Binary Search on Real Numbers

For doing binary search on real numbers, we need to add the tolerance limit to the search space or the precision limit.

Q1. Find the square root of an integer upto 6 places of decimal.

Most General Way:

```
int main()
{
    double low, high;
    int n;
    cin >> n;

    low = 1.0;
    high = n;

    /*
      if you want x digits of precision, make you epsilon of the order greater than
x just to be on the safer side
    */

    const double EPS = 1e-8; //since we want the precision to be of  order of
10^(-6) which is 2 more


    double ans = 1.00; // if n =1 , while loop will not be executed, to handle that
case initialize ans with 1

    while (high - low > EPS)
    {
        double mid = (low + high) / 2.0;
        if (mid * mid > n)
        {
            high = mid;
```

```
        }
        else
        {
            // mid*mid must be lesser than or equal to n
            // mid^2<=n
            // mid<=sqrt(n)
            // so my answer must be greater than or equal to mid
            ans = mid;
            low = mid;
        }
    }

    cout << fixed << setprecision(6) << ans << endl;
    return 0;
}
```

Constant Iterations Method: (Works almost everywhere, faster as well)

We run a loop a constant number of times (in our case 100).

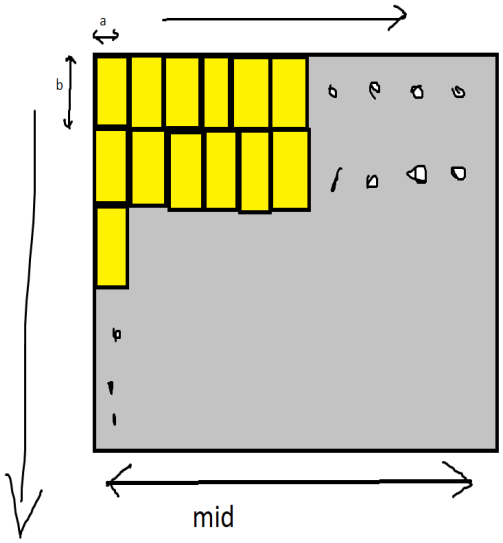
```
double ans = 1.00;
  for(int i=0;i<100;i++)
    {
        double mid = (low + high) / 2.0;
        if (mid * mid > n)
        {
            high = mid;
        }
        else
        {
            // mid*mid must be lesser than or equal to n
            // mid^2<=n
            // mid<=sqrt(n)
```

```
            // so my answer must be greater than or equal to mid
            ans = mid;
            low = mid;
        }
    }

    cout << fixed << setprecision(6) << ans << endl;
    return 0;
```

Time Complexity of Binary Search on real numbers: O(log(range/epsilon)) where range=high-low

2)Given n identical rectangles of size a×b. You need to find a square of the minimum area in which all n rectangles can be placed. Rectangles can touch along a point or side, but they cannot intersect. Rectangles cannot be rotated.

Note: a and b can be real numbers.



Max. number of rectangles in the first row=mid/a (floor value)

Each row can have this number of rectgls.

