {
        getnum<int>(x), v.push_back(x);
        }

        for (int i=0; i<n-k+1; i++) {
             printf("%d ", *max_element(v.begin(),
v.begin()+k));
             v.pop_front();
        }
        printf("\n");
    }

    return 0;
}
```

Time complexity : O((n - k + 1) * k)

Bukti verdict AC:

31975201	2023-10-07 15:34:41	Johannes_C_276	accepted edit ideone it	0.63	5.4M	CPP14-CLANG

Time: 0.63s

Namun , algoritma ini masih bisa dioptimisasi dengan mencari nilai maksimum pilihan permen langsung dari deque tersebut tanpa menggunakan *max element.

Algoritma 2

Langkah :

1) Inisialisasi variable t,n,k,m . Ambil input untuk testcase
2) Untuk setiap testCase:
    - Ambil input untuk n dan k
    - Inisialisasi array sebanyak n
    - Lakukan perulangan sebanyak n untuk mengambil tingkat kemanisan. Masukan kedalam array . Lalu panggil fungsi solve

Fungsi Solve:

1) Inisialisasi deque dan array result [n – k + 1]

2) Lakukan perulangan dari 0 sampai n . Jika deque tidak empty :
    - Hapus elemen dari depan (pop front) deque jika front diluar batas k

- Jika array tingkat kemanisan di index paling belakang deque v[q.back] lebih kecil dari array v[i] di index tersebut , maka lakukan popback
- Push elemen i
- Jika i>= k – 1 , artinya masih ada pilihan permen untuk diberikan ke anak anak , maka res[i-k+1] = v[q.front]

3) Print array res sejumlah n-k+1

```
#include <vector>
#include <deque>
#include <stdio.h>
#define gc getchar_unlocked

using namespace std;

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

void solve(int v[], int k, int n) {
    deque<int> q;
    int res[n - k + 1];

    for (int i = 0; i < n; i++) {
        while (!q.empty() ) {
            if (i - q.front() >= k)
                q.pop_front();
            if (v[q.back()] < v[i])
                q.pop_back();
        }

        q.push_back(i);

        if (i >= k - 1) {
            res[i - k + 1] = v[q.front()];
        }
    }
```

```
    for (int i = 0; i < n - k + 1; i++) {
        printf("%d ", res[i]);
    }
    printf("\n");
}

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

    while (t--) {
        getnum(n);
        getnum(k);

        int v[n];

        for (int i = 0; i < n; i++) {
            getnum(m);
            v[i] = m;
        }
        solve(v, k, n);
    }

    return 0;
}
```

Time complexity : O(n)

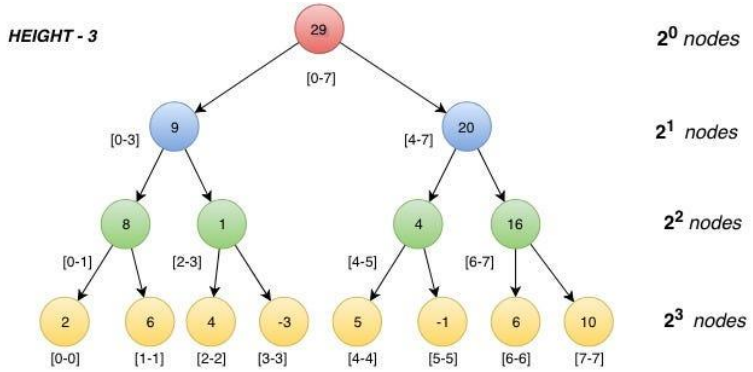
Bukti verdict:

31975300	2023-10-07 15:50:26	Johannes_C_276	accepted edit Ideone it	0.35	5.4M	CPP14

0.35 detik

Algoritma 3 : Segment Tree with Range Maximum Query

Ada algoritma lain yg dapat kita gunakan untuk menyelesaikan problem ini . Yaitu segmen Tree. Fungsi dalam segmen tree adalah untuk mencari range maximum query dari pilihan permen yg akan diberikan.



Segmen tree adalah struktur data yang menyimpan informasi tentang interval array sebagai tree. Hal ini memungkinkan menjawab berbagai persoalan range melalui array secara efisien, namun tetap cukup fleksibel untuk memungkinkan modifikasi array dengan cepat.

