# MINIONS - Editorial

## PROBLEM LINK:

1

Div1
Div2
Practice

**Setter-** Igor Barenblat
**Tester-** Misha Chorniy
**Editorialist-** Abhishek Pandey

## DIFFICULTY:

Medium

## PRE-REQUISITES:

Segment Tree

## PROBLEM:

Given $N$ pairs of integers, find a $sub-set$ of maximum size such that $min(a_1, a_2, \ldots, a_k) \geq \frac{b_1+b_2+..+b_k}{k}$

## QUICK EXPLANATION:

**Key to Success-** A good grasp over segment tree, implementing it properly and/or good debugging skills can help you get an AC. Implementation practice is a must.

We make $2$ arrays of pairs. The first one, $Arr[]$ will store the pair $\{a_i, b_i\}$ and second one $index[]$ will store $\{b_i, i\}$ (where $i$ is the index). We now sort both these arrays. Starting from end of $Arr[]$ (i.e. from the pair with maximum $A_i$ to pair with minimum $A_i$ [helps avoiding finding the minimum $A_i$ every time] ) we try to find maximum number of $\sum_{i=1}^{i=k} b_i$ such that $k * A_i \geq \sum_{i=1}^{i=k} b_i$ which can be done using segment tree.

## EXPLANATION:

This editorial is divided into $2-3$ sections. The first section will describe the concept and thinking process to solve the question. The second one will brief about implementation as I find it tricky solution. We will refer to tester @mgch solution for that.

The concept involved will discuss-

- What to store in segment tree?
- How will we store and why it works?
- Query and Updates. Implementation is integrated with this section.

**1.What to store in segment tree-**

I acknowledge that we can approach this problem in many ways. We will discuss tester's approach here.

He decided to store $2$ things- $\sum_{i=l}^{i=r} b_i$ in that range, and $r-l+1$ (i.e. sum of $b_i$ and number of elements in the given range). Instead of storing them in a single node, he made $2$ arrays for that. Please dont get confused by that :)

**2.How we will store and why it works-**

We made $2$ arrays $Arr[]$ and $Index[]$. We will store **sorted** $\sum_{i=l}^{i=r} b_i$ in the nodes. What does this mean and why does it work?

Lets first discuss the proof that its correct and then discuss further working.

**DING!!**
**DING!!**
**DING!!**

Oh no! You hear that alarm? It means its time for some hand exercises! Grab your pen and paper and lets get to it :).

**Q- Given a sorted array $C[]$ such that $C_1 \leq C_2 \leq C_3 \leq \ldots \leq C_k$, prove that $C_1 \leq \frac{(C_1+C_2)}{2} \leq \frac{(C_1+C_2+C_3)}{3} \ldots$.**

I think the standard mathematics proof is simple. Let me give an intuitional proof! Its given in tab below so that it doesnt give spoilers to those who want to give a try!

**Hide Content**

**Follow this question**
**By Email:**
You are not subscribed to this question.
subscribe me
(you can adjust your notification settings on your profile)
**By RSS:**
Answers
Answers and Comments

Question tags:

medium **×2,409**
segment-tree **×1,580**
mgch **×43**
cook95 **×32**

question asked: **16 Jun, 18:54**
question was seen: **1,951 times**
last updated: **yesterday**

**Related questions**

SEAD - Editorial
ADDMUL - Editorial
SLIS - Editorial
QPAIR - Editorial
DIVMAC - Editorial
MAXGRID - Editorial
PSHTBRTH - Editorial
XORSUMS - Editorial
CSUBQ - Editorial
BTMNTREE - Editorial

**Lets try to prove** $C_1 \leq \frac{C_1+C_2}{2}$

**We know that** $C_1 \leq C_2$ **. Hence, we can write** $C_2 = C_1 + p$ **where** $p \geq 0$**. Now, compare the** $L.H.S.$ **and the** $R.H.S.$

**We have-**

$C_1 \leq \frac{C_1+C_2}{2}$
$\implies C_1 \leq \frac{C_1+(C_1+p)}{2}$
$\implies C_1 \leq \frac{2*C_1+p}{2}$
$\implies C_1 \leq C_1 + p/2$

$\because p \geq 0$ **, the inequality is true for this case.**

**Similarly we can extend this argument for other cases like** $\frac{(C_1+C_2)}{2} \leq \frac{(C_1+C_2+C_3)}{3}$

The tester's proof will be given in Chef Vijju's corner.