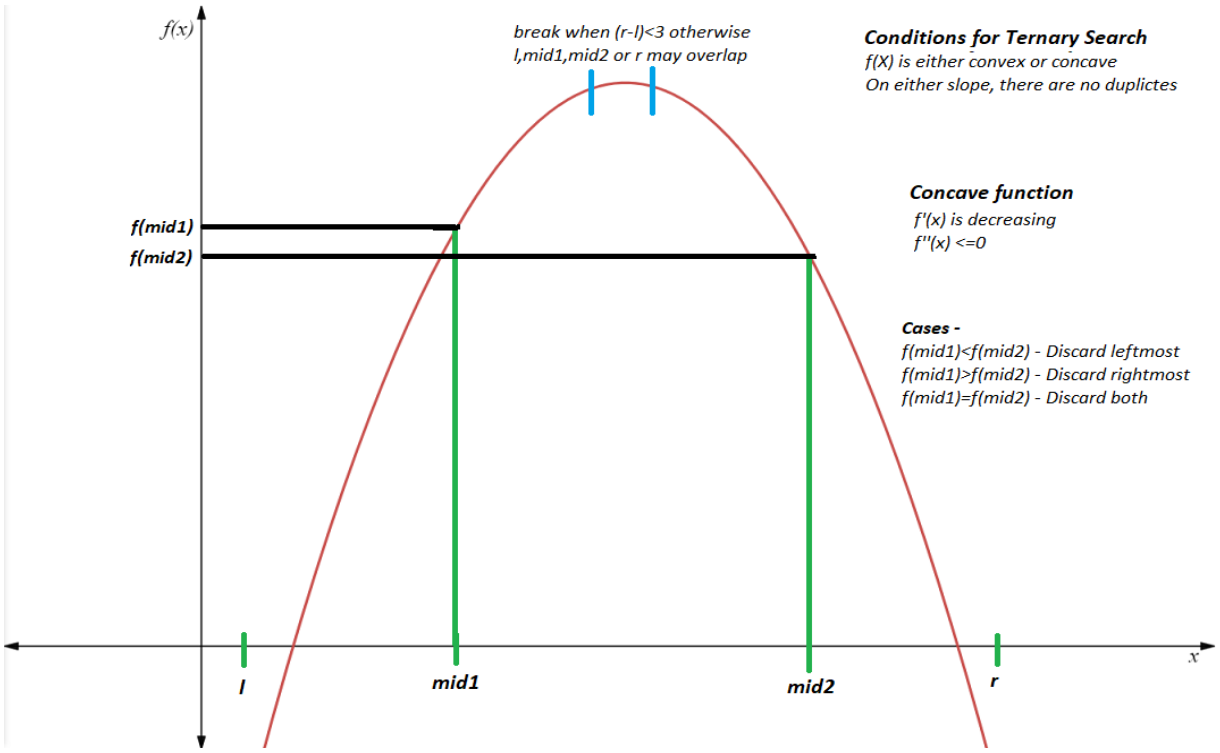
Max. number of rows=mid/b
(floor)

Total number of rectangle which can be accomaodated in square of side mid= Max number of rows * Max number of rectangles in each row
=(mid/b)*(mid/a)=(mid/a)*(mid/b)

```
double low=0.00;
  double high=n*(max(a,b));

  double ans=0.0;

  const double EPS=1e-9;

  while(high-low>EPS){
     double mid=(high+low)/2.00;

     if(floor(mid/a)*floor(mid/b)>=n){
         ans=mid;
         high=mid;
     }
     else
     low=mid;
  }

  double area=ans*ans;

  cout<<fixed<<setprecision(6)<<area<<endl;
```

