Prof. Sergio A. Alvarez
21 Campanella Way, room 569
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CS383, Algorithms
Spring 2009
HW6

1. The computational task of computing the maximum consecutive subsequence sum (MSS) of a numerical array of length n is described below, together with a simple algorithm (Algorithm 1) that performs this task. For example, the correct output for the MSS task for the input instance $a = \{-5, 3, 12, -8, 6, 4\}$ is $S = 17$, $\underline{i} = 2$, $\overline{i} = 6$. Your main goal will be to develop an efficient divide and conquer algorithm for the MSS task.

   Algorithm 1: Simple Maximum Consecutive Subsequence Sum
   Input: Nonempty array $a[1...n]$ of numbers.
   Output: Maximum consecutive sum $S = \sum_{\underline{i} \le i \le \overline{i}} a[i]$, associated indices $\underline{i}$, $\overline{i}$.
   SIMPLEMSS($a[1...n]$)
   (1) $S = a[1]$, $\underline{i} = 1$, $\overline{i} = 1$
   (2) **foreach** $i = 1$ to n
   (3) **foreach** $j = i$ to n
   (4) $S' = \sum_{i \le k \le j} a[k]$
   (5) **if** $S' > S$
   (6) $S = S'$, $\underline{i} = i$, $\overline{i} = j$
   (7) **return** $S, \underline{i}, \overline{i}$

   (a) Find the worst-case asymptotic running time of Algorithm 1 in big Θ notation. Explain carefully.

   (b) Implement (in detailed pseudocode) two functions **MSLeft**($a[1...n]$) and **MSRight**($a[1...n]$) that run in time $\Theta(n)$ and that return the maximum consecutive subsequence sums (and corresponding index ranges) over ranges of the restricted forms $i...\lfloor n/2 \rfloor$ and $\lfloor n/2 \rfloor + 1...i$, respectively (note that these ranges reach the midpoint of the array). For example, for the input array $a = \{-5, 3, 12, -8, 6, 4\}$, **MSLeft**($a$) and **MSRight**($a$) would return the values $S = 15$, $\underline{i} = 2$, $\overline{i} = 3$ and $S = 2$, $\underline{i} = 4$, $\overline{i} = 6$, respectively.

   (c) Design a $\Theta(n)$ gluing function for a divide and conquer algorithm for the MSS task. Provide detailed pseudocode. Gluing inputs include the MSS solutions S_L, $\underline{i}_L$, $\overline{i}_L$ and S_H, $\underline{i}_H$, $\overline{i}_H$ for the lower and upper halves $a[1...\lfloor n/2 \rfloor]$ and $a[\lfloor n/2 \rfloor + 1...n]$, respectively, as well as the full array $a[1...n]$. For example, for $a = \{-5, 3, 12, -8, 6, 4\}$, one would have $S_L = 15$, $\underline{i}_L = 2$, $\overline{i}_L = 3$ and $S_H = 10$, $\underline{i}_H = 5$, $\overline{i}_H = 6$. The output of the gluing function should be a solution S, $\underline{i}$, $\overline{i}$ to the MSS task on the full array $a[1...n]$. Your gluing function may call the functions defined in 1b.

   (d) Write detailed pseudocode for a divide and conquer algorithm that solves the MSS task using the gluing function from 1c. What is your algorithm's running time? Explain.

2. Apply Dijkstra's algorithm to the weighted graph G that has the vertices A, B, C, D, E, F and the edge weights given below in adjacency list format (edges not listed have weight ∞). Let A be the start vertex.

$$w(A,B) = 20,\ w(A,C) = 30,\ w(A,F) = 100$$
$$w(B,C) = 30,\ w(B,D) = 20,\ w(B,F) = 100$$
$$w(C,D) = 40,\ w(C,E) = 15,\ w(C,F) = 60$$
$$w(D,E) = 50,\ w(D,F) = 5$$
$$w(E,F) = 60$$

   Provide a complete iteration table that shows the contents of the distance and predecessor arrays at each stage, and that indicates what vertex is selected for expansion in each case.

3. You're starting a new business that requires delivering orders from your warehouse to customers in various rural towns. You have a fleet of delivery trucks and a known grid of roads that they can travel on. There is a single road between every pair of towns. Road conditions sometimes shut down travel along a stretch of road. Such events are unpredictable, but you have for each road an estimate of the probability that the road will be closed at any given time. You are interested in determining routes that will maximize the probability that a delivery attempt will not run into a closed road anywhere along the route. Assuming that road closings occur independently of one another, this "all clear" probability along a route $t_0, t_1, ...t_k$, where the t_i are the towns along the route, is the *product* of the individual probabilities $p_{t_i,t_{i+1}}$ that each of the roads between consecutive towns t_i and t_{i+1} will be open.

   (a) You wish to compute, for each of the n towns $1, ...n$, the route from the warehouse to that town that maximizes the all clear probability described above. The probability $p_{i,j}$ that the road from town i to town j will be clear (open) is known in advance for every pair of towns. The warehouse is considered to be town 0. Cast this as a computational task involving a graph. Precisely describe the graph involved, and give a detailed input-output specification of the computational task. Make explicit note of any constraints on the inputs.

   (b) Describe an algorithm that solves the computational task described in 3a. Provide detailed pseudocode with explanations.

4. Suppose that a store offers that for each item purchased at full price, you can get another item of the same price or less for free. You would like to buy a total of $2n$ items, in a way that maximizes the money saved through this offer. You know the price of each item, and you wish to pair them in the form $(P_1, F_1),\ (P_2, F_2),\ \cdots (P_n, F_n)$, where you pay for item P_i and get item F_i for free based on the purchase of P_i (so F_i costs no more than P_i).

   (a) Describe an algorithm that takes a list of $2n$ items and their prices as inputs, and returns an optimal pairing $(P_1, F_1),\ (P_2, F_2),\ \cdots (P_n, F_n)$. Optimality here means that the total amount paid should be minimized (equivalently, the total amount saved should be maximized). Give detailed pseudocode.

   (b) Prove that your algorithm from 4a returns an optimal solution. Give a carefully reasoned and complete explanation.