Now, keeping the ahead proof in mind, and our objective to find the maximum size of set, we will query for maximum $k$ such that $\sum_{j=1}^{j=k} b_j + B_i \leq (k+1) * A_i$. Here $\sum_{j=1}^{j=k} b_j$ are already in the tree, and we are querying for pair $\{A_i, B_i\}$

Why $(k+1) * A_i$? Where did the division go? What is this $A_i$? How is this minimum?

**Hide Content**

**Our constraint is-**
$min(a_1, a_2, \ldots, a_{k+1}) \geq \frac{b_1+b_2+..+b_{k+1}}{k+1}$
**Cross Multiply** $k+1$**- (Since** $k \geq 0$ **the sign will not be reversed)**
$\implies (k+1) * min(a_1, a_2, \ldots, a_{k+1}) \geq \sum_{j=1}^{j=k} b_j + B_{k+1}$

**Also, we are iterating backwards. Meaning, we are starting from maximum** $A_i$ **to the minimum** $A_i$**. Hence, the current** $A_i$ **is the minimum one - no need to waste operations finding the minimum** $A_i$**!!**

The required query is standard. But wait! How do we guarantee that the $B_i$ chosen are only from the pairs such that their $A_i$ are greater than or equal to the current $A_i$ we are using this as minimum? This will be discussed in next section, where we will complete whats exactly stored in the tree and how/why it works. Make sure that the question/proof given to you above is clear!

**3. Query and Update-**

Now, you guys would be at least a little confused. Things were going smooothly but then this evil editorialist threw up an evil question on your face :(

Well, turns out its simple to fix as well!

Instead of building tree at once, we build it step by step. What we do, is, that the tree is initially empty.

As we iterative from pairs with maximum $A_i$ to minimum $A_i$, we add the $B_i$ to the tree by updating the tree.

**Hide Content**

**Lets dry run over 2 iterations to understand.**

**First iteration- The tree is empty. The first query happens with pair which has maximum** $A_i$**. Since the tree is empty, the first query returns** $1$ **if** $A_i \geq B_i$ **else we get** $0$

**We know find what index/range in tree to update (using** $index[]$ **array which has** $B_i$ **sorted in the required order). We update the leaf and parent nodes. Now** $B_i$ **has been added to tree. The tree, which was initially empty, now has** $B_i$

**I think now you guys can predict what happens in second iteration. You may want to go above at the find paragraph of Section 2. to dry run it :)**

Reference code is below-

**Hide Content**

```
//For all i from n-1 to 0 (we sorted the pairs already!)
//Do query
//Now we have to update. See code below.
int where = lower_bound(idx.begin(), idx.end(), make_pair(f[i].second, i)) -
idx.begin();//Find required position. idx=index array
upd(where, +f[i].second);
```

What do we do in update function? For that, lets first discuss relation between parent and child.

Recall that we are storing **"storing sorted** $\sum_{i=l}^{i=r} b_i$**, and the number of elements** (for convenience purposes only)**in the nodes"**. So, what can the relation be? If we know the information in $[L, mid]$ and the information in $[mid+1, R]$, how can we calculate information for $[L, R]$??

**Hide Content**

**We are storing the sum of elements in range** $[L, R]$ **and number of elements in that range. Hence, the relation is-**

**Sum in range** $[L, R]$**=Sum in range** $[L, mid]$**+Sum in range** $[mid+1, R]$

**Same for the number of elements. (Its stored only for convenience)**

Can you now, knowing the relation between nodes, try to frame the update and query function? In update function you have to update correct leaf and hence its parents, and in query you must obtain the result. Reference codes of tester (iterative) are given below. do try to put them in recursive for practice :)

upd function is given below-

**Hide Content**

```
 void upd(int r, int val) {
        r += sz;//From leaf
        t[r] += val;//t stores sum of Bi
        c[r] += 1;//Stores number of elements in range
        while (r > 1) {
            t[r >> 1] = t[r] + t[r ^ 1];//Parent child=Stats of Left Child+Stats of Right
Child
            c[r >> 1] = c[r] + c[r ^ 1];
            r >>= 1;
        }
    }
```

