August 25, 2025

MIXED DRILL SESSION

Ques 1 https://atcoder.jp/contests/abc130/tasks/abc130_d

You are given a sequence of positive integers of length N, $A = a_1, a_2, \ldots, a_N$, and an integer K. How many contiguous subsequences of A satisfy the following condition?

- (Condition) The sum of the elements in the contiguous subsequence is at least K.

We consider two contiguous subsequences different if they derive from different positions in A, even if they are the same in content.

Underline text hints at Two Pointers

Note that the answer may not fit into a 32-bit integer type.

Using TP

Sum > K ? Not necessarily

We can solve for (<K) and substract from total.

```
sum = 0
head = -1
tail = 0
ans = n(n+1)/2

while (tail < n) {
  while ((head+1)<n && (a[head+1]+sum)<K)
    { head++
      sum += a[head] }
  Ans = head - tail + 1
  if (tail <= head) {
    sum -= a[tail]
    tail++ }
  else {
    tail++
    head = tail - 1
    }
}
```

Ques 2. Hamburgers

https://codeforces.com/problemset/problem/371/C

Polycarpus loves hamburgers very much. He especially adores the hamburgers he makes with his own hands. Polycarpus thinks that there are only three decent ingredients to make hamburgers from: a bread, sausage and cheese. He writes down the recipe of his favorite "Le Hamburger de Polycarpus" as a string of letters 'B' (bread), 'S' (sausage) и 'C' (cheese). The ingredients in the recipe go from bottom to top, for example, recipe "BSCBS" represents the hamburger where the ingredients go from bottom to top as bread, sausage, cheese, bread and sausage again.

Polycarpus has n_b pieces of bread, n_s pieces of sausage and n_c pieces of cheese in the kitchen. Besides, the shop nearby has all three ingredients, the prices are p_b rubles for a piece of bread, p_s for a piece of sausage and p_c for a piece of cheese.

Binary search

Polycarpus has r rubles and he is ready to shop on them. What maximum number of hamburgers can he cook? You can assume that Polycarpus cannot break or slice any of the pieces of bread, sausage or cheese. Besides, the shop has an unlimited number of pieces of each ingredient.

Need: Cost per item

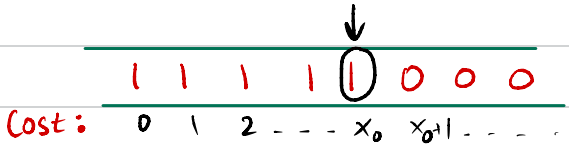
xth Burger → (3x Burger − n_B) * P_B

→ (2x Sausages − n_S) * P_S

→ (x no. of cheese − n_C) * P_C

Given

Total amount spend should be = Ans ≤ r



x_0 denotes that with x_0 cost, we can make a hamburger.
Exceeding x_0 cost and we won't be able to.

```cpp
#define endl '\n'
using lli = long long int;
lli nb_have, ns_have, nc_have;
lli pb_price, ps_price, pc_price;
lli budget;
lli req_b = 0, req_s = 0, req_c = 0;
bool check(lli num_hamburgers) {
    lli total_b_needed = num_hamburgers * req_b;
    lli total_s_needed = num_hamburgers * req_s;
    lli total_c_needed = num_hamburgers * req_c;

    lli buy_b = max(0LL, total_b_needed - nb_have);
    lli buy_s = max(0LL, total_s_needed - ns_have);
    lli buy_c = max(0LL, total_c_needed - nc_have);
    lli total_cost = buy_b * pb_price + buy_s * ps_price + buy_c * pc_price;
    return total_cost <= budget;
}

void solve() {
    string recipe;
    cin >> recipe;
    cin >> nb_have >> ns_have >> nc_have;
    cin >> pb_price >> ps_price >> pc_price;
    cin >> budget;

    for (char c : recipe) {
        if (c == 'B') req_b++;
        else if (c == 'S') req_s++;
        else req_c++;
    }
    lli low = 0;
    lli high = 2e14;
    lli ans = 0;

    while (low <= high) {
        lli mid = low + (high - low) / 2;
        if (check(mid)) {
            ans = mid;
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    cout << ans << endl;
}

signed main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    solve();
}
```

AlgoZenith

Ques 3. The Meeting Place can't be changed

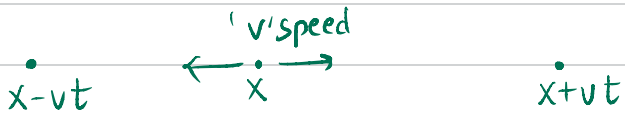
https://codeforces.com/problemset/problem/782/B

The main road in Bytecity is a straight line from south to north. Conveniently, there are coordinates measured in meters from the southernmost building in north direction.

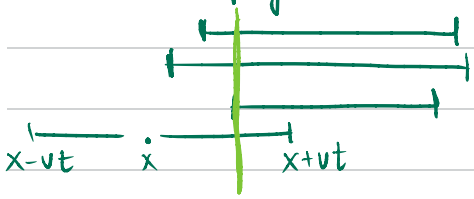
At some points on the road there are n friends, and i-th of them is standing at the point x_i meters and can move with any speed no greater than v_i meters per second in any of the two directions along the road: south or north.

You are to compute the minimum time needed to gather all the n friends at some point on the road. Note that the point they meet at doesn't need to have integer coordinate.

't' time → can they meet at 't' or not



In 't' time, it can be anywhere from x-vt to x+vt. (depending on x, this interval varies)
Now, for all the intervals, there will be a minimum time where intersection happens
If the intersection of all the intervals is non-empty then we have the ans.



