CMPS 6610 Introduction to Algorithms – Fall 16

9/6/16

1. Homework

Due **9/13/16** at the beginning of class

1. **Some Math (4 points)**

   - (2 points) Use the definition of the logarithm to prove for any $a, b, c > 0$:

     $$\log_b a = \frac{\log_c a}{\log_c b}$$

   - (2 points) Use (weak) induction to prove for any $n \geq 1$:

     $$\sum_{i=1}^{n} (i2^i) = 2 + (n-1)2^{n+1}$$

2. **Big-Oh ranking (14 points)**
   Rank the following fifteen functions by order of growth, i.e., find an arrangement $f_1, f_2, ...$ of the functions satisfying $f_1 \in O(f_2)$, $f_2 \in O(f_3)$,... . Partition your list into equivalence classes such that f and g are in the same class if and only if $f \in \Theta(g)$. For every two functions f_i, f_j that are adjacent in your ordering, prove shortly why $f_i \in O(f_j)$ holds. And if f and g are in the same class, prove that $f \in \Theta(g)$.

   $$n \ , \ n^2 \ , \ (\frac{3}{2})^n \ , \ \log^2 n \ , \ n2^n \ , \ 4^{\log n} \ , \ \log n \ , \ 2^n \ , \ \sqrt{n} \ ,$$
   $$2^{\log n} \ , \ 2^{(2^n)} \ , \ \sqrt{\log n} \ , \ n^{\log\log n} \ , \ 2^{\sqrt{2\log n}} \ , \ n \log n$$

   Bear in mind that in some cases it might be useful to show $f(n) \in o(g(n))$, since $o(g(n)) \subset O(g(n))$. If you try to show that $f(n) \in o(g(n))$, then it might be useful to apply the rule of l'Hôpital which states that

   $$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

   if the limits exist; where f' and g' are the derivatives of f and g, respectively.

3. **Big-Oh and Theta (6 points)**

   Show using the definitions of big-Oh and Θ:

   (a) (4 points) Let $a \in \mathbb{N}$ and $b \in \mathbb{R}^+$ be constants. Prove using the definition of Θ that $(n+a)^b \in \Theta(n^b)$.
   (Hint: You may need to distinguish different cases.)

   (b) (2 points) Prove if $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ then $f(n) \in O(h(n))$.

Flip over to back page $\Longrightarrow$

4. **Code snippet (3 points)**

Give the Θ-runtime for the code snippet below, depending on n. Make sure your bound is tight. Justify your answer. Assume all stack operations take constant time.

```
for(i=n; i>=1; i=i/5)
  stack.push(i);

for(i=1; i<=n; i++){
  while(!stack.isEmpty())
    stack.pop();
}
```