The query function is also easy. The tester used iterative segment tree. The query function (along with update) is given below-

**Hide Content**

```
    for (int i = n - 1; i >= 0; --i) {//For all pairs from n-1 to 0
        if (f[i].first >= f[i].second) {
            ans = max(ans, 1);//Ans at least 1 if Ai>=Bi
        }
        long long cur = f[i].second;
        int now = 1, root = 1;//Root is 1.
        while (root < sz) {//sz=size of tree
            //Even child (2n) has lesser sum than (2n+1) child.
            if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
                now += c[root];//Now= current Ans
                ans = max(ans, now);
                break;
            }
            root <<= 1;//Check the child
            if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
                now += c[root];
                cur += t[root];//Curr=Sum Bi till now
                ans = max(ans, now);
                root |= 1;//equivalent to root+=1. Done to check odd child in next
iteration
            }
        }
            //After querying, add the Bi to the tree.
        int where = lower_bound(idx.begin(), idx.end(), make_pair(f[i].second, i)) -
idx.begin();
        upd(where, +f[i].second);
```

Now your turn! Refer to tester's code. Right now, you might feel its easy to do. Try writing your own recursive version of the code! You will face a few (or many) WA, dont worry. Debug them, that will give you tremendous improvement. Refer to editorial, ask doubts! Dont get disheartened by WAs :). Debugging segment tree sometimes give headaches even to red coders, so practice the proper implementation by writing the recusrive solution!

## SOLUTION:

The codes which I received are pasted below for reference. This is done so that you dont have to wait for @admin to link solutions up. Please copy them and paste at a place where you are comfortable to read :).

Setter

**Hide Content**

```
#include <bits/stdc++.h>

using namespace std;

const int MaxN = 4e5 + 15;
const int INF = 1e9;

pair<long long, int> tree[MaxN * 32];
int sz;

int L[MaxN * 32], R[MaxN * 32];
```

```
int findRes(int x, int l, int r, int k, pair<long long, int> cur)
{
    if(!x)
        return cur.second;

    if(l == r)
    {
        int ll = 0;
        int rr = tree[x].second;

        int res = 0;

        while(ll <= rr)
        {
            int mid = (ll + rr) / 2;
            if(cur.first + l * 1ll * mid <= k * 1ll * (mid + cur.second))
            {
                res = mid;
                ll = mid + 1;
            }else
                rr = mid - 1;
        }

        return cur.second + res;

    }else
    {
        int mid = (l + r) / 2;
        if(cur.first + tree[L[x]].first <= k * 1ll * (tree[L[x]].second + cur.second))
        {
            cur.first += tree[L[x]].first;
            cur.second += tree[L[x]].second;
            return findRes(R[x], mid + 1, r, k, cur);
        }else
            return findRes(L[x], l, mid, k, cur);
    }
}

void up(int x, int l, int r, int pos)
{
    if(l == r)
    {
        tree[x].first += l;
        tree[x].second++;
    }else
    {
        int mid = (l + r) / 2;
        if(mid >= pos)
        {
            if(!L[x])
                L[x] = ++sz;
            up(L[x], l, mid, pos);
        }else
        {
            if(!R[x])
                R[x] = ++sz;
            up(R[x], mid + 1, r, pos);
        }

        tree[x].first = tree[L[x]].first + tree[R[x]].first;
        tree[x].second = tree[L[x]].second + tree[R[x]].second;
    }
}

void solve()
{
    //freopen("input.txt", "r", stdin);
    //freopen("output.txt", "w", stdout);

    vector <pair<int, int> > a;

    int n;
    cin >> n;
    a.resize(n);
    int root=++sz;

    for(auto & x : a)
        cin >> x.first >> x.second;
```

```
    sort(a.begin(), a.end());
    reverse(a.begin(), a.end());

    int ans = 0;

    for(auto x : a)
    {
        up(root, 1, INF, x.second);
        ans = max(ans, findRes(root, 1, INF, x.first, {0, 0}));
    }

    cout << ans << '\n';
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int test;
    cin>>test;
    while (test--){
        solve();
    }
}
```

