November 8, 2017

1 Warmup Runtime Questions

1.1 Informal

What is the runtime of:

1. Searching for a element in an unordered list?
2. BFS
3. Checking if a number is even or odd?
4. Travelling Salesman?
5. Searching for element in balanced binary search tree?

2 Formal

Formal definition of Big Oh:

$$\exists c, n_0 : \forall n > n_0 : f(n) \leq cg(n)$$

1. Prove that $f(n) = O(n^2)$, given

$$f(n) = O(n + g(n))$$

$$g(n) = O(n^2)$$

2. Prove that $f(n) = O(n^3)$, given

$$f(n) = O(n * g(n))$$

$$g(n) = O(n^2)$$

3. Prove that $f(n) = O(\log n)$, given

$$f(n) = O(\log(g(n)))$$

$$g(n) = \frac{2}{3}n + 20c$$

3 Sorting Algorithms

Sorting algorithms are given an unsorted list of elements and output a list with the same elements, now sorted by some key. We can assume that the input is a list of unique integers without loss of generality.
Online algorithm visualizers can be helpful to remind yourself how they work.

3.1 InsertionSort

```
def InsertionSort(lst):
        \\ 1st loop
        for (i in range(1,len(lst)-1)):
                j = i - 1
                \\ 2nd loop
                while (j > 0 and lst[j]<lst[i]:
                        j-=1
                lst.insert(j, lst.pop(i))
        return lst
```

How can we analyze the runtime? Try to find the worse case: Reverse sorted.
Ex: $8, 7, 6, 5, 4, 3, 2, 1$
Draw a $n*n$ grid to show the maximum number of operations. For each element in the outer loop, i, it is potentially compared to every element before it, j. The first i only is compared once so fill in 1 box in the first row of the grid (from the left). Next, the second i can be compared twice, so fill in 2 boxes. Continue to show how the number of operations can be represented by a triangle in the grid (looks like lower triangular matrix).

$$1 + 2 + 3 + 4 + ... + n - 1 + n = \frac{n(n+1)}{2} = O(n^2)$$