CS41 Lab 2

September 17, 2018

In typical labs this semester, you'll be working on a number of problems. You are encouraged to work with one or two others. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

Note: If you do not feel fully confident when it comes to asymptotics, induction, or proofs, I strongly encourage you to focus on the initial two problems first.

1. **Induction.** Using induction, show that the following summations hold for all $n \geq 0$.
   - $\displaystyle\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$.
   - $\displaystyle\sum_{k=0}^{n} 2^k = 2^{n+1} - 1$.
   - for all positive $c \neq 1$, $\displaystyle\sum_{k=0}^{n} c^k = \frac{c^{n+1} - 1}{c - 1}$.
2. **Asymptotic analysis.** Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.
   (a) $\log_2(f(n))$ is $O(\log_2(g(n)))$.
   (b) $2^{f(n)}$ is $O(2^{g(n)})$.
   (c) $(f(n))^2$ is $O((g(n))^2)$.
   (d) If $g(n)$ is $O(h(n))$, then $f(n)$ is $O(h(n))$.
   (e) $g(n)$ is $\Omega(f(n))$.
3. **More asymptotic analysis.** Assume you have functions f, g, and h. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.
   (a) If f is $\Omega(g)$ and g is $\Omega(h)$, then f is $\Omega(h)$.
   (b) If f is $O(h)$ and g is $O(h)$, then $f + g$ is $O(h)$.
   (c) If f is $O(h)$ and g is $O(h)$, then $f \cdot g$ is $O(h)$.
   (d) If f is not $O(g)$, then g is $O(f)$.
4. **Pseudocode.** On the first day of the semester, you saw the following pseudocode for Dijkstra's Algorithm, which finds the shortest path in a graph G between a start vertex s and any other vertex.

Dijkstra(G, s, ℓ)

$S = \{s\}$.
$d[s] = 0$.
while $S \neq V$
    pick $v \in V \setminus S$ to minimize $\min_{e=(u,v):u \in S} d[u] + \ell_e$.
    add v to S.
    $d[v] = d[u] + \ell_e$
Return $d[\ldots]$.

(a) This pseudocode is *high-level* pseudocode. One reason why this is high-level is because it doesn't specify how to compute the edge $e = (u, v)$ such that $u \in S$ that minimized $d[u] + \ell_e$. Identify two other reasons why this is high-level pseudocode.

(b) Give low-level pseudocode for Dijkstra's algorithm. Don't worry about proving correctness or showing runtime analysis. Instead just focus on the pseudocode.