- How to find intersection between two intervals (l_1, r_1) and (l_2, r_2)

⇒ $[\max(l_1, l_2), \min(r_1, r_2)]$
for 3 points
⇒ $[\max(l_1, l_2, l_3], \min(r_1, r_2, r_3)]$

```cpp
vector<int> x, v;

bool check(double time) {
    double max_left_bound = 0.0;
    double min_right_bound = 1e10;

    for (int i = 0; i < n; i++) {
        max_left_bound = max(max_left_bound, (double)x[i] - (double)v[i] * time);
        min_right_bound = min(min_right_bound, (double)x[i] + (double)v[i] * time);
    }

    return max_left_bound <= min_right_bound;
}

void solve() {
    cin >> n;
    x.resize(n);
    v.resize(n);

    for (int i = 0; i < n; i++) cin >> x[i];
    for (int i = 0; i < n; i++) cin >> v[i];

    double low = 0.0;
    double high = 1e9 + 7;

    for(int i = 0; i < 100; i++) {
        double mid = low + (high - low) / 2.0;
        if (check(mid)) {
            high = mid;
        } else {
            low = mid;
        }
    }

    cout << fixed << setprecision(12) << high << endl;
}
```

Ques 4. Quiz Master

https://codeforces.com/problemset/problem/1777/C

A school has to decide on its team for an international quiz. There are n students in the school. We can describe the students using an array a where a_i is the smartness of the i-th $(1 \le i \le n)$ student.

There are m topics $1, 2, 3, \ldots, m$ from which the quiz questions will be formed. The i-th student is considered proficient in a topic T if $(a_i \bmod T) = 0$. Otherwise, he is a rookie in that topic.

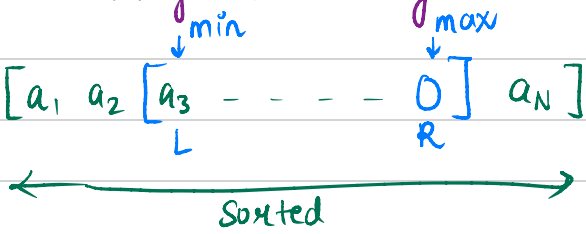
We say that a team of students is collectively proficient in all the topics if for every topic there is a member of the team proficient in this topic.

B·S on every start

Find a team that is collectively proficient in all the topics such that the maximum difference between the smartness of any two students in that team is **minimized**. Output this difference.

can be solved using both BS & TP.

- Minimize the gap (max - min)↓ of the collection you are building



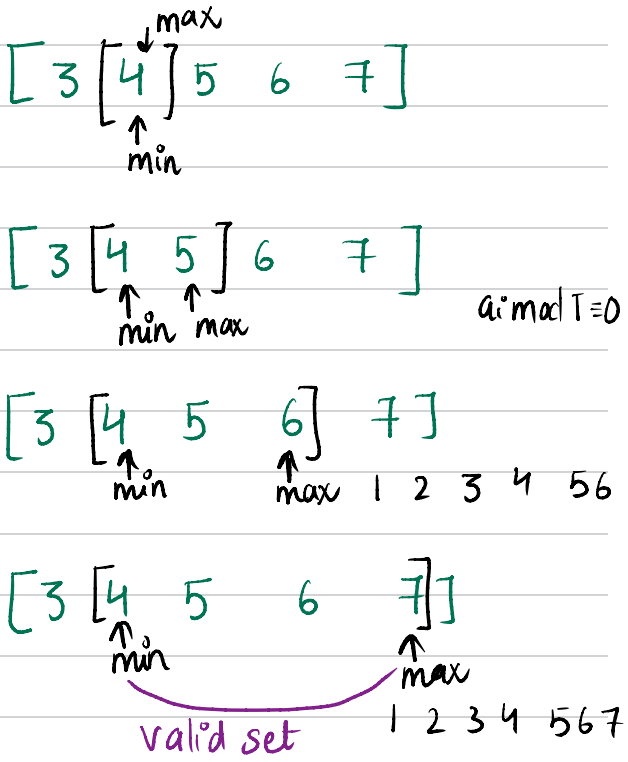
Example : no. of students

5 7 → m = 7 [1, 7]

6 4 3 5 7

There is no benefit of having duplicates. As whatever one will cover, the other will also cover the same thing.

→ Take in set to make distinct.



Then we will move our min but the above valid set will be the ans.

```cpp
int n,m;
vector<int> divisors[100010];
void pre(){
    for(int i=1;i<=100000;i++){
        for(int j=i;j<=100000;j+=i){
            divisors[j].push_back(i);
        }
    }
}
```

```cpp
// DS
int distict = 0;
int freq[100010];

void insert(int x){
    for(auto v:divisors[x]){
        if(v>m)break;
        if(freq[v]==0)distict++;
        freq[v]++;
    }

}

void erase(int x){
    for(auto v:divisors[x]){
        if(v>m)break;
        freq[v]--;
        if(freq[v]==0)distict--;
    }
}
```

```cpp
void solve(){
    cin>>n>>m;
    set<int> st;
    for(int i=0;i<n;i++){
        int x;cin>>x;
        st.insert(x);
    }

    vector<int> arr;
    for(auto v:st)arr.push_back(v);
    n = arr.size();

    ans = 1e9;
    // TP
    int head = -1,tail =0;
    while(tail<n){
        while(head+1<n && distict<m){
            head++;
            insert(arr[head]);
        }

    if(distict==m){
        ans = min(ans,arr[head]-arr[tail]);
    }

    if(tail<=head){
        erase(arr[tail]);
        tail++;
    }else{
        tail++;
        head = tail-1;
    }
}

if(ans==1e9){
    cout<<-1<<endl;
}else{
    cout<<ans<<endl;
}
```