Codeforces Edu: Binary search

General binary search

```
// format: FFFFFFVVVVVVV
// swap r = m / l = m if VVVVFFFF

// set as needed
int l = 0, r = n
// if you don't know r
while (!ok(r)) r *= 2;

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

// resulting pair will be ok(l) = F, ok(r) = V
```

Working with real numbers

It's preferable to, instead of writing `while (r - l > 1)`, use a fixed amount of iterations. 100 is a good *magic number* to get 10^{-9} precision, but formally it's $\frac{maxAns}{\epsilon}$, where $maxAns$ is the maximum possible answer and ϵ is the desired precision.

```
double l = 0, r = n;

for (int i = 0; i < 100; i++) {
    double m = (r - l) / 2 + l;
    if (ok(m)) r = m; // remember to make ok accept doubles!!
    else l = m;
}
```

Minimax problems

This happens when you wanna get the minimum value V, when V is obtained by the maximum of some things. Either min(max(...)) or the max(min(...)). Example:

> Obtain the path with minimum value M on a graph, where M is the maximum edge weight in that path.

For this, one must obtain:

$min_M(M = max(\cdots))$

One could pose an `ok(x)` that checks if $M \leq x$. If $M \leq x$, then $M \leq x + k$. So we have a `FFFFVVVVVV` behavior and could binary search the first `V`.

Then, one must ask, what does it mean that $max(\cdots) \leq x$? Many times it means that all elements involved in that max are $\leq x$. This could be a way to simplify the calculation.

Maximum average

This happens when one has a set of elements A, and wants to pick the subset S with maximum (or minimum) average:

$$\frac{\sum_{i \in S} a_i}{|S|} \to \max$$

We could pose an `ok(x)` that checks if

$$\frac{\sum_{i \in S} a_i}{|S|} \geq x$$

Note that if this is true for x, it's true for $x - h$, so this follows format `VVVVFFFFFF` and we could binary search the last `V`, which would represent the maximum average. It's important to treat x as a real number in our binary search.

To make an efficient problem, let's simplify this:

$$\frac{\sum_{i \in S} a_i}{|S|} \geq x$$

$$\sum_{i \in S} a_i \geq x\,|S|$$

$$\sum_{i \in S} a_i - x\,|S| \geq 0$$

$$\sum_{i \in S} a_i - \sum_{i \in S} x \geq 0$$

$$\sum_{i \in S} (a_i - x) \geq 0$$

So we can check if this sum renders positive for any subset.

- If we're checking for segments in an array, we could turn this into a prefix sum `p` and calculate `p[r] - p[l-1]`
- If these segments should be $\geq D$ in size, we could make a prefix minimum array from `p`; check the article at this point lol
- If we're checking for paths in a graph, you could do a modified DFS that tracks if any path's sum renders positive. This is $O(2^n)$ though
  - Particularly if it's a DAG, we could start off at its "root" and calculate sequentially the minimum sum for each node, similar to a topological sort kind of search. Then find the minimum sum at the end node that we're checking (or at all nodes)

K-th element of a sequence

This happens when one wants to find the k-th element of an **ascending** (or descending) sequence or list, that is obtained through some odd method.

We could consider the function `cnt(x)` which returns the amount of elements in that list that are $< x$.

Then, we can note something: if we are searching for the element at `k = 4` on a *0-indexed* list like this:

```
1 2 3 3 3 4 4 5 6
      ^
```

Then the amount of numbers that are less than 3 is $\leq k$, and the amount of numbers that are less than 4 is $> k$. Thus, we want to get the maximum element such that

$$cnt(x) \leq k$$

So we can apply binary search to find x. The other part of the problem will be finding how to calculate `cnt(x)`.