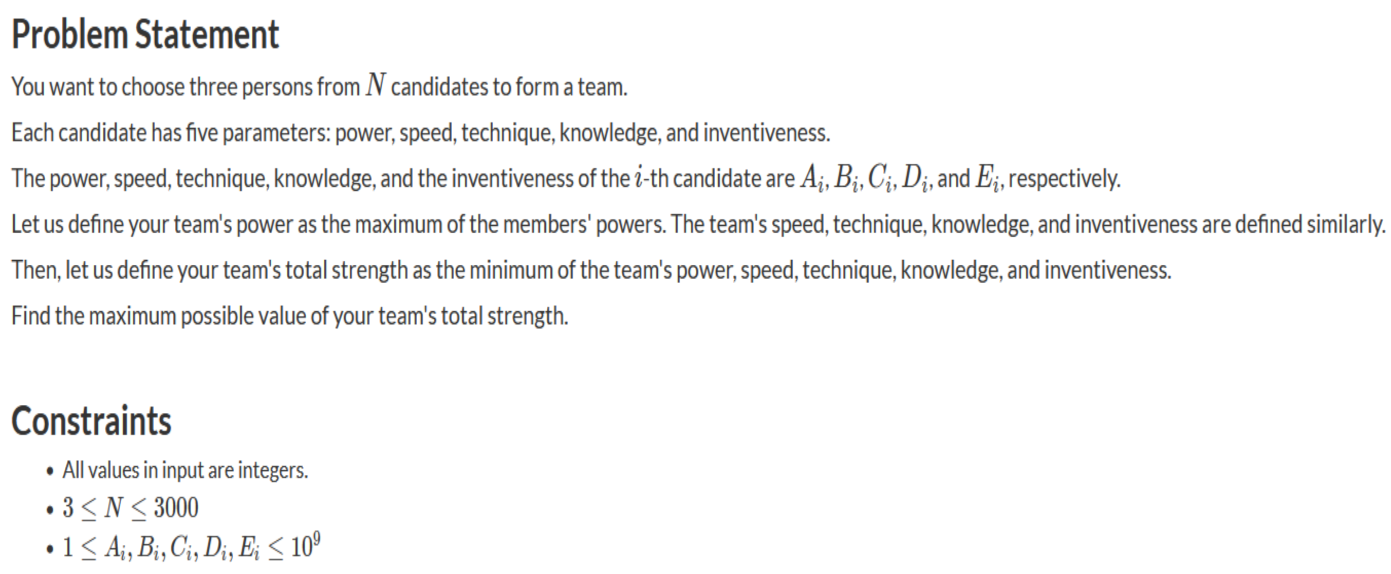
Binary and Ternary Search

Q. C - MAD TEAM Editorial



Problem Statement

You want to choose three persons from N candidates to form a team.

Each candidate has five parameters: power, speed, technique, knowledge, and inventiveness.

The power, speed, technique, knowledge, and the inventiveness of the i-th candidate are A_i, B_i, C_i, D_i, and E_i, respectively.

Let us define your team's power as the maximum of the members' powers. The team's speed, technique, knowledge, and inventiveness are defined similarly.

Then, let us define your team's total strength as the minimum of the team's power, speed, technique, knowledge, and inventiveness.

Find the maximum possible value of your team's total strength.

Constraints

- All values in input are integers.
- $3 \leq N \leq 3000$
- $1 \leq A_i, B_i, C_i, D_i, E_i \leq 10^9$

```
10  0  0  0  0
0  10  0  0  0
0  0  10  0  0
0  0  0  10  0
0  0  0  0  10
```

3
3 9 6 4 6
6 9 3 1 1
8 8 9 3 7

10010
01001
10100
(True)

(answer 3 or more)

1 1 1 1 1
1 1 1 0 0
1 1 1 1 1
(for answer 4 or more)

0 1 1 1 1	30
1 1 0 0 0	3
1 1 1 0 1	23

3 9 6 4 6
6 9 3 1 1
8 8 9 3 7
(for answer 5 or more)