Tester

**Hide Content**

```
#include <bits/stdc++.h>

using namespace std;

const int MaxN = (int)4e5 + 10;
const int MOD = (int)1e9 + 7;
const int INF = 1e9;

int n;
pair < int, int > f[MaxN];
int c[1 << 20], sz;
long long t[1 << 20];


long long readInt(long long l,long long r,char endd){
    long long x=0;
    int cnt=0;
    int fi=-1;
    bool is_neg=false;
    while(true){
        char g=getchar();
        if(g=='-'){
            assert(fi==-1);
            is_neg=true;
            continue;
        }
        if('0'<=g && g<='9'){
            x*=10;
            x+=g-'0';
            if(cnt==0){
                fi=g-'0';
            }
            cnt++;
            assert(fi!=0 || cnt==1);
            assert(fi!=0 || is_neg==false);

            assert(!(cnt>19 || ( cnt==19 && fi>1) ));
        } else if(g==endd){
            if(is_neg){
                x= -x;
            }
            assert(l<=x && x<=r);
            return x;
        } else {
            assert(false);
        }
    }
```

```
}
string readString(int l,int r,char endd){
    string ret="";
    int cnt=0;
    while(true){
        char g=getchar();
        assert(g!=-1);
        if(g==endd){
            break;
        }
        cnt++;
        ret+=g;
    }
    assert(l<=cnt && cnt<=r);
    return ret;
}
long long readIntSp(long long l,long long r){
    return readInt(l,r,' ');
}
long long readIntLn(long long l,long long r){
    return readInt(l,r,'\n');
}
string readStringLn(int l,int r){
    return readString(l,r,'\n');
}
string readStringSp(int l,int r){
    return readString(l,r,' ');
}

void upd(int r, int val) {
    r += sz;
    t[r] += val;
    c[r] += 1;
    while (r > 1) {
        t[r >> 1] = t[r] + t[r ^ 1];
        c[r >> 1] = c[r] + c[r ^ 1];
        r >>= 1;
    }
}

int en;

void solve() {
//  scanf("%d", &n);
    n = readIntLn(1, 4e5);
    en += n;
    assert (en <= 4e5);
//  k * min(a1, a2, ..., ai) >= b1 + ... + bi
    vector < pair < int, int > > idx(n);
    for (int i = 0; i < n; ++i) {
        f[i].first = readIntSp(1, 1e9);
        f[i].second = readIntLn(1, 1e9);
//      scanf("%d%d", &f[i].first, &f[i].second);
    }
    sort(f, f + n);
    for (int i = 0; i < n; ++i) {
        idx[i] = make_pair(f[i].second, i);
    }
    sz = 1;
    while (sz < n) {
        sz *= 2;
    }
    sort(idx.begin(), idx.end());
    for (int i = 0; i < 2 * sz; ++i) {
        t[i] = c[i] = 0;
    }
    int ans = 0;
    for (int i = n - 1; i >= 0; --i) {
        if (f[i].first >= f[i].second) {
            ans = max(ans, 1);
        }
        long long cur = f[i].second;
        int now = 1, root = 1;
        while (root < sz) {
            if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
                now += c[root];
                ans = max(ans, now);
                break;
```

```
            }
            root <<= 1;
            if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
                now += c[root];
                cur += t[root];
                ans = max(ans, now);
                root |= 1;
            }
        }
        int where = lower_bound(idx.begin(), idx.end(), make_pair(f[i].second, i)) -
idx.begin();
        upd(where, +f[i].second);
    }
    printf("%d\n", ans);
}

int main() {
//  freopen("input.txt", "r", stdin);
    int t = readIntLn(1, 1e3);
    while (t --> 0) {
        solve();
    }
    assert (getchar() == EOF);
    return 0;
}
```

Editorialist's solution will be put on demand.

