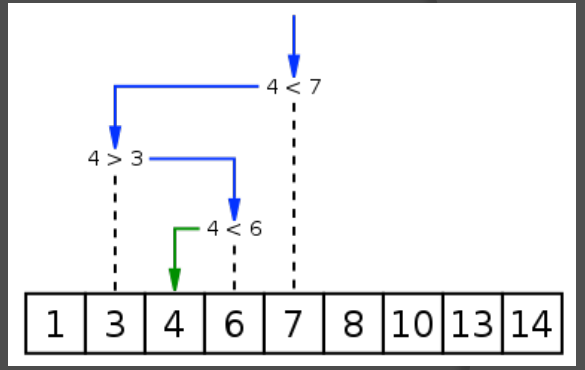
Advanced Computer Contest Preparation

Lecture 11

BINARY SEARCH

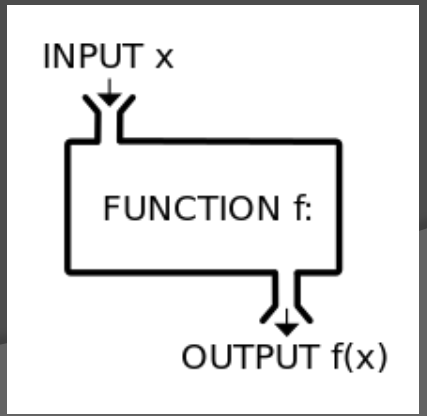
Recall: Binary Search

- What have we already learned about binary searches?
- We can perform them on **sorted** arrays
- We check the middle element of our search range
- If our target is less, our search range becomes the left half
- If our target is greater, our search range becomes the right half
- Otherwise, we have found our target



Recall: Binary Search

- What have we not learned about binary search?
- We can use binary search on things other than arrays!
- We primarily binary search on either arrays or **functions**



Binary Search on Functions – Example

- What is an example of a function $f(x)$ that we can binary search on?
- Sub-problem from DMOPC '16 November (Zeros):
  - What is the first factorial ($x!$) that contains at least N trailing zeros?
- What would $f(x)$ represent?
  - $f(x)$ represents the number of trailing zeros $x!$ has

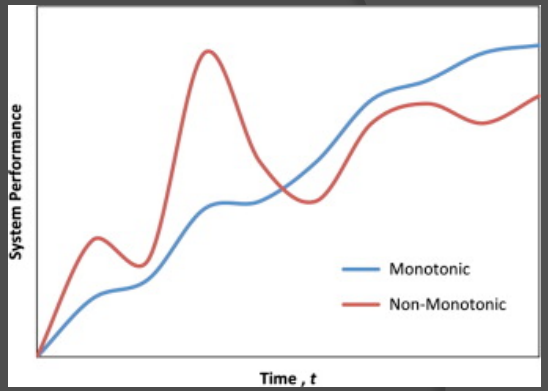
n	n!
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39,916,800
12	479,001,600
13	6,227,020,800
14	87,178,291,200
15	1,307,674,368,000
16	20,922,789,888,000

Built-in Binary Search Methods

- The built-in binary search methods are very useful when binary searching on arrays
- When we binary search functions, the upper limit of a function can be very high (could be 10^{18} or higher)
- We cannot use built in binary search methods since we cannot have an iterator that can access many (e.g. 10^{18}) elements
- Thus, we must implement our own binary search

Criteria for Binary Searching

- The primary requirement for a function $f(x)$ to be binary searchable is that $f(x)$ is **monotonic**
  - $f(x)$ is either non-increasing or non-decreasing
- If $f(x)$ only has two values (boolean), one side should all be `true` and the other side `false`
  - Note that this satisfies a monotonic property



0	0	0	1	1	1	1	1	1

Lower Bound

- The lower bound binary search method should return the first x such that $f(x)$ **does not compare less** than the target
  - **Greater or equal** to the target

Lower Bound – Pseudocode

```
lo = 0, hi = N, target;
while(lo < hi){
      mid = (lo + hi)/2;
      if (f(mid) >= target)
            hi = mid;
      else
            lo = mid+1;
}
print(lo);
```

Upper Bound

- The upper bound binary search method should return the first x such that $f(x)$ is **strictly greater** than the target
  - **Not less than or equal** to the target

Upper Bound – Pseudocode

```
lo = 0, hi = N, target;
while(lo < hi){
     mid = (lo + hi)/2;
     if (f(mid) > target)
          hi = mid;
     else
          lo = mid+1;
}
print(lo);
```

Out of Bounds Tip

- Suppose all values of $f(x)$ are less than our target value
- After a binary search, lo will be equal to the initial hi value
- Can we check if $f(lo)$ is less than our target value without computing $f(lo)$?
- Set initial hi value to 1 more than highest value (i.e. $N+1$)
- Initial hi value is known to be an "invalid" answer
- Initial hi value will never be used to compute answer
  - hi will never increase
  - When $lo < hi$, mid will always be less than hi
  - When $lo = hi$, the binary search is finished
- To check if all values of $f(x)$ are less than our target value, simply check if lo is equal to the initial hi after the binary search

Alternate Mid Calculation

- Some sources tell you to use

  ```
  mid = lo + (hi-lo)/2
  ```

  instead of

  ```
  mid = (lo+hi)/2
  ```

- Reason is due to potential integer overflow error of the second version
- Integer overflow error is not common unless limits of *hi* approach your data type limit
  - Since during computation of *mid*, *lo* can equal *hi-1*, overflow can happen if *hi >= limit/2*
- Both versions will produce the same values otherwise

