



**INFORMATICS.
OLYMPIAD.CH**

INFORMATIK-OLYMPIADE
OLYMPIADES D'INFORMATIQUE
OLIMPIADI DELL'INFORMATICA

Binary Search

Sarnen Camp 2023

February 15, 2023

```
auto f = [&](int& x) { return ... }  
  
// invariant: f(l)==true, f(r)==false  
int l=0, r=2e9;  
while (r-l > 1) {  
    auto m = (l+r)/2;  
    if (f(m))  
        l = m;  
    else  
        r = m;  
}
```

Binary Search

```
// Input: [0,0,....,0,1,....,1,1]  
// Output: Index of last 0, or -1 if there is none  
int search(vector<int> const& a) {  
    // Invariant: l==-1 or a[l]=0  
    int l=-1, r=a.size();  
    while (r - l > 1) {  
        int m = l + (r - l)/2;  
        if (a[m] == 0)  
            l = m;  
        else  
            r = m;  
    }  
    return l;  
}
```

Implicit representation of array with predicates.

$$P(x) \wedge (x < y) \implies P(y)$$

Example: Find index of value x in array a_0, \dots, a_{n-1}

Implicit representation of array with predicates.

$$P(x) \wedge (x < y) \implies P(y)$$

Example: Find index of value x in array a_0, \dots, a_{n-1}

$$P(i) = (a[i] > x)$$

$$P(-1) = 0$$

Make sure that the list starts with 0 and ends with 1.

Add $-\infty$ to the front and ∞ to the end to avoid any problems.

Example 1 (lawfirm)

N employees

K filing cabinets that contain a_0, \dots, a_{K-1} documents

Assign employee i to consecutive cabinets l_i, \dots, r_i

All cabinets need to be covered by exactly one employee

Minimize $\max_i \{a_{l_i} + a_{l_i+1} + \dots + a_{r_i-1}\}$

Reduction to a simpler problem using binary search:

What is the minimum cost?



Can it be done with cost x ?

<https://ioinformatics.org/files/ioi2010problem3.pdf>

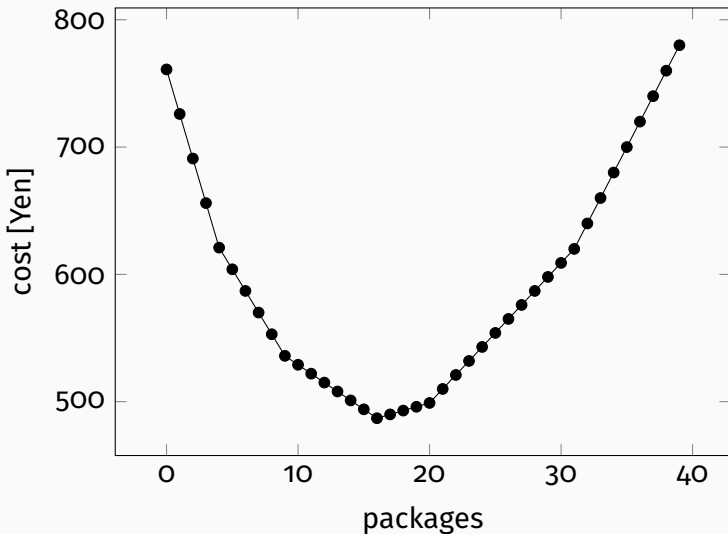
Cities in Alberta tend to be laid out as rectangular grids of blocks. Blocks are labeled with coordinates 0 to $R-1$ from north to south and 0 to $C-1$ from west to east.

The quality of living in each particular block has been ranked by a distinct number, called *quality rank*, between 1 and $R*C$, where 1 is the best and $R*C$ is the worst.

The city planning department wishes to identify a rectangular set of blocks with dimensions H from north to south and W from west to east, such that the median quality rank among all blocks in the rectangle is the best. H and W are *odd* numbers not exceeding R and C respectively. The *median quality rank* among an odd number of quality ranks is defined to be the quality rank m in the set such that the number of quality ranks better than m equals the number of quality ranks worse than m .

You are to implement a procedure **rectangle**(R,C,H,W,Q) where R and C represent the total size of the city, H and W represent the dimensions of the set of blocks, and Q is an array such that $Q[a][b]$ is the quality rank for the block labeled a from north to south and b from west to east.

Your implementation of **rectangle** must return a number: the best (numerically smallest) possible median quality rank of an H by W rectangle of blocks.



- Binary search on derivative
- Ternary search (never!)

Example 3

Given a sequence of n integers a_1, a_2, \dots, a_n .

Determine a real number x such that the weakness of the sequence $a_1 - x, a_2 - x, \dots, a_n - x$ is as small as possible.

Weakness: maximum value of the poorness over all segments (contiguous subsequences) of a sequence.

Poorness of a segment: absolute value of the sum of the elements of segment.

Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-6}

Example 3: Take Away

```
int search(double l, double r, double EPS) {  
    // Invariant: a[l]=0  
    while (r-l > EPS) {  
        double m = (l + r)/2;  
        if (pred(m))  
            l = m;  
        else  
            r = m;  
    }  
    return l;  
}
```

How many operations?

How many operations?

$$\mathcal{O}(\log((r - l)/e))$$

x : Solution

\tilde{x} : Approximation of x

Absolute error (difference to solution?):

$$|x - \tilde{x}|$$

Relative error (percentage of error?):

$$\frac{|x - \tilde{x}|}{|x|} = \left| \frac{\tilde{x}}{x} - 1 \right|$$


```
def IsApproximatelyEqual(x, y, epsilon=1e-6):  
    # Check absolute precision.  
    if -epsilon <= x - y <= epsilon:  
        return True  
  
    # Is x or y too close to zero?  
    if (-epsilon <= x <= epsilon or  
        -epsilon <= y <= epsilon):  
        return False  
  
    # Check relative precision.  
    return (-epsilon <= (x - y) / x <= epsilon or  
            -epsilon <= (x - y) / y <= epsilon)
```

Intuitive idea: Faster convergence for binary search on exponent

After some math:

$$m = \sqrt{lr} = e^{\frac{1}{2}(\log(l)+\log(r))}$$

Some pointer: <http://codeforces.com/blog/entry/49189>

Doubles are discrete. They can be casted to 64 bit integers.

Find the best double to approximate a problem: Binary search!

```
int search(double l, double r) {
    for (;;) {
        uint64_t l_bits = *(uint64_t*)&l;
        uint64_t r_bits = *(uint64_t*)&r;
        if (r_bits - l_bits <= 1)
            return l;
        uint64_t m_bits = l_bits+((r_bits-l_bits)>>1);
        double m = *(double*)&m_bits;
        if (pred(m))
            l = m;
        else
            r = m;
    }
}
```

Minimize absolute *and* relative error?

- Absolute error at most EPS if $x < 1$.
- Relative error at most EPS if $x > 1$.

Absolute error:

$$|x - \tilde{x}|$$

Relative error:

$$\frac{|x - \tilde{x}|}{|x|} = \left| \frac{\tilde{x}}{x} - 1 \right|$$

Sometimes, there is no upper bound or you want your solution to be in $\mathcal{O}(\log(ans))$.

```
int r = 1;
while (!pred(r))
    r *= 2;
return search(1, r);
```

Can visualize binary search as a tree.

We look for the value that minimizes $(\text{Pred}(x), -x)$.

Branch and Bound: Define LB and UB => exactly the same as binary search.