Ternary Search

Concave function -

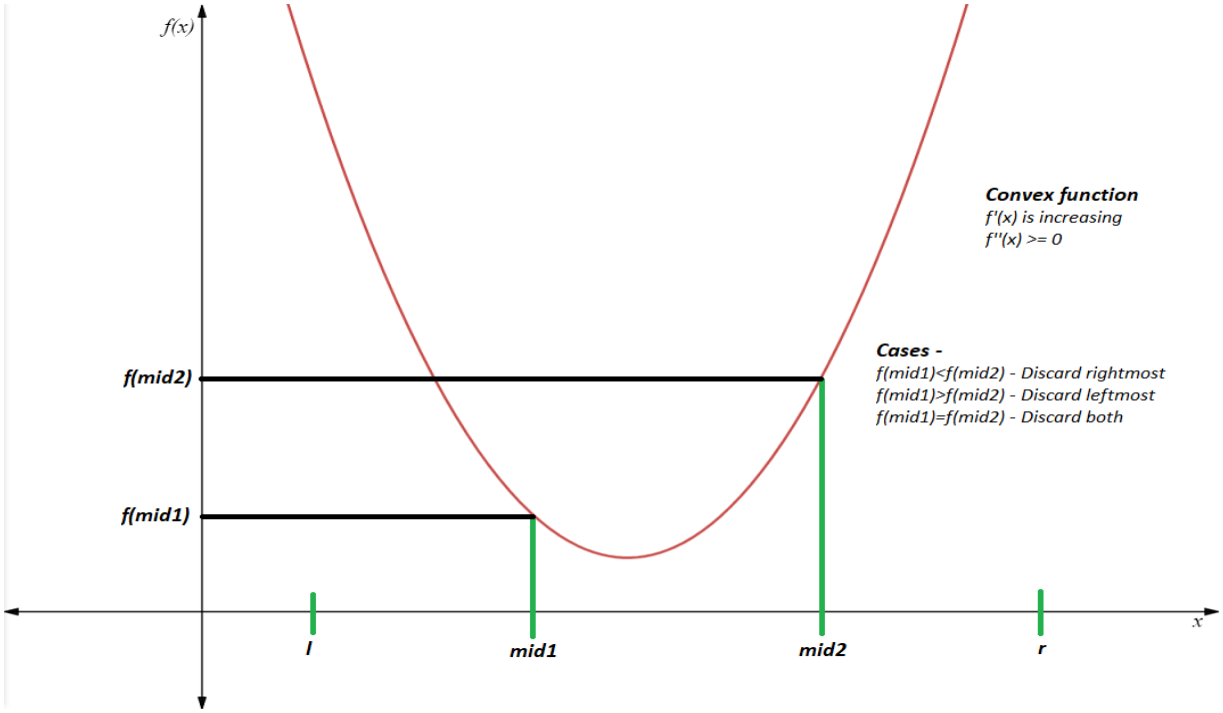


To find the Maxima

```
while(hi-lo>=3)
   {
      int mid1 = lo + (hi-lo)/3;
      int mid2 = hi - (hi-lo)/3;
      int f1 = f(mid1);
      int f2 = f(mid2);
      if(f1>f2)
      {
         hi = mid2;
      }
      else if(f1<f2)
      {
         lo = mid1;
```

```
        }
        else{
            lo = mid1;
            hi = mid2;
        }
    }
// Iterate from lo to hi (since hi-lo<3  now)

for(int i=lo;i<=hi;i++){
Ans = max(ans,f(i));
}
```

Convex function -



To find the minima

```
while(hi-lo>=3)
    {
        int mid1 = lo + (hi-lo)/3;
        int mid2 = hi - (hi-lo)/3;
        int f1 = f(mid1);
        int f2 = f(mid2);
        if(f1<f2)
```

```
        {
            hi = mid2;
        }
        else if(f1>f2)
        {
            lo = mid1;
        }
        else{
            lo = mid1;
            hi = mid2;
        }
    }
// Iterate from lo to hi (since hi-lo<3  now)

for(int i=lo;i<=hi;i++){
Ans = min(ans,f(i));
}
```

Time Complexity

Each time search space reduces from n->⅔(n)
Time complexity will be log(3/2)(n)

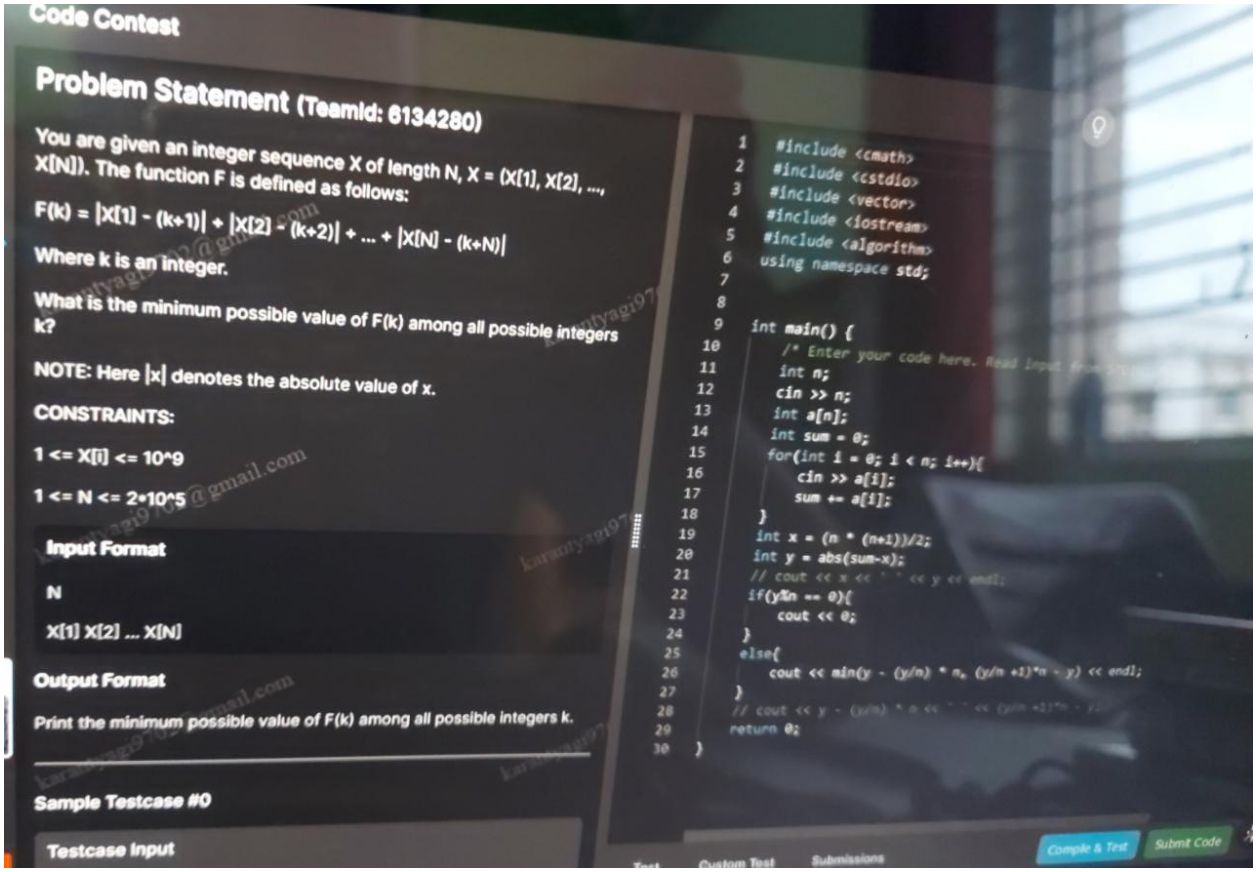
If you reduce the search space from n->(n/3)
In which you proceed with only one of the three parts,
Time Complexity will be log3(n)