```
#include <stdio.h>
#include <vector>
#include <iostream>
#include <algorithm>
#define gc getchar_unlocked
using namespace std;

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

const int N = 1e5;
int n;

void build(int t[]) {
    for (int i = n - 1; i > 0; --i) t[i] = max(t[i << 1], t[i << 1 |
1]);
}

int query(int l, int r, int t[]) {
    int res = -1;
    for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
        if (l & 1) res = max(res, t[l++]);
        if (r & 1) res = max(res, t[--r]);
    }
```

```
    return res;
}

int main() {
    int cases;
    getnum(cases);

    int k;
    while (cases--) {
        getnum<int>(n);
        getnum<int>(k);

        int tree[2*N];

        for (int i = 0; i < n; ++i) {
            getnum<int>(tree[n+i]);
        }
        build(tree);

        for (int i=0; i<n-k+1; i++) {
            printf("%d ", query(i,i+k, tree));
        }
        printf("\n");
    }

    return 0;
}
```

Sebenarnya code seqmen tree yg saya buat merupakan class . namun untuk fleksibilitas dan penggunaan yg mudah , saya mengeluarkan class tersebut dan menginisialisasi array pada perulangan di setiap testCases. Lalu hanya memanggil fungsi build untuk memasukan array kedalam tree.

Time complexity : O(n * log n)

Verdict Ac :

0.39 detik

Laporan “AKVMSFT2 - Appreciating Guards in ISM 375 Points”

Permasalahan :

Setiap satpam memiliki uang yang bervariasi. Ada yg sama, lebih rendah , dan lebih tinggi dari yang lain. Karena administrasi mau semua satpam untuk memiliki uang yg sama, maka setiap satpam harus memiliki uang yang sama . Mereka menggunakan algoritma yaitu, jika satpam A memiliki maximum uang (x) dan satpam Y memiliki minimum uang (y) , maka satpam A harus mentransfer $\frac{(x-y)}{2}$ uang ke satpam B. hasil dari LIF(x) ini adalah integer yg kurang dari x. Proses ini akan dilakukan terus menerus sampai semua satpam memiliki uang yang sama. Print -1 jika tidak bisa di distribusi secara adil.

Input:

- Baris pertama adalah T , yaitu testcase
- Baris selanjutya adalah N yaitu banyaknya satpam dan baris selanjutnya ialah integer A$[i]$ sejumlah N yg menyatakan nilai uang tiap satpam

Output:

Setiap testcase, print integer yg menyatakan step minimum untuk membagi rata uang kepada seluruh satpam menggunakan algoritma LIF . print -1 jika tidak bisa di distribusi menggunakan algoritma LIF ini.

Penjelasan Masalah :

Inti dari permasalahan ini adalah kita harus menghitung langkah minimum yg dilakukan untuk membagi rata semua uang yang dimiliki oleh masing masing satpam menggunakan LIF(x) = $\frac{(x-y)}{2}$. Proses ini dilakukan secara terus menerus sampai semua satpam memiliki uang yang sama . Print -1 jika tidak bisa distribusi secara merata menggunakan algoritma LIF.

Untuk mengecek apakah dengan jumlah satpam dan total uang yang ada bisa dibagi rata adalah : $jumlah\ satpam\ \%\ n\ ==\ 0$. Jika sama dengan 0 , artinya dapat dibagi rata. Dari penjelasan deskripsi soal, kita harus menggunakan tipe data integer.

Penjelasan test case 1 :

```
2

3 2
```

Output : -1

Karena tidak bisa dibagi secara rata .

Penjelasan test case 2 :

```
4

1 5 5 9
```

$1 $5 $5 $9

Step 1 : Satpam D memberikan uang ke Satpam A

$\frac{9-1}{2}$ = $4

Satpam A = 1 + 4 = $5

Satpam D = 9 - 4 = $5

$5 $5 $5 $5

Output : 1 karena sudah terdistribusi secara rata

Penjelasan test case 3:

