CMPS 2200 Introduction to Algorithms – Fall 12

10/22/12

6. Homework

Due **10/30/12** in the lab

1. **Heaps with links (6 points)**
   (a) (2 points) Consider storing a heap as a linked binary tree with pointers. Please give pseudo-code on how you would store a heap node, and describe which modifications you need to make to the heap routines that we discussed in class. What are the runtimes of the heap routines?
   (b) (2 points) Now consider storing a heap as a linked list with pointers. Please give pseudo-code on how you would store a heap node, and describe which modifications you need to make to the heap routines that we discussed in class. What are the runtimes of the heap routines?
   (c) (2 points) Assume you are given two heaps of height h each, that are given as linked binary trees. And assume we do not require that the last level of the heap is "flushed left", i.e., keys can be in any place in the last level. Give an $O(\log n)$-time algorithm that merges those two heaps into one heap (without the "flushed left" condition).

2. **Heapify (2 points)**
   Give a worst-case example of a heap that will cause **heapify_down(A,n,0)** to run in $\Omega(\log n)$ time. Justify your answer shortly.

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

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

5. **Bellman-Ford (3 points)**
   Let $G = (V, E)$ be a weighted, directed graph that possibly has negative weights but that has no negative weight cycles. And let $s \in V$ be a source vertex. For any $v \in V$ let $l(s, v)$ be the minimum number of edges in a shortest path from s to v (where the shortest path is of course based on the edge weights). Now, let $k = \max_{v \in V} l(s, v)$. Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $k + 1$ passes. **Do not assume that k is known in advance.**