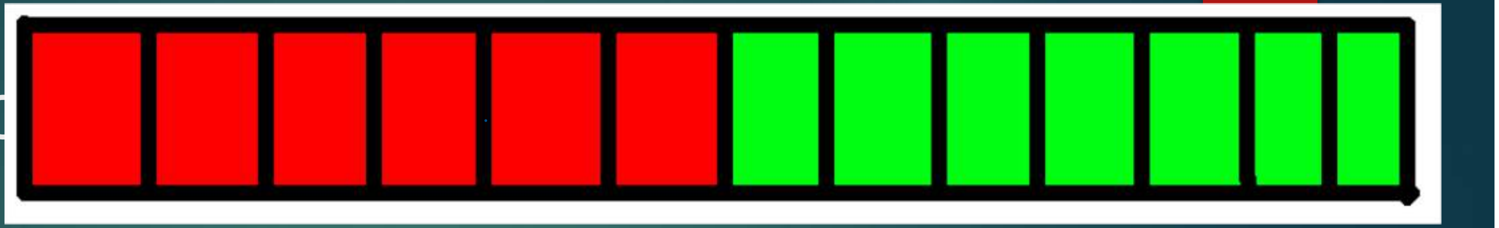
Binary Search

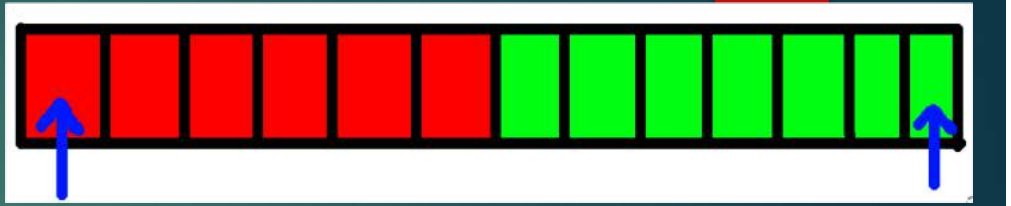
SUMMER OF COMPETITIVE PROGRAMMING
ALGORITHMS AND CODING CLUB IIT DELHI

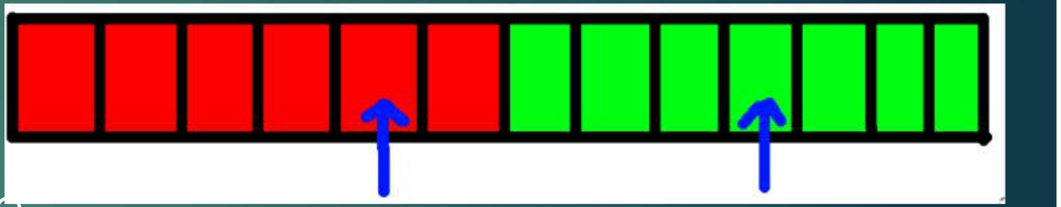
Binary Searc



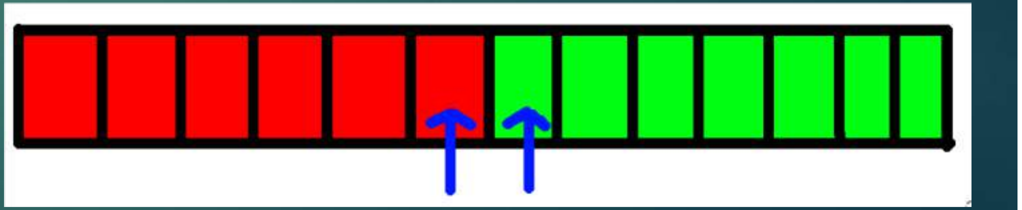
- Some boolean function f(x) on integers - "predicate"
- Monotone: is false for a while then true forever
  - Or vice versa
- Want to find switchover point
- When searching for x in array we use f(i) = a[i]<=x

Invariant Idea





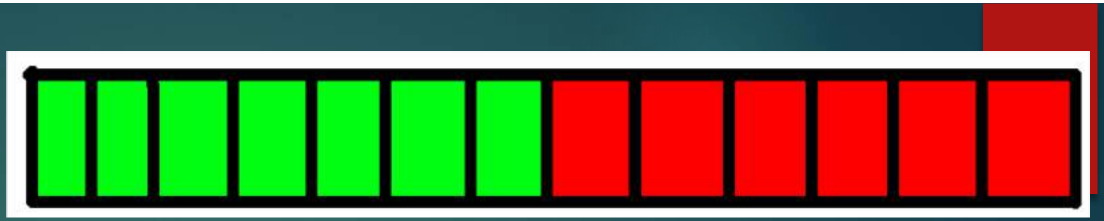
- Idea: maintain two pointers
- One of them is true, other false
    - This condition always holds
- Keep shrinking till the border