Search using Ternary Search (O(log3n)):

```
At each Point make a decision to search in one of the three parts.
      If N is range to be searched, then at each point
      N -> N/3
      Overall Time Complexity O(log(3)N)
```

```cpp
int main(){

    int n;
    cin>>n;
    int k;
    cin>>k; // element to be searched
    int ar[n];
    for(int i=0;i<n;i++)
    cin>>ar[i];
    int lo  =0;
    int hi = n-1;
    while(hi-lo>=3)
    {
        int mid1 = lo + (hi-lo)/3;
        int mid2 = hi - (hi-lo)/3;
        if(k<ar[mid1])
        hi = mid1;
        else if(k>ar[mid2])
        lo = mid2;
        else {
            lo = mid1;
            hi  =mid2;
        }

    }
```

Questions




```
int f(int k,int x[], int n)
{
    int ans = 0;
    for(int i=1;i<=n;i++)
    ans+=abs(x[i]-(k+i));
    return ans;
}
```

```
int main(){

int n;
cin>>n;
int a[n+1];
for(int i=1;i<=n;i++)
cin>>a[i];
int lo = -1e10;
int high = 1e10;
    while(hi-lo>=3)
```

```
    {
        int mid1 = lo + (hi-lo)/3;
        int mid2 = hi - (hi-lo)/3;
        int f1 = f(mid1,ar,n);
        int f2 = f(mid2,ar,n);
        if(f1<f2)
        {
            hi = mid2;
        }
        else if(f1>f2)
        {
            lo = mid1;
        }
        else{
            lo = mid1;
            hi = mid2;
        }
    }
    int ans = INF;
    for(int i=lo;i<=hi;i++)
    {
        ans = min(ans,f(i,ar,n));
    }
cout<<ans<<"\n";
    return 0;
}
```

https://codeforces.com/problemset/problem/439/D

```cpp
#define int long long

int calca(int val, int a[], int n)
{
	int ans = 0;
	for(int i=0;i<n;i++)
	{
		ans+=max(0ll,val - a[i]);
	}
	return ans;
}
int calcb(int val, int b[], int m)
{
	int ans = 0;
	for(int i=0;i<m;i++)
	{
		ans+=max(0ll,b[i] - val);
	}
	return ans;
}


void solve()
{
	int n,m;
	cin>>n>>m;
	int a[n];
	int b[m];
	int i;
	for(int i=0;i<n;i++)
	cin>>a[i];
	for(int i=0;i<m;i++)
	cin>>b[i];
	int mn = 0;
	int mx = 1e9+5;
	while(mx-mn>2)
	{
		int mid1 = mn+(mx-mn)/3;
		int mid2 = mx - (mx-mn)/3;
		int f1 = calca(mid1,a,n)+calcb(mid1,b,m);
		int f2 = calca(mid2,a,n)+calcb(mid2,b,m);
		if(f1>f2)
		{
			mn = mid1;
		}
		else
		{
			mx = mid2;
		}

	}
	int ans = 1e18;
	for(int i=mn;i<=mx;i++)
	{
		ans = min(ans,calca(i,a,n)+calcb(i,b,m));
```

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

    }
```