Purdue University — Gopal/Hambrusch
CS 381 — Spring 2008

Assignment 4

Due: Friday, February 15, 2008 (before class)

1) (25 points) Consider a highway that is M miles long consisting of L lanes. At every mile, there exists a lane that is blocked from mile i to (just before) mile $i+1$. This information is recorded in an array LC of length M, where $LC(i) = j$ means lane j is blocked from mile i to $i+1$. For example, $LC = [1, 1, 3, 4, 1, 2, 2, 5, 4, 2, 2]$ describes the blockages on a 5-line highway of length 11. The objective is to find a route making the minimum number of lane changes. Changing from one lane into another one has a cost of 1. For the example given above, a solution with one lane change exists: start in lane 5 and change into lane 1 at mile 6 or later. It is not hard to see that more than one optimal solution can exist.

(i) For $M = 12$ and $L = 4$ give an example of blocked positions that maximizes the number of lane changes. Using the structure of the example, give a bound on the maximum number of lane changes in terms of M and L and explain your answer.

(ii) Consider now a greedy algorithm that starts in the lane having the farthest occurrence of a blockage and, whenever forced to switch lanes, it always switches into the lane having the farthest occurrence of a blockage. In the above example, greedy will start in lane 5 (since lane 5 has the farthest occurrence of the blockage) and at mile 6 change into lane 1 or lane 3 (these two lanes have no further blockages). Describe this greedy approach in pseudo-code and analyze its time performance in terms of M and L. Do not assume that either of these variables is a constant. Make sure to describe what data structures are used to select the next lane after a lane change. Full credit is given only for the best achievable time bound.

(iii) Prove that this greedy approach always generates a solution minimizing the number of lane changes.

2) (10 points) In the bin packing problem we are given a set of n objects, where the size s_i of the i-th object satisfies $0 < s_i < 1$, and an unlimited supply of unit-sized bins. We wish to pack all the objects into the minimum number of bins. Each bin can hold any subset of the objects as long as the total size of the objects does not exceed 1.

Consider the following greedy first-fit algorithm for assigning objects to bins: Assume you have already placed objects $1, 2, \ldots, k-1$ into bins $1, 2, ..., i$. To place object k, consider the bins with increasing indices and place object k into the first bin that can accommodate it. If none of the i bins can, place object k into a new bin. Show that this greedy algorithm does not always return the optimal solution to the bin packing problem. Note that you do not have to give an implementation of the algorithm.

3) (15 points) Consider the problem of finding the repeated element that was asked in Assignment 3 (Problem 1). You were asked to give an algorithm that ran in $O(n \log k)$ time and used $O(n)$ space. If you thought that this is the best possible running time, no not really! We can do even better using hashing: an algorithm that runs in *expected* $O(n)$ time and $O(n)$ (worst-case) space (note that the running time does not depend on k). The idea is to use universal hashing to hash each of the input elements to a hash table of size n. Collisions will be resolved by chaining the numbers that hash to the same slot using a linked list.

(i) Use universal hashing to describe an algorithm that runs in $O(n)$ expected time and $O(n)$ (worst-case) space. State the properties that your universal family of hash functions should satisfy. (You need not explicitly construct such a family.)

(ii) Analyze the expected running time of your algorithm. (Hint: Analyze the total expected number of collisions as follows. Let X_{ij} be the indicator random variable that indicates whether element i collides with element j. Compute $E[\sum_{\text{over all pairs } i \text{ and } j} X_{ij}]$ using linearity of expectation.)