0 1 1 0 1 // 13
1 1 0 0 0 // 24
1 1 1 0 1 // 29

0 31

2^0 + 2^1 + 2^2 ...2^4

Solution with bitmasks

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
const long long N=200005, INF=2000000000000000000;


int n;
int a[3005][5];
int b[5];



bool good(int m){

    int bin[n][5];

    for(int i=0;i<n;i++){
        for(int j=0;j<5;j++){
            bin[i][j] = (a[i][j]>=m);
        }
    }

    set<int> distinct;

    for(int i=0;i<n;i++){
        int mask=0;
        for(int j=0;j<5;j++){
            mask|=(bin[i][j]<<j);
        }
        distinct.insert(mask);
    }

    for(auto i:distinct){
        for(auto j:distinct){
            for(auto k:distinct){
                int val = i|j|k;
                if(val==31) return true;
            }
        }
    }

    return false;
}

int32_t main()
{
    cin>>n;
    memset(b, 0, sizeof b);
    for(int i=0;i<n;i++){
        for(int j=0;j<5;j++){
            cin>>a[i][j];
            b[j] = max(b[j], a[i][j]);
        }
    }

    int l=0; // good
    int r=1e9+5; //bad


    while(r>l+1){
        int m = (l+r)/2;
        if(good(m)){
            l=m;
        }else{
            r=m;
        }
    }


    cout<<l<<"\n";

}
```

Solution without bitmasks

Claim : For three person to satisfy the criteria, we claim that at least two person amongst them should satisfy for four properties.
Proof:
Let us suppose that there exist a answer that have two person which satisfy x<=3 properties only.
Then it means that the remaining properties would be satisfied by the remaining person for these three two be valid combination.
So if x = 3 and two person have only first three properties satisfied, then their combination would be
1 1 1 0 0
and fourth person should must have last two satisfying so for third person 0 0 0 1 1
Now the first two can be one of these two worst cases:
1 0 0 0 0 and 0 1 1 0 0
or 1 1 0 0 0 and 0 0 1 0 0
In either of the case we can choose two person such that four properites would be satisfied.
Similarly you can try with x = 0, 1, and 2
Hence we proved our claim.

So then the question simply converts to checking only for combination of two person and check whether they satisfy four or more properties.
And for the property which is not satisfied we can use the person amongst all with the largest value of that property and check if it is
greater than equal to ANSWER.

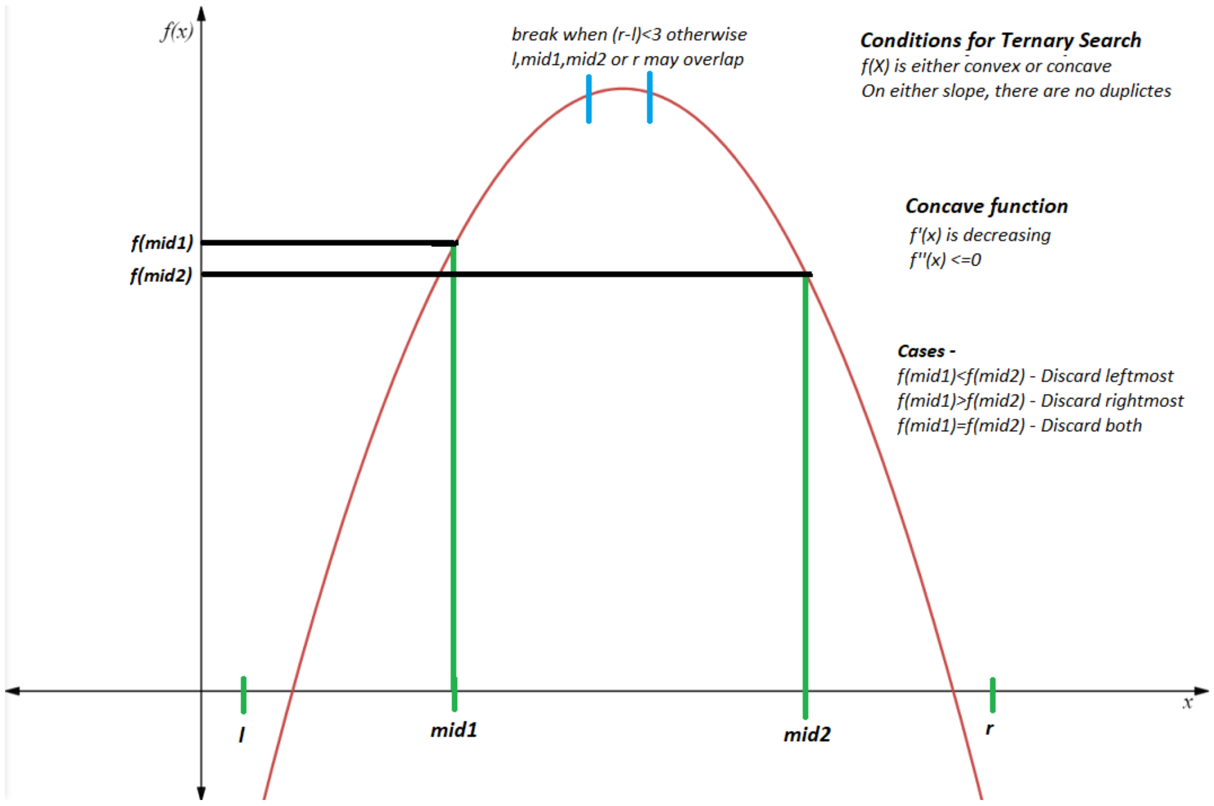
```
#include <bits/stdc++.h>
using namespace std;