Binary Search Example: Zeros

- Problem: What is the first factorial ($x!$) that contains at least N trailing zeros?
- What would $f(x)$ represent?
  - $f(x)$ represents the number of trailing zeros $x!$ has

Binary Search Example: Zeros

- Problem: What is the first factorial ($x!$) that contains at least N trailing zeros?
- How can we compute $f(x)$?
  - Note that we have an additional trailing zero for each factor of 10 we can form
  - Since $10 = 2*5$ and 2 appears as a factor more often, we want to count the number of 5s that appear as a factor from 1 to x
  - Thus, $f(x) = \sum_{i=1}^{\infty} \left\lfloor \frac{x}{5^i} \right\rfloor$

Floor function (round down)

Binary Search Example: Zeros

- Use lower bound method to find the first value of x such that $f(x)$ is at least N
- Example: if $N = 10$

lo = 1, hi = 100
mid = (1+100)/2 = 50
f(50) = 10+2 = 12 >= 10
hi = mid = 50
mid = (1+50)/2 = 25
f(25) = 5+1 = 6 < 10
lo = mid+1 = 26
mid = (26+50)/2 = 38
f(38) = 7+1 = 8 < 10
lo = mid+1 = 39
mid = (39+50)/2 = 44
f(44) = 8+1 = 9 < 10
lo = mid+1 = 45
mid = (45+50)/2 = 47
f(47) = 9+1 = 10 <= 10
hi = mid = 47
mid = (45+47)/2 = 46
f(46) = 9+1 = 10 <= 10
hi = mid = 46
mid = (45+46)/2 = 45
f(45) = 9+1 = 10 <= 10
hi = mid = 45
Since *lo = hi*, the answer is *45*.

Binary Search Analysis

- The binary search itself is $O(\log N)$
- At each step of the binary search, we need to run our function $f(x)$, which has a time complexity of $O(g(x))$, $g(x)$ is some function
  - The true time complexity to compute $f(x)$ depends on nature of the function
- The total time complexity is $O(g(x) \log N)$

Challenges

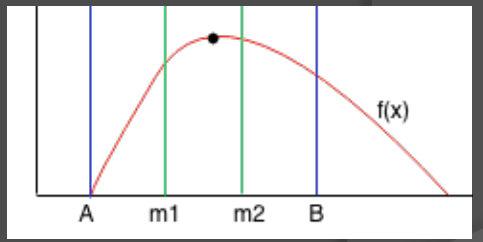
- Binary search problems have some challenges; here are a list of some of them:
  - Realizing whether a binary search can be used
  - Finding out what the function $f(x)$ should represent and how it should be computed
  - Off by one errors; could possibly result in infinite loops if code is incorrect

Extension to Real Numbers

- Sometimes, the domain of $f(x)$ includes numbers that are not integers
- Binary search as normal, but some changes are necessary:
  - Do not need to add 1 to `lo`
  - Use epsilon value in conditions

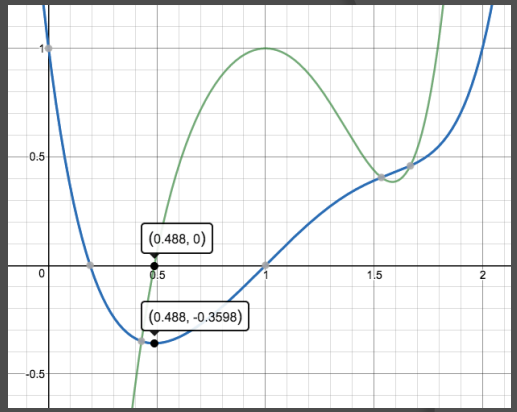
Extension: Ternary Search

- Given a function that only has 1 local maximum or minimum can we find it in sub $O(N)$ time?
  - Function can be discrete or continuous
  - Function might not be explicit
  - Function can only have slope of 0 at maximum/ minimum
- Ternary search is a way of solving this method
- Ternary search divides search area into thirds
- Checks if either the first third or the last third cannot contain the maximum/minimum
- Reduces search area to the remaining two-thirds
- We can also solve this problem using a binary search



Extension: Ternary Search

- Take derivative of the function
  - Calculate the difference between current point and previous/next point
- One side of the maximum/minimum will have derivative values less than 0 and the other side will have values greater than 0
  - Note that the derivative itself might not be monotonic
- Use binary search to find the point where the derivative changes sign (search for when derivative is 0)
  - Use lower or upper bound depending on problem



Blue line is original function, Green line is derivative

Ternary Search Example

- Find the maximum of this sequence:

`-2,-1,2,5,10,24,25,32,33,35,8,-10`

Let f(x) be seq(x+1) – seq(x)
lo = 0, hi = 11
mid = (0+11)/2 = 5
f(5) = seq(6) – seq(5) = 25-24 = 1 > 0
lo = mid+1 = 6
mid = (6+11)/2 = 8
f(8) = seq(9) – seq(8) = 35-33 = 2 > 0
lo = mid+1 = 9
mid = (9+11)/2 = 10
f(10) = seq(11) – seq(10) = -10-8 = -18 <= 0
hi = mid = 10
mid = (9+10)/2 = 9
f(9) = seq(10) – seq(9) = 8-35 = -27 <= 0
hi = mid = 9
Since *lo = hi*, the maximum is *seq(9) = 35*

THANK YOU!