Edit- A lot of people have been expressing their unability to be able to come up with a recursive function of tester's solution. Its ok, its a part of learning. I will update editorialist's solution for you to cross check. The code also has some questions related to implementation. Further, I have added one test file in test case bank. Its huge, but it was the most trickiest one, so I hope it helps. Please mail me/ping me if there are ANY concerns. Thank you :)

Editorialist's Solution (Recursive Segment Tree)

$Time\ complexity = O(NlogN)$

## CHEF VIJJU'S CORNER:

**1.** I mentioned a line in the comments of query function. `//Even child (2n) has lesser sum than (2n+1) child..` Prove this. Hint in tab below. (You may assume same number of terms in both nodes)

**Hide Content**

**$B_i$ are sorted! Also, even child stores sum from range $[l, mid]$ and odd child stores from $[mid+1, r]$. Assume their sizes are same (i.e mid-l+1==r-mid).**

**2. Tester's Notes-**

**Hide Content**

**Sort all pairs increasing by $(A_i, B_i)$. We're trying to fix the element with minimal $A_i$. After that, let's take all $(A_j, B_j) \geq (A_i, B_i)$, let's sort them by $B$ and consider only $B's$.**

**Now we're interested in finding the maximal $K$ such that $A_i >= (B_1 + \ldots + B_k + B_i)/(K+1)$. It can be done with some logarithmic search on a segment tree with compressed coordinates of array $B$.**

**Prove it: if you have sorted array $C = (C_1, C_2, \ldots, C_k)$.**
**$Now, C_1/1 \leq (C_1 + C_2)/2 \leq (C_1 + C_2 + C_3)/3 \leq \ldots \leq (C_1 + \ldots + C_k)/k$.**

**Once again, let's fix $A_i$ $(i = n..1)$ one by one and maintain all sorted $B(j = i+1..n)$ in segment tree. Let's find the largest $k$ such $(C_1 + \ldots + C_k + B_i)/(K+1) \leq A_i$. Make sure that ST contains exactly $2^l$ leaves.**

**3. Tester's Proof-**

**Hide Content**

**Okay, ($(C_1 + \ldots + C_i)/i \leq (C_1 + \ldots + C_{i+1})/(i+1)$, $C_1 <= C_2 <= \ldots <= C_{i+1}$ Let's prove it:**

$(C_1 + \ldots + C_i)/i * (i+1) \leq C_1 + \ldots + C_{i+1}$

$(C_1 + \ldots + C_i) * 1/i \leq C_{i+1}$

$(C_1 + \ldots + C_i) * 1/i \leq (C_{i+1} + \ldots + C_{i+1}) * 1/i$

$0 \leq 1/i((C_{i+1} - C_1), + \ldots + (C_{i+1} - C_i))$

**I hope it's clear :)**

**4.** Read tester's notes. He said something about "**Tree must have $2^l$ leaves**" for his iterative version to work. Why?

**5.** Refer to tester's solution. Try to write a **recursive solution** to the same. :)

**6. Test Case Bank-**

**Hide Content**

**Tester's Suggestion- Be careful when ALL elements are included and when answer is $0$!!**

**All test cases were huge. Give me some solutions to analyze and we will add it. Community requested to contribute by giving any failing case in format-**

```
Input
Expected Output
Cause of Error
```

**One of extra test case from setter-**

```
1
20
10 1
10 5
5 4
3 9
6 9
10 3
1 5
5 3
3 6
2 1
9 10
5 10
2 4
8 1
1 1
10 7
10 10
5 3
6 9
1 5
Answer:
11
```

Official Test Case (one of the files)- Uploaded on my github. Pastebin etc. were not allowing pasting a file this big.
Link- https://github.com/Vijju1234567890/Test-Cases

**7.** OMG YOU READ TILL HERE? Here, I have an awesome blog on segment trees for you :) . Click here

**8. Related Problems-**

- Set of Questions
- One question from Hackerearth. Check out their section for more! Refer to my previous segment tree editorials as well for more :)

segment-tree medium cook95 mgch

This question is marked "community wiki".

edited **20 Jun, 14:06**

asked **16 Jun, 18:54**

5★ vijju123 ♦
[13.8k]●1●11●40
accept rate: 19%

Where you stated that we find maximum k such that, $\sum_{j=1}^{k} b_j + B_i <= (k+1) * A_i$ I dont understand why we $\sum b_j$ from 1 to k, we could have found maximum and why not for some other l and r?