int n;
int a[3005][5];
int b[5];

bool good(int m)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            int c[5];
            int cnt = 0;
            for (int k = 0; k < 5; k++)
            {
                c[k] = max(a[i][k], a[j][k]);
                cnt += (c[k] < m);
            }
            if (cnt > 1)
            {
                continue;
            }

            if (cnt == 0)
                return true;

            for (int k = 0; k < 5; k++)
            {
                if (c[k] < m)
                {
                    if (b[k] >= m)
                        return true;
                }
            }
        }
    }

    return false;
}

int32_t main()
{
    cin >> n;
    memset(b, 0, sizeof b);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            cin >> a[i][j];
            b[j] = max(b[j], a[i][j]);
        }
    }

    int l = 0;        // good
    int r = 1e9 + 5; //bad

    while (r > l + 1)
    {
        int m = (l + r) / 2;
        if (good(m))
        {
            l = m;
        }
        else
        {
            r = m;
        }
    }

    cout << l << "\n";
}
```

```
set <vector <int> > s;
for(int i=0;i<n;i+)
{
        Vector <int> temp;
        for(int j=0;j<5;j++)
        temp.push_back(a[i][j]);
        s.insert(temp);
}
```

__builtin_popcount(n); 7 (111 binary)- 3
__builtin_clz(n);
1
__builtin_ctz(n);
4

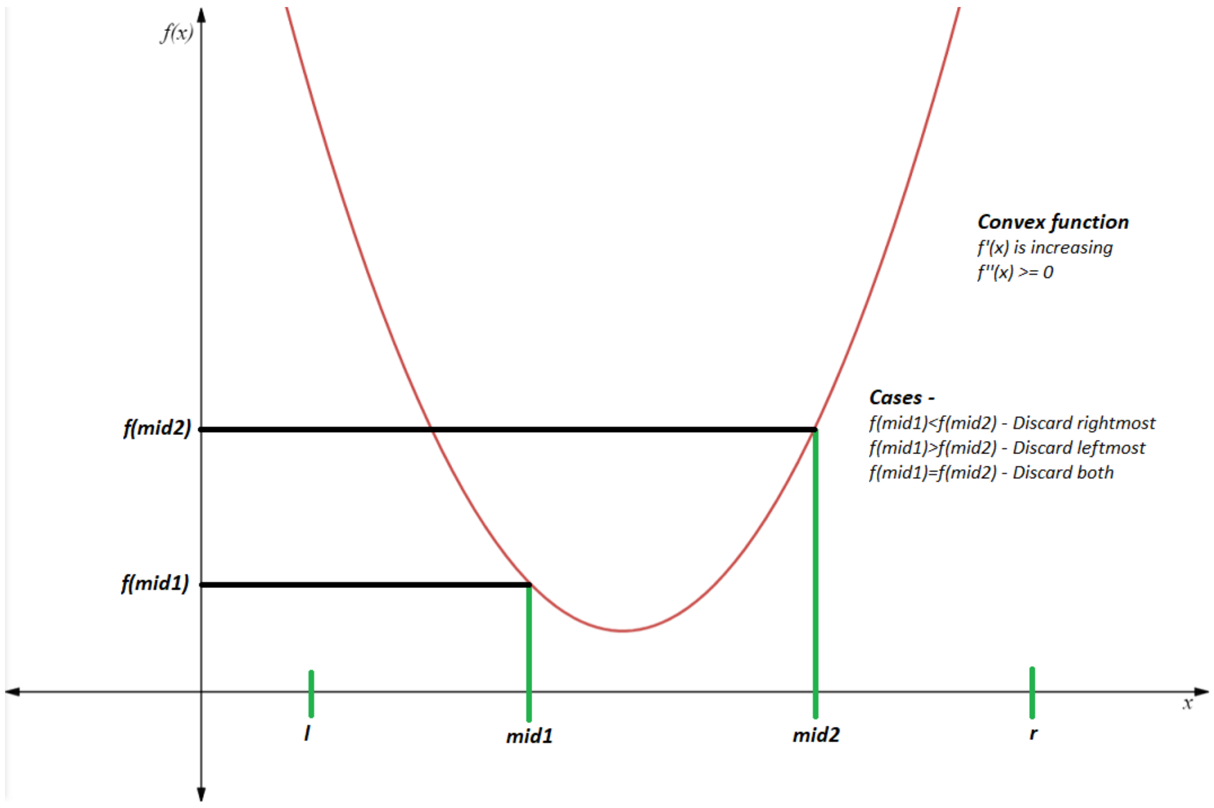
Ternary Search

Concave function -


f(x)
break when (r-l)<3 otherwise
l,mid1,mid2 or r may overlap
Conditions for Ternary Search
f(X) is either convex or concave
On either slope, there are no duplictes
Concave function
f'(x) is decreasing
f''(x) <=0
Cases -
f(mid1)<f(mid2) - Discard leftmost
f(mid1)>f(mid2) - Discard rightmost
f(mid1)=f(mid2) - Discard both
f(mid1)
f(mid2)
l
mid1
mid2
r
x

