Purdue University | Hambrusch
CS 580 | Spring 2010

Assignment 4

Due: Thursday, February 25, 2010 (before class)
Midterm Exam: Wednesday, March 10, 8-10pm, Lawson B155

1) (18 pts.) Consider the Set Cover problem: Given is a universal set U of size n and m sets, $S_1, S_2, \ldots S_m$, with $S_i \subseteq U$ and $\bigcup_{i=1}^{m} S_i = U$. You are to determine a selection of the sets such that their union is U and the number of sets chosen is a minimum. See Section 35.3 of CLRS for additional material.

(i) Describe an algorithm generating an optimum solution. Your solution can have exponential running time, but give an asymptotically tight bound.

(ii) Consider the following greedy strategy: An element of U is uncovered if it is not contained in any of the chosen sets. As the next set to include into the set cover choose the set with the largest number of uncovered elements. Repeat this process until there are no more uncovered sets. Describe an efficient implementation of this greedy strategy.

(iii) Show that the greedy strategy does not always generate an optimum solution. In particular, show that for a universe of size n there exist input instances for which the greedy strategy selects $\Theta(\log n)$ sets while the optimum solution contains two sets.

2) (16 pts.) Let T be a free tree on n vertices. There are no weights on the edges. Translate the following idea into an $O(n)$ time algorithms for determining the length of the longest path in T.

1. Set $c = 0$
2. Repeat the following until all vertices have been deleted: Delete from T all the vertices that are incident to fewer than 2 edges (the vertex and the edge are removed). Distinguish three cases:
   - Only two vertices are deleted and they are adjacent. This corresponds to the case where the tree had become a single edge, and now becomes empty as a result of the deletion of that edges endpoints. Update c as necessary.
   - A single vertex is deleted. This corresponds to the case where the tree had previously become a single vertex, and now becomes empty as a result of the deletion of that vertex. Update c as necessary.
   - If neither of the above two cases holds, update c as necessary.
3. Output c.

Give an efficient implementation and show the correctness of the algorithm.

3) (16 pts.) You are given an infinite series of stacks, $S_0, S_1, S_2, \ldots, S_i, \ldots$. The i-th stack can hold up to 3^i elements. The operation $push(x)$ pushes element x onto stack S_0. If stack S_0 already contains an element, the following process is used to free up space for x: if stack S_i is full, all elements in stack S_i are moved to stack S_{i+1} to make room. If S_{i+1} is full, its elements are moved to S_{i+2} to make room for the elements in S_{i+1}, $i = 0, 1, 2, \ldots$. The process eventually stops and x is put into S_0. Moving a single element from one stack to the next takes O(1) time.

(i) What is the asymptotic worst-case time bound of operation $push(x)$ when the stack structure contains a total of n elements?

(ii) Prove that the amortized cost of a push operation is $O(\log n)$, where n is the maximum number of elements in the stack structure.