CS 173: Discrete Structures, Summer 2014
Homework 6 Solutions

This homework contains 3 problems and is due in class on Thursday, July 31st. **Please follow the guidelines on the class web page about homework format and style.**

In all questions, you must explain how you get your answers. Stating the answer with no supporting work will not receive full credit.

1. **Graphs** [**15 points**]

   Suppose G is a simple graph with n nodes. Prove that if G has more than $n-1$ edges, then G contains a cycle. [Note: "It's obvious" (or variations thereof) is not a proof.][Hint: Imagine starting with n nodes and no edges, and adding the edges one by one. What do you have to do to not create a cycle?]

   Solution. Suppose G is a simple graph with n nodes and m edges, where $m \geq n$. We want to show that G has a cycle.

   Let H be a graph such that $V(H) = V(G)$ and $E(H) = \emptyset$. Let $e_1, \ldots, e_m$ be the edges of G.

   Imagine adding the edges of G to H one at a time. At the end of this process, H will have turned into G.

   During this process, whenever we add an edge $e \in E(G)$, the two endpoints u and v of e are either in different connected components or already in the same connected component. There are two cases.

   Case 1: For some edge $e \in E(G)$, nodes u and v are in the same component. In this case, adding e creates a cycle: before e is added, there is already a path P from u to v, which we can combine with (v, u) to form a cycle. This is what we were trying to show.

   Case 2: For all edges $e \in E(G)$, u and v are always in different connected components. In this case, adding e reduces the number of connected components by 1. Since H starts with n connected components and there are m edges in G to add, at the end we will have $n - m \leq 0$ components in $H = G$, which is impossible. Thus, this case cannot occur.

   We have thus shown that case 1 must occur, in which case G has a cycle.

   $\square$

2. **Big O notation** [**15 points**]

   (a) Prove that x^4 is not $O(x^3)$.

Solution. We must show that for all $c, k \in \mathbb{R}^+$, there exists $n \geq k$ such that $n^4 > cn^3$.

Let $n = \max\{c, k\} + 1$. Then,

$$n^4 = n \times n^3 > cn^3,$$

as desired. $\square$

(b) Prove that $\log_2 n$ is $O(\log_{100} n)$.

Solution. We must show that there exist $c, k \in \mathbb{R}^+$ such that for all $n \geq k$, we have $\log_2 n \leq c \log_{100} n$.

Let $c = \log_2 100$ and $k = 1$. Then, for $n \geq k$,

$$\log_2 n = (\log_2 100)(\log_{100} n) = c \log_{100} n.$$

$\square$

3. **Algorithm Analysis [20 points]** Here is pseudocode for a procedure F. The inputs a and n are both natural numbers.

$\underline{F(a, n)}$:

- if $(n = 0)$
  - return a
- else
  - return $F(a, n-1) + F(a, n-1)$

(a) What mathematical function does F compute? You don't need to prove your answer, but some justification is required.

Solution. Since $F(a, n) = 2F(a, n-1)$, F doubles whenever n increases by 1, so $F(a, n) = c2^n$ for some constant c. Since $F(a, 0) = a$, we must have $c = a$, so $F(a, n) = a2^n$. $\square$

(b) Let $T(n)$ be the number of operations required to compute $F(a, n)$. Find a recurrence relation involving $T(n)$. [Note: Computing $F(a, n-1) + F(a, n-1)$ does not take the same number of operations as computing $2 \times F(a, n-1)$ does.]

Solution. If $n = 0$, then F just returns a. Otherwise, F calls itself twice with n replaced with $n - 1$, and adds the results together. Thus,

$$T(n) = \begin{cases} c & : n = 0 \\ 2T(n-1) + d & : \text{otherwise} \end{cases},$$

where c and d are constants. $\square$

(c) Solve the recurrence relation you had in part (b) to find a closed form for $T(n)$.

Solution.

$$
\begin{aligned}
T(n) &= 2T(n-1) + d && (1)\\
&= 2(2T(n-2) + d) + d && (2)\\
&= 2^2 T(n-2) + 2d + d && (3)\\
&= 2^2(2T(n-3) + d) + 2d + d && (4)\\
&= 2^3 T(n-3) + 2^2 d + 2d + d && (5)\\
&\vdots && (6)\\
&= 2^k T(n-k) + 2^{k-1} d + \cdots + d && (7)\\
&\vdots && (8)\\
&= 2^n T(0) + (2^{n-1} + \cdots + 1)d && (9)\\
&= 2^n c + (2^n - 1)d && (10)
\end{aligned}
$$

□