```
int l = 0, r = 1e9;
while(r-l>=3)
{
        int mid1 = l + (r-l)/3;
        int mid2 = r - (r-l)/3;
        if(f(mid1)>f(mid2))
                r = mid2;
        else if(f(mid1)<f(mid2))
                l = mid1;
        else
        {
                l=mid1;
                r=mid2;
        }
}
```

Convex function -



f(x)
Convex function
f'(x) is increasing
f''(x) >= 0
Cases -
f(mid1)<f(mid2) - Discard rightmost
f(mid1)>f(mid2) - Discard leftmost
f(mid1)=f(mid2) - Discard both
f(mid2)
f(mid1)
l
mid1
mid2
r
x

```
int l = 0, r = 1e9;
while(r-l>=3)
{
        int mid1 = l + (r-l)/3;
        int mid2 = r - (r-l)/3;
        if(f(mid1)<f(mid2))
                r = mid2;
        else if(f(mid1)>f(mid2))
                l = mid1;
        else
        {
                l=mid1;
                r=mid2;
        }
}
```

Time Complexity -

Binary Search - n -> n/2 : O(log_2 n)
Ternary Search - n -> 2n/3 : O(log_(3/2) n)
(⅔)^k * n = 1
(⅔)^k = 1/n
(3/2)^k = n
k = log_(3/2) n

Q. Problem - 439D

f(x) = Sum of b[i]-x for all b[i]>x + Sum of x-a[i] for all a[i]<x
f'(x) is increasing
f''(x) >= 0
You have p elements in b which are > x and you have q elements in a which are < x
f(x) = b[i1]-x+b[i2]-x+...+b[ip]-x + x-a[j1]+x-a[j2]+...+x-a[jq]
f'(x) = -p+q
if x increases, p can decrease and q can increase
f'(x) will increase
f''(x)>=0
So, our function f(x) is convex.
If we increase x to x+1, f(x) will decrease by p and increase by q. If p!=q then f(x) will change. if p=q, f'(x) = 0 meaning that this is our minima.

We can apply ternary search.

