Friday 7/18: Algorithms pt. 1

Big-O Induction

Yesterday we looked at how we can show one function is big-O of another function, by choosing c and k values that fulfill the big-O definition. We can also use induction for a more rigorous proof. Note that you do not need to do a proof by induction unless you are asked for it.

Big-O induction example: Prove that 2^n is $O(n!)$

To show that 2^n is $O(n!)$ we need to show that there are positive real numbers c and k such that $0 \le 2^n \le c \cdot n!$ for all $n \ge k$. We can select $c = 1$ and $k = 4$ (try a few values so we know this is true). We now will prove $2^n \le n!$ for all $n \ge 4$.

Proof by induction on n:

Base Case: $n = 4$

$$2^4 = 16 \le 24 = 4!$$

Inductive Hypothesis: Assume that $2^n \le n!$ for all $4 \le n < j$.
Now we want to show $2^j \le j!$. We know $j! = j(j-1)!$
By IH, $(j-1)! \ge 2^{j-1}$ so

$$j! = j(j-1)! \ge j \cdot 2^{j-1}$$

Since $j \ge 4$ it holds that $j \ge 4 > 2$ so

$$j! = j(j-1)! \ge j \cdot 2^{j-1} > 2 \cdot 2^{j-1} > 2^j$$

Thus $2^n \le n!$ for all $n \ge 4$, so 2^n is $O(n!)$

Basic Data Structure Review

- array element access: $O(1)$
- changing array length, adding/deleting elements: $O(n)$
- adding/removing/reading/writing at the head of a linked list, or at a constant point in the list: $O(1)$
- adding/removing/reading/writing from the tail, or at a point that relies on n: $O(n)$

The material after this point is not on Monday's exam.

Pseudo-code analysis example: What is the big-O running time of the below code? What does it compute?

```
01 closestpair(p_1, ..., p_n) : array of 2D points)
02         best1 = p_1
03         best2 = p_2
04         bestdist = dist(p_1,p_2)
05         for i = 1 to n
06                 for j = 1 to n
07                         newdist = dist(p_i,p_j)
08                         if (i ≠ j and newdist < bestdist)
09                                 best1 = p_i
10                                 best2 = p_j
11                                 bestdist = newdist
12         return (best1, best2)
```

Lines 02-04 take $O(1)$.
Lines 05-11 have 2 nested for-loops, and the 08-11 takes $O(1)$, so in total this is $O(n^2)$.
Line 12 takes $O(1)$.
In total, the code takes $O(1 + n^2 + 1) = O(n^2 + 2) = O(n^2)$.
This code finds which 2 points are the closest to each other from the list.