Pseudocode

```
int l = -100000 // always red
int r = 100000 // always green

while r-l > 1:
  int m = (r+l)/2
  if works(m): r=m
  else: l=m
```



Important Points

- Since L/R >1 apart, it always keeps shrinking
    - Never worry about end infinite looping/off by ones
- Can do same thing but with L green, R red
    - Have to flip the if statement
- Original L/R never get called
    - If it's possible to be all green/red, u can just use out of bounds value for original L/R

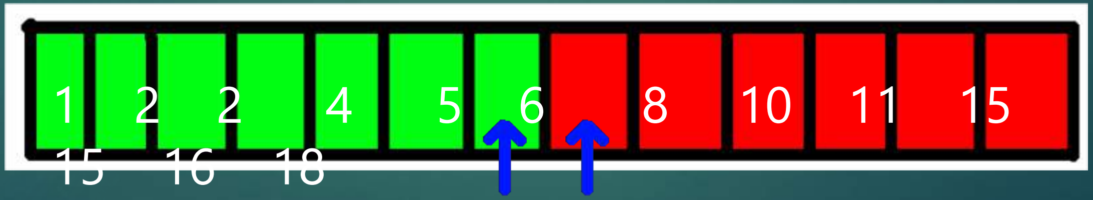
Time Complexity

- We shrink either L or R to the midpoint
- Range cuts in half every time
- O(log(n)) time complexity
- This is very good
  - On range of size million, log(million)=20

Example: Search in array

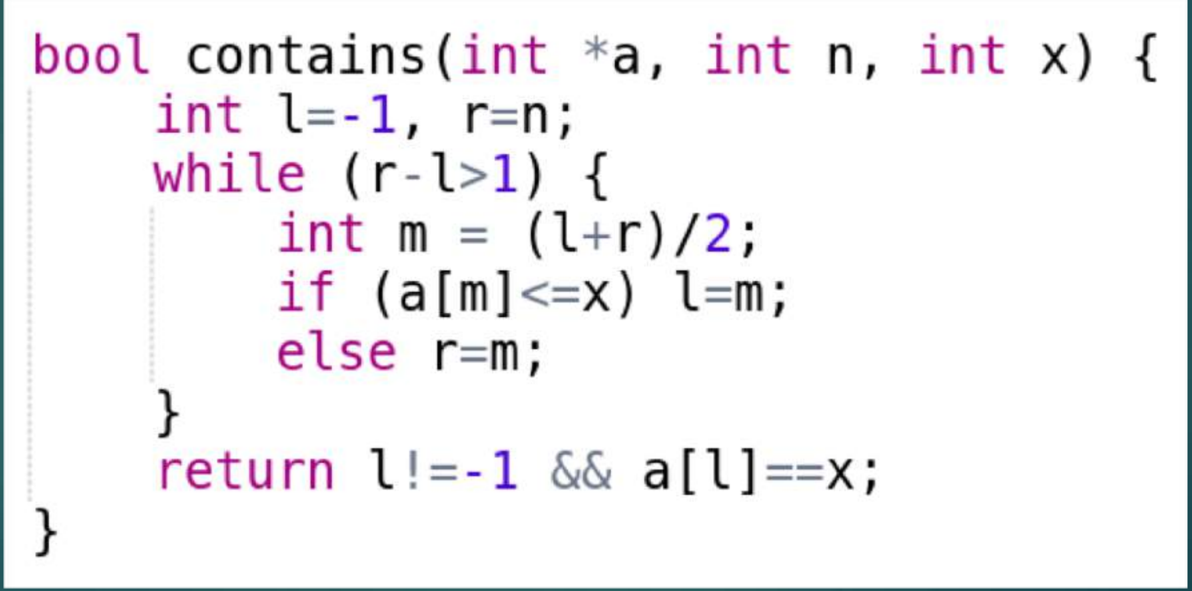
- We'll make predicate be a[i]<=x
- Then, L will be the last thing <=x
  - R will be the first thing >x
- We need to check if a[l]==x

Search for 8:



