

BINARY SEARCH & BISECTION

Intro Competitive Programming
Rodney Newell & Braddock Levit

C++

FINAL PROJECT

What is Binary Search, Bisection?

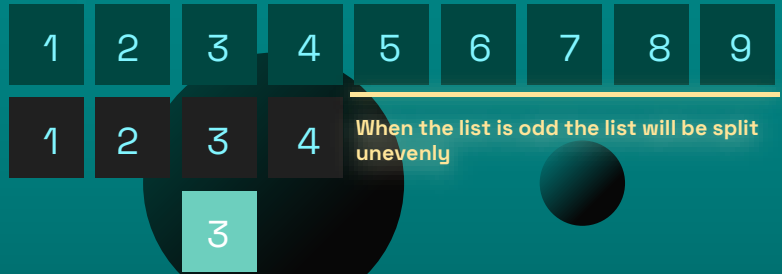
Binary Search in “English”:

Splitting a sorted list in half and looking through those parts of the list until the number you are looking for is found. For example, in the list 1-9 on the right, if you're looking for 3, you'd split the list in half and only consider the elements below the middle number as it's clear that 5 and above cannot include 3. This is done until 3 is found.

Computers have the ability to do this more efficiently than looking through a given list linearly, which is why Binary Search is very useful when you want to conserve time.

TARGET: 3

EXPLANATION AND SUMMARY

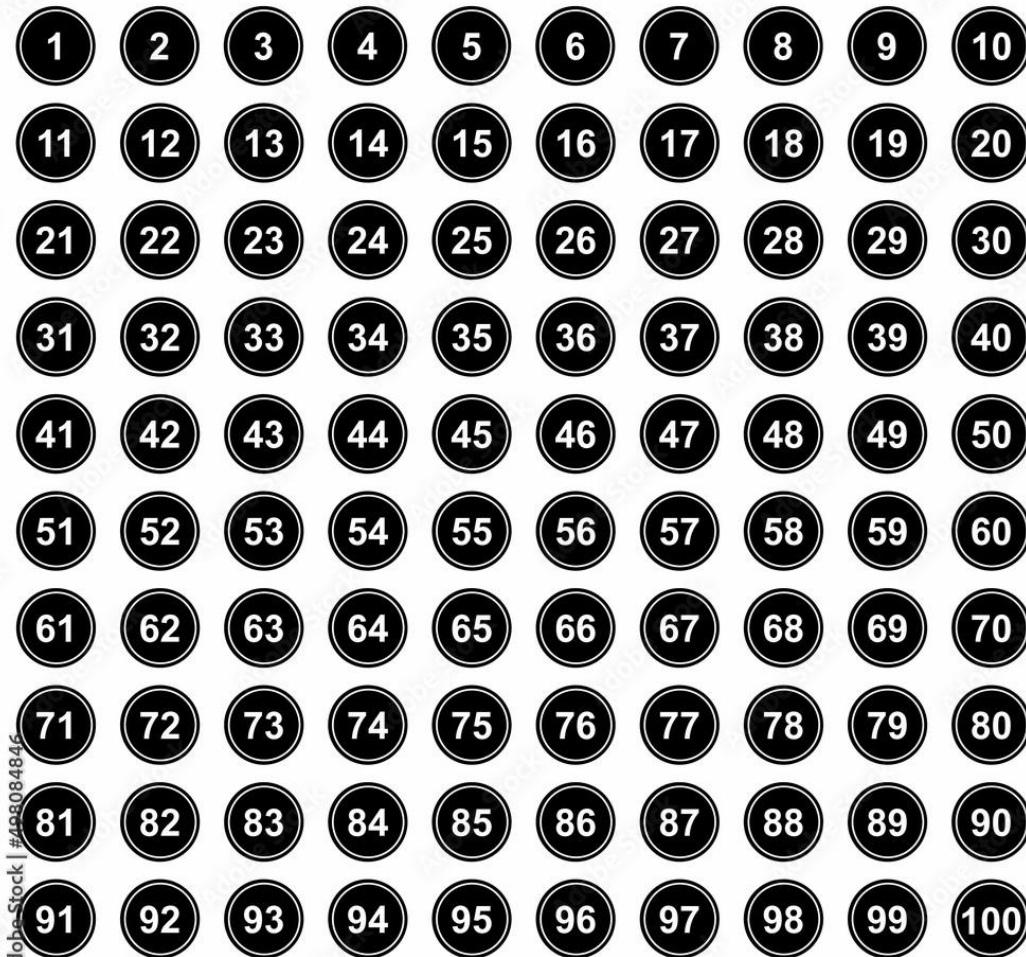


BINARY SEARCH

An algorithm used to find a target within an array (list of integers or doubles) with efficiency in mind. This is done by splitting the list in half and looking through each part to see whether the target element is found.

