CS 5633 Analysis of Algorithms – Spring 12

1/19/12

1. Homework

Due **1/26/12** before class

1. **Loop invariant (10 points)**

   Consider the code below which computes b^n for all $n \geq 0$. Assume b is a constant.

```
power(b,n){
  result = 1;
  i = n;
  while(i>0){
    // Loop invariant:
    result = result * b;
    i--;
  }
  return result;
}
```

   (a) State a loop invariant for the while loop that will allow you to prove the correctness of the algorithm.

   (b) Use the loop invariant to prove the correctness of the algorithm. For this you need to prove by induction that the loop invariant holds for all iterations of your loop ("base step" and "inductive step"), and then use the loop invariant in the "termination step" to prove the correctness of the algorithm.

   (c) Give the runtime of this algorithm.

2. **Code snippets (10 points)**

   For each of the code snippets below give their Θ-runtime depending on n. Justify your answers.

   (a) Assume that `sqrt(k)` refers to $\lfloor \sqrt{k} \rfloor$ and that it can be computed in constant time.

```
for(i=2*n; i>=1; i=i-3)
  for(j=n; j>=4; j=j/5)
      for(k=n; k>=2; k=sqrt(k))
        print(" ");
```

   (b)

```
for(i=1; i<=n; i++)
  for(j=1; j+i<=n; j++)
    print(" ");

for(i=1; i<=n; i++)
  for(j=1; j*j<=i; j++)
    print(" ");
```

FLIP OVER TO BACK PAGE ⟹

(c) Assume all stack operations take constant time.

```
for(i=1; i<=n; i++)
  stack.push(i);

for(i=1; i<=n; i++){
  for(j=n; j>=1; j=j-5*i)
    while(!stack.isEmpty())
      stack.pop();
```

3. **Big-Oh Proofs (10 points)**
   Show using the definitions of big-Oh, Ω, and Θ:

   (a) $6n^3 - 7n^2 + 5 \in \Theta(n^3)$

   (b) $4n^3 + 6n \notin O(n^2)$

   (c) Use induction to prove $n! \in \Omega(2^n)$

4. **Big-Oh ranking (10 points)**
   Rank the following eleven 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 $f \in \Theta(g)$.

2^n	$\log^2 n$	n^2
$\log\log n$	$n\sqrt{n}$	2^{2n}
$\log n$	$\sqrt{n}$	$\log(2n)$
$\log\sqrt{n}$	$n\log n$	

As a reminder: $\log^2 n = (\log n)^2$ and $\log\log n = \log(\log n)$. 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 $f'(n)$ and $g'(n)$ are the derivatives of f and g, respectively.