1 2 2 4 5 6 8 10 11 15
15 16 18

Code for array search

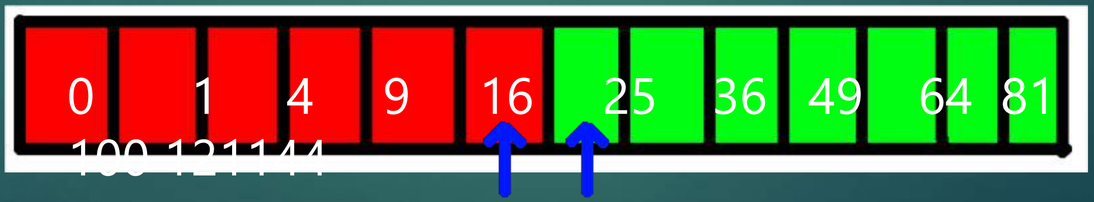


```
bool contains(int *a, int n, int x) {
    int l=-1, r=n;
    while (r-l>1) {
        int m = (l+r)/2;
        if (a[m]<=x) l=m;
        else r=m;
    }
    return l!=-1 && a[l]==x;
}
```

Example: Square Root

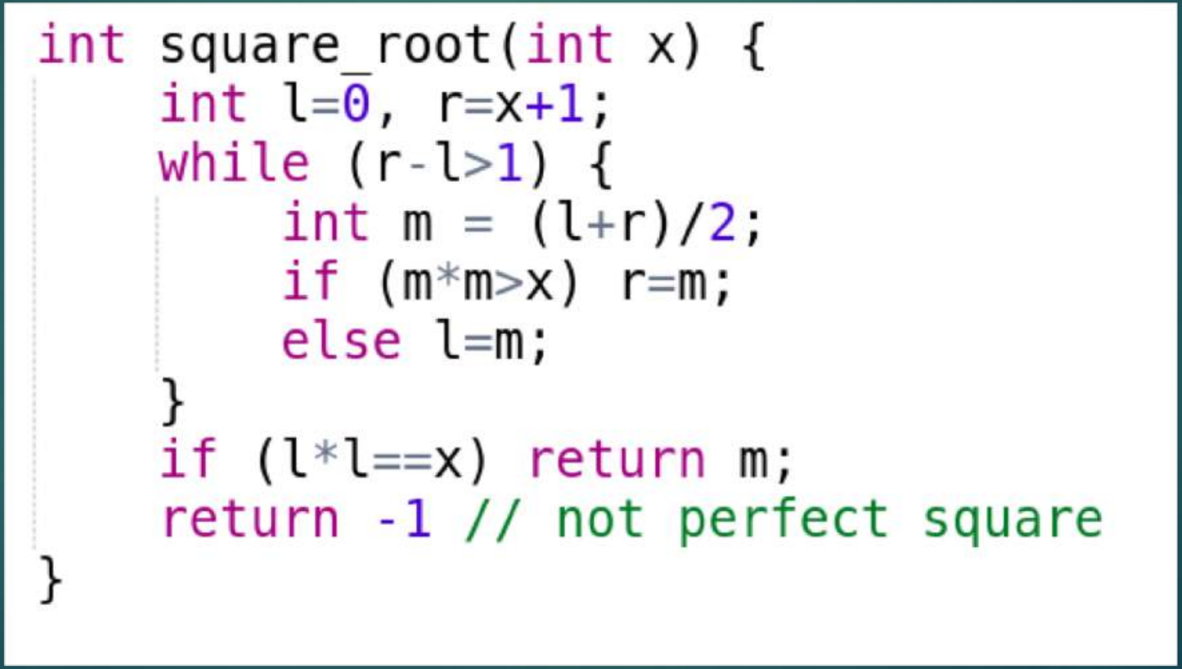
- We'll make predicate be m*m>x
- Then, R will be first thing with square >x
  - L will be last with square <=x

Search for 25:



0 1 4 9 16 25 36 49 64 81 100 121144

Code for square root



```
int square_root(int x) {
    int l=0, r=x+1;
    while (r-l>1) {
        int m = (l+r)/2;
        if (m*m>x) r=m;
        else l=m;
    }
    if (l*l==x) return m;
    return -1 // not perfect square
}
```