```
inf f(int x)
{
        int sum = 0;
        for(int i=0;i<n;i++)
        {
                if(a[i]<x)
                        sum+=x-a[i];
        }
        for(int i=0;i<m;i++)
        {
                if(b[i]>x)
                        sum+=b[i]-x;
        }
        return sum;
}

int l = 1, r = 1e9;
while(r-l>=3)
{
        int mid1 = l + (r-l)/3;
        int mid2 = r - (r-l)/3;
        if(f(mid1)<f(mid2))
                r = mid2;
        else if(f(mid1)>f(mid2))
                l = mid1;
        else
        {
                l=mid1;
                r=mid2;
        }
}
```

```
int mn = INF;
for(int i=l;i<=r;i++)
        mn = min(mn,f(i));
cout << mn << '\n';
```

Google Kickstart Round F Problem B -

```cpp
 #include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#define int long long
#define IOS std::ios::sync_with_stdio(false);
cin.tie(NULL);cout.tie(NULL);cout.precision(dbl::max_digits10);
#define pb push_back
#define mod 1000000007ll //998244353ll
#define lld long double
#define mii map<int, int>
#define pii pair<int, int>
#define ff first
#define ss second
#define all(x) (x).begin(), (x).end()
#define rep(i,x,y) for(int i=x; i<y; i++)
#define fill(a,b) memset(a, b, sizeof(a))
#define vi vector<int>
#define setbits(x) __builtin_popcountll(x)
#define print2d(dp,n,m) for(int i=0;i<=n;i++){for(int
j=0;j<=m;j++)cout<<dp[i][j]<<" ";cout<<"\n";}
typedef std::numeric_limits< double > dbl;
using namespace __gnu_pbds;
using namespace std;
typedef tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> indexed_set;
//member functions :
//1. order_of_key(k) : number of elements strictly lesser than k
//2. find_by_order(k) : k-th element in the set
const long long N=200005, INF=2000000000000000000;
lld pi=3.1415926535897932;
```

```cpp
int lcm(int a, int b)
{
    int g=__gcd(a, b);
    return a/g*b;
}
int power(int a, int b, int p)
    {
        if(a==0)
        return 0;
        int res=1;
        a%=p;
        while(b>0)
        {
            if(b&1)
            res=(res*a)%p;
            b>>=1;
            a=(a*a)%p;
        }
        return res;
    }
void pr(int i, int ans)
{
    cout<<"Case #"<<i<<": "<<ans<<"\n";
}
void prv(int i, vi v)
{
    cout<<"Case #"<<i<<": "<<v.size()<<" ";
    for(auto num:v)
    cout<<num<<" ";
    cout<<"\n";
}
int32_t main()
{
    IOS;
    int t;
    cin>>t;
```

```cpp
for(int z=1;z<=t;z++)
{
    int d, n, k;
    cin>>d>>n>>k;
    vector <pii> v[d+2];
    rep(i,0,n)
    {
        int h, l, r;
        cin>>h>>l>>r;
        v[l].pb({h, 1});
        v[r+1].pb({h, 0});
    }
    mii mp1, mp2;
    int s=0, sum=0, ans=0;
    for(int i=1;i<=d;i++)
    {
        for(auto p:v[i])
        {
            int h=p.ff;
            if(p.ss)
            {
                if(s<k)
                {
                    s++;
                    mp1[h]++;
                    sum+=h;
                }
                else if((*mp1.begin()).ff < h)
                {
                    int val=(*mp1.begin()).ff;
                    mp1[val]--;
                    if(mp1[val]==0)
                    mp1.erase(val);
                    mp2[val]++;
                    mp1[h]++;
                    sum+=(h-val);
                }
                else
                mp2[h]++;
            }
            else
            {
                if((*mp1.begin()).ff<=h)
                {
                    mp1[h]--;
                    if(mp1[h]==0)
                    mp1.erase(h);
                    s--;
                    sum-=h;
                    if(mp2.size())
                    {
                        auto it=mp2.end();
                        it--;
                        int val=(*it).ff;
                        mp2[val]--;
                        if(mp2[val]==0)
                        mp2.erase(val);
                        mp1[val]++;
                        s++;
                        sum+=val;
                    }
                }
                else
                {
                    mp2[h]--;
                    if(mp2[h]==0)
                    mp2.erase(h);
                }
            }
        }
        ans=max(ans, sum);
    }
```

```
        pr(z, ans);
    }
}
```