CMPUT204
Department of Computing Science
University of Alberta

Homework Assignment #3
Due: Nov 3, 2004

Question 1. (15 marks)

a. (10 marks) Consider sorting an array of n floating-point numbers. The array, though, contains at most k unique numbers, where k is much smaller than n (i.e., $k < \log n$). Describe an efficient algorithm for sorting this array. The running time of your algorithm should be in $o(n \log n)$ (i.e., faster than MergeSort).

b. (5 marks) In terms of n and k, what is the asymptotic running time of your algorithm?

Question 2. (15 marks)

a. (12 marks) The heap property is one example of a *tree sort*. A rooted tree (not necessarily binary, d-ary, complete, or balanced), is sorted if and only if, for all non-root nodes n, key$[n] \leq$ key[parent$[n]$]. Write an efficient procedure TreeSort(T), that takes an arbitary rooted tree T and sorts the keys in the tree (i.e., the structure of the tree remains unchanged but the tree is sorted.) In your pseudocode feel free to use any convenient access functions you would like, (e.g., root$[T]$, children$[n]$, parent$[n]$).

b. (3 marks) What is the worst-case running time of your procedure for a tree with n nodes?

Question 3. (10 marks)

a. (5 marks) Write the array containing elements $\{1, \ldots, 10\}$ for which quicksort will require the most key comparisons.

b. (5 marks) Write the array containing elements $\{1, \ldots, 10\}$ for which quicksort will require the fewest key comparisons.

Question 4. (10 marks) Exercise 8.1-4 from the textbook.

Question 5. (20 marks)

a. (15 marks) Consider a robot moving in an $n \times n$ grid. It starts at the coordinate $(1, 1)$ and it moves either one square up or one square right, i.e., it increases either one but not both of its coordinates by exactly one (if not already equal to n). Eventually the robot will arrive at the coordinates (n, n). You also know, from past observations, the robot's probability of various moves. In particular, if the robot is at position (x, y), its next move will move it up with probability $u(x, y)$ and right with probability $1 - u(x, y)$. You want to prevent this robot from reaching its destination by placing a wall along its path (you cannot place a wall in $(1, 1)$ or (n, n)). Write a procedure to find the optimal place for a wall that will maximize the probability that the robot will enter the chosen location during its trip. Your algorithm should run in $O(n^2)$ time.

b. (5 marks) Describe how you could adapt the above algorithm to answer the question, "where is the robot most likely to be after k moves?"

Question 6. (15 marks)

a. (12 marks) There are n trading posts on the Saskatchewan River. At any of these posts you can rent a canoe to be returned at any other post down-stream (it is next to impossible to paddle against the current). For each possible

departure point i and each possible arrival point j the cost of a rental between i and j is $cost[i, j]$. However, it can happen that the cost of renting from i to j is higher than the total cost of a series of shorter rentals, in which case you can return the first canoe at some post k between i and j and continue the journey in a second canoe. There is no extra charge if you change canoes in this way. Give an efficient procedure to determine, given n and costs $cost[i, j]$, the minimum cost of a trip by canoe from post 1 to post n.

b. (3 marks) In terms of n, what is the running time of your algorithm?