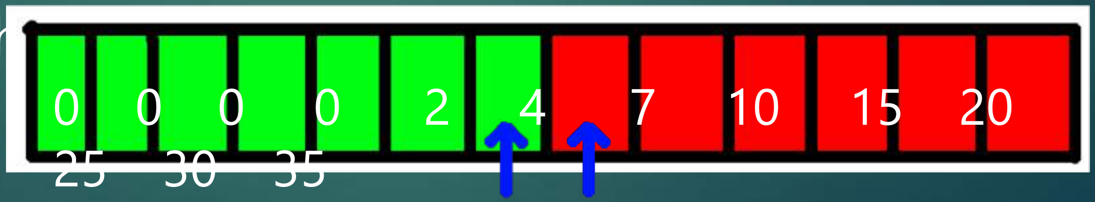
Problem: Magic Powder

- Cookie recipe with N(<=1e5) ingredients
- Need a[i] (<=1e9) grams of the i^{th} ingredient per cookie
- Have b[i] (<=1e9) grams of the i^{th} ingredient at home
- Have k (<=1e9) grams of magic powder
  - This can substitute any other ingredient
- How many cookies can you make
- Ex: a=[2,1,4]; b=[11,3,16]; k=1 → 4 cookies (powder on 2nd ingredient)

Magic Powder - Solution

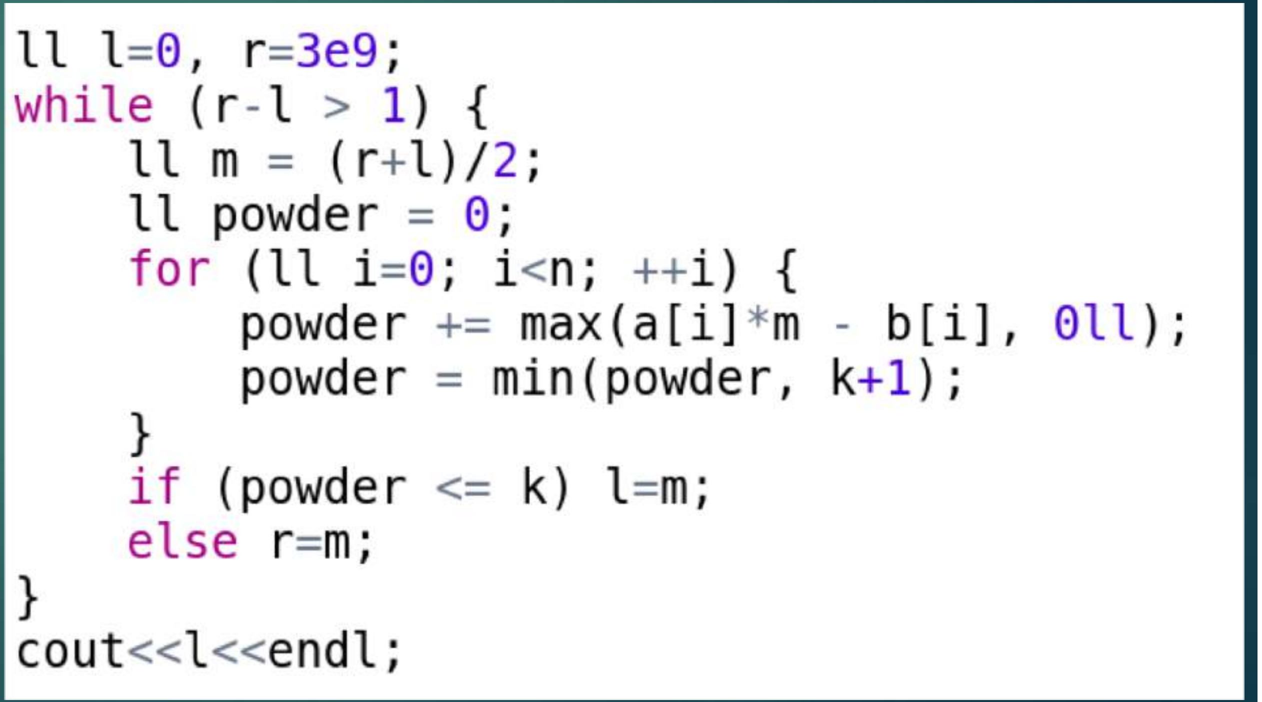
- Binary search on answer
- If we make m cookies, we know how much magic powder we'll use
- Predicate is if we have that much powder
- Then, L will be th

Ex: (k=8)



0 0 0 0 2 4 7 10 15 20 25 30 35

Magic Powder - Code



```
ll l=0, r=3e9;
while (r-l > 1) {
    ll m = (r+l)/2;
    ll powder = 0;
    for (ll i=0; i<n; ++i) {
        powder += max(a[i]*m - b[i], 0ll);
        powder = min(powder, k+1);
    }
    if (powder <= k) l=m;
    else r=m;
}
cout<<l<<endl;
```