2★ ay2306 (18 Jun, 19:47)

What?

I had to use some notation to convey that we are querying over sum of $b_j$. I think you are getting unnecessarily confused?

5★ vijju123 ♦ (18 Jun, 19:51)

@vijju123 @ay2306 is asking for purpose of sorting. And how it is helpful here.

3★ aryanc403 (18 Jun, 20:23)

1 He should check the proof and hand exercises then. I think the proof and deduction via it is the best way for him :)

5★ vijju123 ♦ (18 Jun, 20:29)

**13 Answers:**

oldest answers newest answers popular answers

1 2 next »

**11**

@akshatsharma Yes , I solved it using binary search. We can binary search on the maximum size of the subset. Now to check if it is possible to form a good subset of size "k", we first make a pair of { power , endurance } and sort it in descending order of power. Now we iterate from the highest power to lowest power. When we are at any index idx, its corresponding power will be minimum from 1 to idx and we can consider all endurances from 1 to idx - 1 and take the minimum k - 1 endurances. We can keep track of k - 1 minimum endurances using priority queue. Let the sum of minimum k - 1 endurances be sum , then if power[idx] * k >= (sum + endurance[idx] ) , then it is possible to form a good subset of size k. Unfortunately, due to a silly bug I couldn't get it accepted during contest :( . AC Code. Do let me know if you didn't understand anything :)

link | award points

answered **18 Jun, 18:57**

4★ tihorsharma123
[457]●1●8
accept rate: 15%

complexity O(nlogn) Nice Approach @tihorsharma.. got Ac in java
https://www.codechef.com/viewsolution/18943916

3★ hemant_dhanuka (19 Jun, 01:49)

1 Thank you :)

4★ tihorsharma123 (19 Jun, 01:56)

**3** Yes surely. See what I am doing is sorting at first wrt to ai, then taking a seperate array which stores {bi,i} (here i is the index after sorting first time) and then sort that 2nd array. You can easily see for a[0] you can include all the elements if you want. Formally for any ai, you need sum of b's in sorted order from i to n. Also note that once a bk gets included for some aj, it's not removed for some ai>aj until bk belongs to some ak<ai.(WHY? Because otherwise ak will be the minimum.) . So for this reason I introduced the blocked array so that I don't include some bk in my answer when ai is the current minimum and ak<ai, where {ak,bk} was a pair. Have a look at my implementation, hopefully you will understand.

link | award points

answered **18 Jun, 01:09**

6★ soham1234
[1.7k]●6●14
accept rate: 20%

**1** @vijju123 Tester's code gives wrong answer for following case
1
12
10 100
12 1
12 1
12 1
12 1
12 1
12 1
12 1
12 1
12 1
12 1
12 100000000
correct answer is 11 whereas Tester's code outputs 10.

link | award points

answered **22 Jun, 23:29**

6★ ankurdua15
[36]●1
accept rate: 20%

Will inform him that his solution got hacked xD. Nice job. The setter's solution (used to make TC) is correct. The idea is also correct, he made a minor bug in implementation, so no major problem. Thanks :)

5★ vijju123 ♦ (23 Jun, 01:25)

**2** I don't think segment tree is necessary. We see that if we store {bi,i} in increasing order then our pointer always moves right. Also only case where it might fail is when the bi we are going to include belongs to some ai which is less than our current minimum. But this can be tackled introducing blocked array.

link | award points

answered **18 Jun, 00:52**

6★ soham1234
[1.7k]●6●14
accept rate: 20%

**1** Yes. Even my intuition was that segment tree can be avoided. But I had no time to experiment as I wanted to publish editorials on time. Theres a reason why its said "Editorialist's solution to be provided on demand" instead of giving it by default. I feel without proper investigation/explorations they wont be worthy enough :)

5★ vijju123 ♦ (18 Jun, 01:03)

@soham1234 @vijju123 Can you please tell me why my code is giving me WA. Thanks
https://www.codechef.com/viewsolution/18941742

4★ underdog_eagle (18 Jun, 19:52)

**1** Can anyone provide a solution without using segment tree? With detailed explanation.

link | award points

answered **18 Jun, 18:50**

3★ romok
[11]●2
accept rate: 0%

i do agree with you that it would be so appreciable if setter has added comments in his code , that would make it lot easier.

