CS 3343 Analysis of Algorithms – Fall 10

9/7/10

2. Homework

Due **9/14/10** before class

1. **Big-Oh ranking (8 points)**
   Rank the following 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 $f \in \Theta(g)$.

   $$\frac{1}{3}n^2 + 3n\ ,\ 3\ ,\ \sqrt{n}\ ,\ n^3\ ,\ 2^{\frac{n}{2}}\ ,\ \log n\ ,\ n^2\ ,\ 2^n\ ,\ n^2 \log n\ ,$$

   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.

2. **Code snippets (6 points)**
   For each of the code snippets below give their Θ-runtime depending on n. Justify your answers.

   (a) **(2 points)**

```
for(i=2n; i>=2; i=i-2){
  for(j=n; j>=2; j=j/2){
      print(" ");
  }
}
```

   (b) **(2 points)** Note that `sqrt` is the square root.

```
for(i=1; i<=sqrt(n); i=i+1){
    print(" ");
}

for(j=1; sqrt(j)<=n; j=j+1){
    print(" ");
}
```

FLIP OVER TO BACK PAGE ⟹

(c) **(2 points)**

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

3. **Heapify (2 points)**
   Give a worst-case example of a heap that will cause **Heapify_down** (in the book: MAX-HEAPIFY) to run in $\Omega(\log n)$ time. Justify your answer shortly.

4. **Sorted array (2 points)**
   Is an array that is sorted in increasing order a max-heap? What about an array that is sorted in decreasing order? Justify your answer.

5. **Min in a Max-Heap (2 points)**
   Where is the minimum element located in a max-heap? How can you compute it, and what is the runtime of your algorithm?