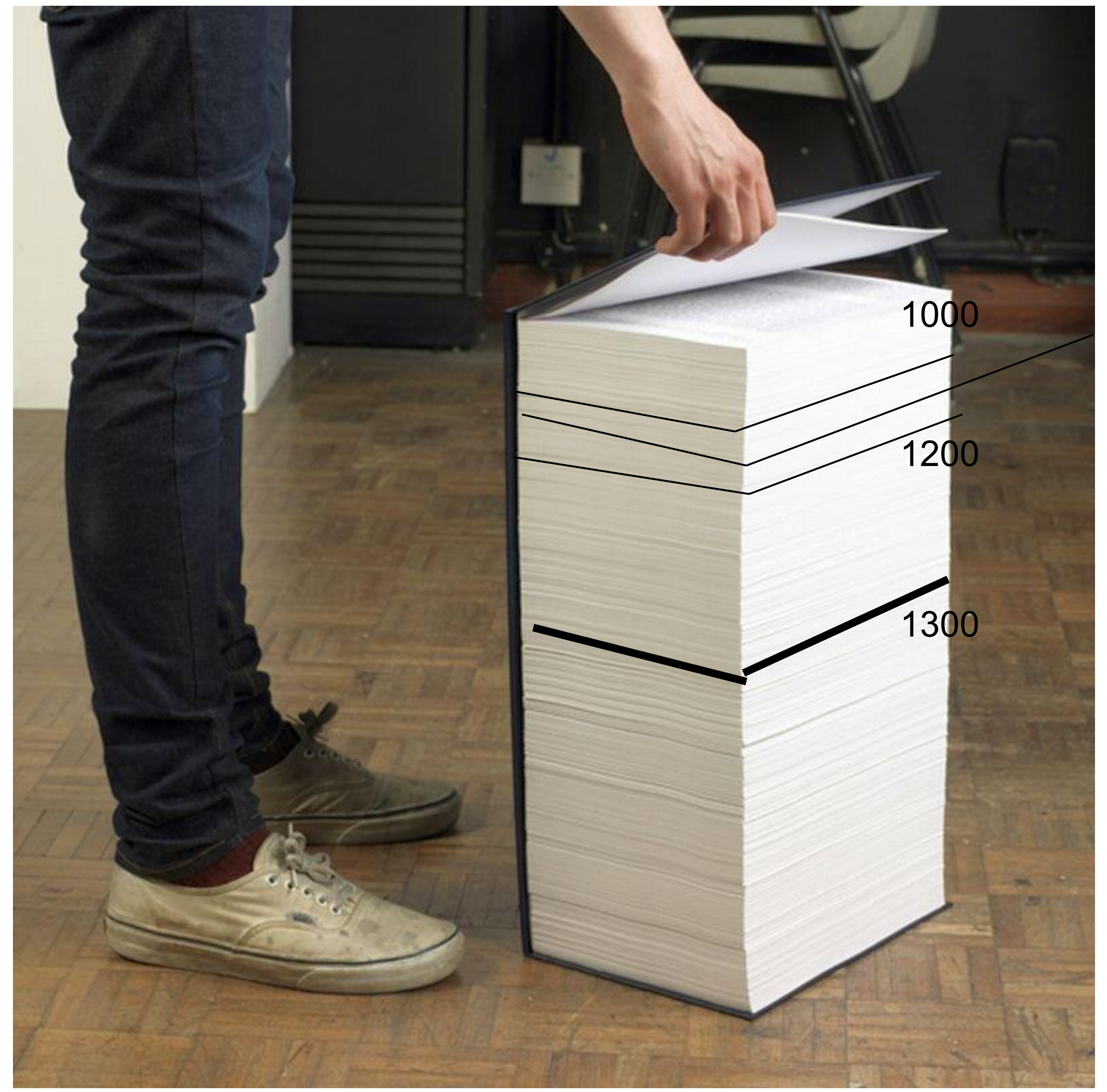
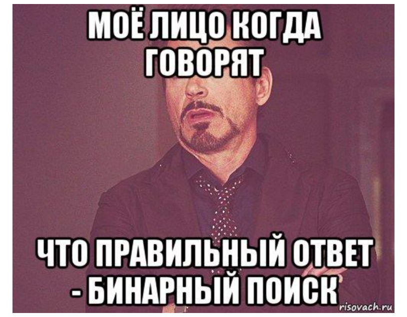
Binary search only for ordered array





torn out pages

find out if the book has a specific page 1147

10000 pages
2^10=1024
2^14=16…..
14 steps

aaabb
aaabc
aaad

facebook
2 000 000 000 users
2*1000^3~2*(2^10)^3=
=2*2^30=2^31
31 steps

last=5 arr[5]=20
first=4 arr[4]=10
target=15
middle=4

last=2 arr[2]=15
arr[1]=12
first=0 arr[0=10
target=15
middle=1

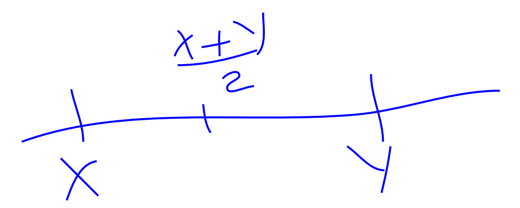
the best student is not the one who has learned everything. But who can invent on the go what he doesn't know