3★ gyanendra371 (19 Jun, 10:29)

**0** @soham1234 can you please explain your solution a bit more elaborately (the significance of blocked array)

only case where it might fail is when the bi we are going to include belongs to some ai . What does it mean?

link | award points

edited **18 Jun, 01:08**

answered **18 Jun, 01:02**

5★ aman_robotics
[31]●4
accept rate: 11%

**0** min(a1,a2,...,ak)≥ (b1+b2+..+bk)/k , can we solve this query using binary search ?

link | award points

answered **18 Jun, 18:04**

4★ akshatsharma
[28]●2

accept rate: 0%

I did saw some solutions using binary search. I think we can!

5★ vijju123 ♦ (18 Jun, 18:09)

I'm also searching soln similar with this idea. If someone finds or had done this. Do post their soln here.

3★ aryanc403 (18 Jun, 19:25)

@aryanc - Check accepted answer of @tihorsharma123 , he used Binary Search afaik

5★ vijju123 ♦ (18 Jun, 19:39)

**0**

I came across this solution which uses only priority queue and no binary search( https://www.codechef.com/viewsolution/18928339 ) .It got AC during contest.

For test case:-

1

3 1

1 2

3 3

This code returns 1 for above testcase. Shouldn't the answer be 2? We can have minions {1,3} with mean 2 and minimum as 3. Is there something wrong with my understanding?

link | award points edited **18 Jun, 20:46** answered **18 Jun, 20:46**

3★ zephyr_23
[3]●2
accept rate: 0%

You mentioned number of minions as 1 here. So only minion 3 1 is considered.

5★ vijju123 ♦ (18 Jun, 20:48)

**0**

@tihorsharma123 @vijju123, why are you maintaining "ascending order or sorted B array of k elements" when doing binary search. I don't understand why is it necessary for array B to be sorted in ascending order.

link | award points edited **18 Jun, 21:40** answered **18 Jun, 21:36**

rajesh_xyz
[1]●1
accept rate: 0%

Did you try this proof-

Q- Given a sorted array C[] such that C1≤C2≤C3≤...≤Ck, prove that C1 ≤(C1+C2)/2≤(C1+C2+C3)/3.

5★ vijju123 ♦ (18 Jun, 21:41)

Because of the inequality given in the question. In my approach ,I fixed the size of the subset ( let it be k ) and current minimum to be pow. Then pow >= (b1 + b2 + ... bk) / k which is nothing but pow * k >= (b1 + b2 + .... bk) . Therefore using greedy approach it makes sense to take the minimum k values of B array. Hope it helps. Let me know if something is unclear.

4★ tihorsharma123 (18 Jun, 21:43)

Yes, I tried that proof. When searching for appropriate subset, why does popping minimum element in array B work ?

rajesh_xyz (18 Jun, 21:43)

You want fit in as many balls in a bag of weight $W$. What do you chose? Lighter balls or heavy balls? We only have to maximise number of balls in bag.

Then apply the logic to current scenario. Thanks @tihorsharma123 for helping me :)

5★ vijju123 ♦ (18 Jun, 21:53)

**1** No problem :) @vijju123

4★ tihorsharma123 (19 Jun, 01:56)

**0**

how to convert the query function into recursive solution? mainly this part from tester solution:

```
while (root  sz) {
        if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
            now += c[root];
            ans = max(ans, now);
            break;
        }
        root <<= 1;
        if ((t[root] + cur) <= 1LL * (c[root] + now) * f[i].first) {
            now += c[root];
            cur += t[root];
            ans = max(ans, now);
            root |= 1;
        }
    }
```

link | award points answered **19 Jun, 02:02**

4★ akshatsharma
[28]●2
accept rate: 0%

Setter's function is recursive. Try checking it for hints first. :)

5★ vijju123 ♦ (19 Jun, 02:19)

1 2 next »

## Your answer

[hide preview] community wiki:

Preview

Answer the question