```
5

1 2 3 4 5
```

$1 $2 $3 $4 $5

Step 1 : Satpam E memberikan uang ke satpam A :

$\frac{5-1}{2}$ = $2

Satpam A : 1 + 2 = $3 . Satpam E : 5 – 2 = $3

$3 $2 $3 $4 $3

Step 2 : Satpam D memberikan uang ke satpam B

$\frac{4-2}{2}$ = $1

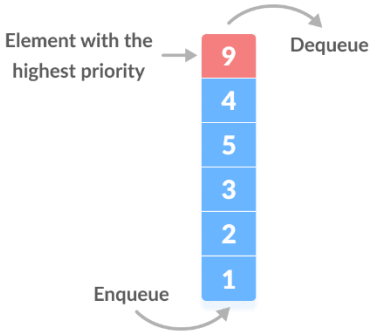
Satpam B : 2 + 1 = $3

Satpam D : 4 - 1 = $3

$3 $3 $3 $3 $3

Output : 2 karena membutuhkan dua step untuk membagi rata

Dari permasalahan dan ilustrasi tersebut, kita dapat meggunakan STL class Priority Queue untuk menyelesaikan masalah ini . Priority queue adalah jenis struktur data yang menyimpan elemen berdasarkan urutan prioritas nya (bisa secara ascending dan descending). Sifat dari priority queue adalah FIFO (First In First Out)



Priority Queue STL Method

Method	Definition
empty()	Returns whether the queue is empty.
size()	Returns the size of the queue.
top()	Returns a reference to the topmost element of the queue.
push()	Adds the element to the queue.
pop()	Deletes the first element of the queue.

Mengapa kita menggunakan priority queue untuk permasalahan ini ?

Karnena sesuai rumus LIF, yaitu $\frac{(x-y)}{2}$ dimana x adalah jumlah uang maksimal dan y adalah jumlah uang minimum . Setelah itu hasil dari x akan dikurangi dengan res dan y akan ditambah dengan res. Sehingga saat diupdate dalam array , perlu di sorting ulang apabila tidak menggunakan priority queue. Maka dari itu, kita menggunakan priority queue untuk mempermudah .

Perancangan dan Analisis Algoritma :

1) Inisialisasi integer testCases, sum , step , n , avg, min , max , res
2) Untuk setiap testcase
   - Deklarasi value sum sebagai 0 . Ambi input dari n
   - Inisialisasi 2 priority_queue yaitu p dan q
   - Lakukan perulangan sebanyak n untuk mengambil money[i] . lalu tambahkan money[i] ke sum
   - Apabila sudah selesai, cek apakah sum % n == 0 . Jika tidak = 0 , maka tidak bisa di distribusi secara rata dan print -1 . continue ke test case selanjutnya
   - Inisialisasi step = 0 dan avarage = sum/n
3) Lakukan perulangan sebanyak n untuk :
   - Apabila money[i] lebih besar dari avg , push kedalam queue p
   - Apabila money[i] lebih kecil sama dengan avg , push dalam queue q namun value dari money[i] di negatif kan
4) Lakukan while loop apabila p dan q tidak empty :
   - Ambil data dari queue p dan q
     Min = -q.top() . pop queue q
     Max = p.top() . pop queue p
   - Deklarasi variable res = (max – min) / 2
   - Kurangi nilai max dengan res dan tambahkan nilai min dengan res
   - Increment ans sebanyak 1 kali
   - Jika min lebih besar dari avg , push min ke queue p
   - Jika min lebih kecil dari avg , push -min ke queue q
   - Jika max lebih besar dari avg , push max ke queue p
   - Jika max lebih kecil dari avg , push -max ke queue q
5) Jika p dan q sudah empty , maka print step sebagai jumlah step

```
#include <cstdio>
#include <queue>
#include <vector>
#define gc getchar

using namespace std;

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

```
}

int x[2002];
vector<int> c;

int main() {
    c.reserve(2002);
    int testCases, sum, step, n, avg, min, max, res;
    getnum(testCases);

    while (testCases--) {
        sum = 0;
        getnum(n);
        priority_queue<int, vector<int>, less<int>> p(less<int>(),
move(c));
        priority_queue<int, vector<int>, less<int>> q(less<int>(),
move(c));

        for (int i = 0; i < n; i++) {
            getnum(x[i]);
            sum += x[i];
        }

        if (sum % n != 0) {
            printf("-1\n");
            continue;
        }

        step = 0;
        avg = sum / n;

        for (int i = 0; i < n; i++) {
            if (x[i] > avg)
                p.push(x[i]);
            else if (x[i] < avg)
                q.push(-x[i]);
        }

        while (!p.empty() || !q.empty()) {
            min = -q.top(), q.pop();
            max = p.top(), p.pop();
            res = (max - min) / 2;
```

```
            max -= res;
            min += res;
            step++;

            if (min > avg)
                p.push(min);
            else if (min < avg)
                q.push(-min);

            if (max > avg)
                p.push(max);
            else if (max < avg)
                q.push(-max);
        }

        printf("%d\n", step);
    }

    return 0;
}
```

Time complexity : O (n log n)

Bukti verdict AC :

Time : 0.10 s