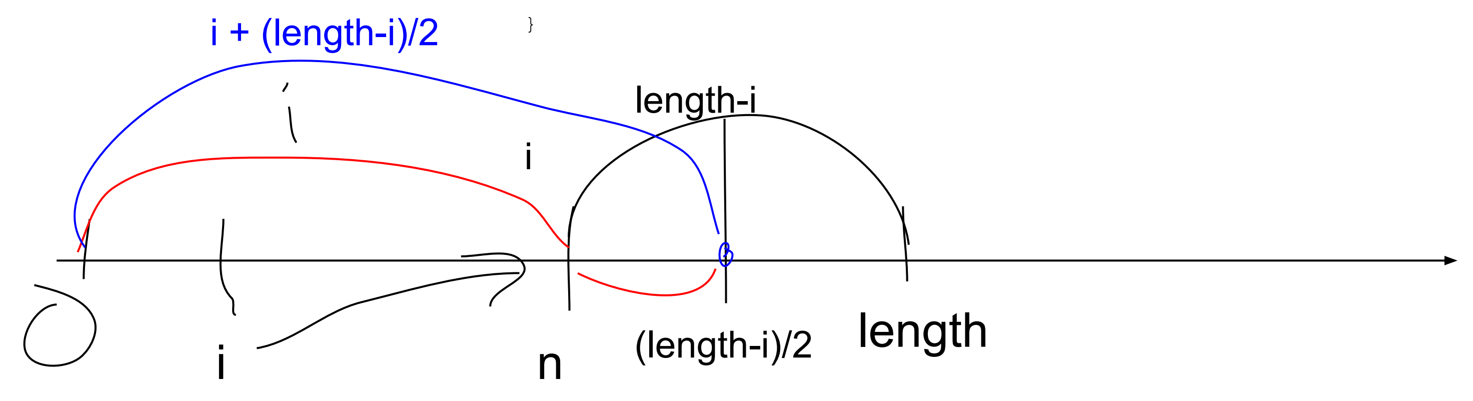
```
void binary_search(int arr[],int length,int target)
{
        if(...)
                cout<<"ok'<<endl;
        else
                cout<<"ok'<<endl;
}

int main()
{
        int arr[15]={};//zeros
        length=15;
        get_rand_arr_less_100(arr,length);
        print(arr,length);
        binary_search(arr,length,179);
}
```

```
void binarySearch(int arr[], int length, int target) {
  int first = 0;
  int last = length  - 1;
  int middle;
  int flag = 0;
  while (last > (first + 1)) {
     middle = (first + last) / 2;    // binary search needs middle part, with variables so it can change
     if (arr[middle] < target) {          // if array # in middle less than target
        first = middle;              // change range
     }
     else if (arr[middle] > target) {
        last = middle;
     }
     else if (arr[middle] == target) {
        flag = 1;
        break;
     }
  }
  if (flag == 0) {
     if (target == arr[first]) {          // need to check that the new 'first' and 'last variables
     // did not get too close without one ending up as target, the loop will not find them if they're next to each other
     // this second chepoint just sees if one of these is the target
        std::cout << "Number found at index: " << first << std::endl;
     }
     else if (target == arr[last]) {
        std::cout << "Number found at index: " << last << std::endl;
     }
     else {
        std::cout << "Number is not found in array." << std::endl;
     }
  }
  else if (flag == 1) {
     std::cout << "Number found at index: " << middle << std::endl;
  }
}
```

```
void binarySearch2(int arr[], int length, int target) {
  int n = length / 2;
  int i = 0;
  while (arr[n] != target && (length - 1) > 1) { // loop goes while target not found & while going through length of array
     if (arr[n] < target) {
        i = n;          // i is lower boundary , n is always the 'middle'
        n = (length - i) / 2 + i; // the middle is (same as previous program):(length of array + i) / 2
     }
     else if (arr[n] > target) {
        length = n;          // length is the top boundary
        n = (length - i) / 2 + i;
     }
  }
  if (arr[n] == target) {
     std::cout << "Number found: " << n << std::endl;
  }
  else {
     std::cout << "Not found. " << std::endl;
  }
}
```




i + (length-i)/2
length-i
i
0
i
n
(length-i)/2
length