BISECTION

A technique similar to binary search, but instead of looking at a list of sorted numbers, the computer is finding the root of a function.



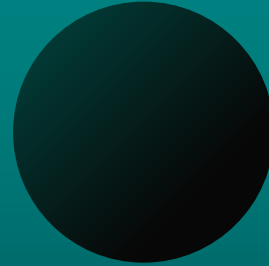
15

1	1	3	12	16	17	21	21	25	32
42	45	47	47	50	52	65	67	67	70
71	72	75	81	89	90	92	95	97	97
99	101	107	119	121	121	125	132	134	154
154	157	160	163	167	172	173	189	194	201
207	211	214	221	225	227	234	257	261	277
281	290	295	309	311	315	324	354	365	371
375	381	396	399	415	417	426	445	445	451
476	482	484	486	498	507	511	512	515	522
525	534	534	555	555	568	574	593	593	601

Uses of Binary Search

The main purpose for binary search is to look for a certain element in a list of sorted numbers. The reason is because of time limit. You see, computers can only look at one element at a time. By checking if an element is in a list by going through every single item, it would take a long time: $O(n)$. However, by using binary search, the run-time is only $O(\log(n))$. That means if you're looking through every item in a list with 1,000,000 items in it, it would take one million times. With binary search, it would only take 20!

Not only is binary search used for searching through a list of sorted items, but it also has many other uses as well, like bisection, which is finding the root of a function. This is helpful in jobs like construction or architecture, where math and coding is necessary for keeping structures safe.



Real-Life Applications of Binary Search

Binary Search can be used in real-life scenarios. For example, it might be implemented at a bank. Let's say a bank gives every individual a number as their account by the order they came in (ex: 1st person's account is 1, 2nd person's account 2, 3rd person's account is 3, etc). So, to keep up with modern-day developments, they've decided to implement an online system that keeps track of all of the users and the current users online. When someone logs in to the system, binary search can be used to see if their account number is a valid number and if they are not already on the system.

To do this, you would use a list of all the users and the users on the website. Then, using binary search, check if the number is valid by looking through the list of all the numbers. If the number is in that list, check if the number is in the list of current users. If it's not in this list, then the user can successfully log on.



Kattis Problem: Bootstrapping Number

EXAMPLE

Bootstrapping makes use of Binary Search in a function called *binarySearch()*.

In this problem we need to find a variable x , where $x^x = n$. To do this, we used binary search to shorten the range of the possible numbers x can be. This is done till we get into decimals, which is how we can find real specific numbers.

Unique Qualities of These Topics:

```
#include <bits/stdc++.h>
using namespace std;

double binarySearch(double n) {
    double low=1, high=20;
    for (int i=1; i<60; i++) {
        double mid = (low+high)/2;
        if (n < pow(mid,mid))
            high=mid;
        else
            low=mid;
    }
    return low;
}

int main() {
    double n;
    cin >> n;

    n = binarySearch(n);
    cout << fixed << setprecision(10) << n << endl;
}
```

Kattis Problem: H-Index

EXAMPLE

Given a list of numbers, find the biggest index that is smaller than or equal to its corresponding number.

Ex:

Numbers: 1, 1, 3, 5, 7

Indices: 5, 4, 3, 2, 1



```
using namespace std;
#include <bits/stdc++.h>
```

```
int binarysearch(vector<int> v){
    int low = 0;
    int high = v.size()-1;
    while (low<high) {
        int mid = (high+low)/2;
        if (v[mid]>=v.size()-mid) {
            high = mid;
        }
        else {
            low = mid+1;
        }
    }
    if (v[low]==0) {
        return 0;
    }
    return v.size()-low;
}
```

```
int main() {
    int num;
    cin >> num;
    vector<int> citations(num);
    for (int i=0;i<num;i++) {
        cin >> citations[i];
    }
    sort(citations.begin(),citations.end());
    cout << binarysearch(citations) << endl;
    return 0;
}
```

The Importance of Binary Search

To summarize, binary search is a useful process that shortens the time it takes to find an item in a list. It can be used anywhere in the modern-day world, helping computers run more smoothly.

