Analysis of Algorithms

Functions in Increasing Order of Growth Rate

• Constant	c
• Logarithmic	log N
• Log-squared	$\log^2 N$
• Linear	N
• N log N	N log N
• Quadratic	N^2
• Cubic	N^3
• Exponential	2^N

Pigeonholing

- Realistically, we have eight choices
- Pretty good odds

Algorithms vs. Programs

- We analyze algorithms rather than programs
- Ignores poor implementation
- Focuses on the big picture

What to Analyze?

- Determine statements that contribute to the major work being done by the algorithm.
- Determine the number of times they get executed.

Worst, best, average case

- We **<u>ALWAYS</u>** perform an analysis for the general case of n.
- *Best case*: For input of size n, what is the **best** possible running time.
- *Worst case*: For input of size n, what is the **worst** possible running time.
- *Average case*: For input of size n, what is the **average** running time.

Linear Search

```
public static int linearSearch(int[]a, int e){
   for (int i = 0; i < a.length; i++){
       if (a[i] == e) return i;
   }
   return -1;
}
```


Best Case Analysis of Linear Search

- Size of array is of length *n*.
- In **best** case, the element we are looking for is in the first position of the array.
- In this case, we have one comparison.
- O(1)

Worst Case Analysis of Linear Search

- Size of array is of length *n*.
- In **worst** case, the element we are looking for is in the last position of the array or not located in the array
- In these cases, we have to look at all elements of the array, giving n comparison.
- O(n)

Average Case Analysis of Linear Search

- Chances of looking for 1st element in array: 1/n
- Same for all other elements
- Number of elements to compare:
  - 1st element: 1
  - 2nd element: 2
  - nth element: n

Average Case Analysis of Linear Search

- Sum of all cases: 1/n*1 + 1/n*2 + … + 1/n*n
- Factor out 1/n: 1/n*(1 + 2 + … + n)
- Change notation: 1/n * $\sum_{i=0}^{n} i$
- By induction, you can show that:
  $\sum_{i=0}^{n} i$ = n*(n+1)/2
- Dividing by n: (n+1)/2
- O(n)

EZ Analysis

- Implement algorithm
- Count the number of times key statements get executed
- Have the computer print the best, worst and average cases.