CSE332 Data Abstractions, Summer 2010

Homework 6

Due Friday, August 6th, 2010 at the beginning of class.

Problem 1: Dijkstra's Algorithm

a) Weiss, problem 9.5(a). Use Dijkstra's algorithm and show the results of the algorithm in the form used in lecture — a table showing for each vertex its best-known distance from the starting vertex and its predecessor vertex on the path. Also show the order in which the vertices are added to the "cloud" of known vertices as the algorithm progresses.
b) If there is more than one minimum cost path from v to w, will Dijkstra's algorithm always find the path with the fewest edges? If not, explain in a few sentences how to modify Dijkstra's algorithm so that if there is more than one minimum path from v to w, a path with the fewest edges is chosen.
c) Give an example where Dijkstra's algorithm gives the wrong answer in the presence of a negative-cost edge but no negative-cost cycles. Explain briefly why Dijkstra's algorithm fails on your example.
d) Suppose you are given a graph that has negative-cost edges but no negative-cost cycles. Consider the following strategy to find shortest paths in this graph: uniformly add a constant k to the cost of every edge, so that all costs become non-negative, then run Dijkstra's algorithm and return that result with the edge costs reverted back to their original values (i.e., with k subtracted). Give an example where this technique fails and explain why it does so. (Hint: one simple example uses only three vertices.) Also, give a general explanation as to why this technique does not work.

Problem 2: Forkjoin Parallelism: Longest Series

Consider the problem of finding the longest sequence of some number in an array of numbers: longest sequence(i,arr) returns the longest number of consecutive i in arr. For example, if arr is {2,17,17,8,17,17,17,0,17,1} then longest sequence(17,arr) is 3 and longest sequence(9,arr) is 0.

(a) In pseudocode, give a parallel fork-join algorithm for implementing longest sequence. Your algorithm should have work O(n) and span O(log n) where n is the length of the array. Do not employ a sequential cut-off: your base case should process an array range containing one element.
Hint: Use this definition:

```
class Result {
      int numLeftEdge;
      int numRightEdge;
      int numLongest;
      boolean entireRange;
      Result(int l, int r, int m, boolean a) {
            numLeftEdge=l; numRightEdge=r;
            numLongest=m; entireRange=a;
      }
}
```

For example, numLeftEdge should represent the length of the sequence at the beginning of the range processed by a subproblem. Think carefully about how to combine results; given left and right 'Result's, how can you compute the merged 'Result'?

(b) In English, describe how you would make your answer to part (a) more efficient by using a sequential cut-off. In pseudocode, show the code you would use below this cut-off.

Problem 3: Forkjoin Parallelism: Leftmost Occurrence of Substring

Consider the problem of finding the leftmost occurrence of the sequence of characters cseRox in an array of characters, returning the index of the leftmost occurrence or -1 if there is none. For example, the answer for the sequence cseRhellocseRoxmomcseRox is 9.

(a) In English (though some high-level pseudocode will probably help), describe a fork-join algorithm similar in design to your solution in problem 2. Use a sequential cut-off of at least 6 (the length of cseRox) and explain why this significantly simplifies your solution. Notice you still must deal with the leftmost occurrence being "split" across two recursive subproblems.
(b) Give a much simpler fork-join solution to the problem that avoids the possibility of a "split" by using slightly overlapping subproblems. Assume a larger sequential cut-off, for example 100. Give your solution precisely in pseudocode.