CSCE 500 Midterm Exam #2

11/16/2016

1. The recurrence of Procedure CUT-ROD(p, n) is given by $T(n) = 1 + \sum_{j=0}^{n-1} T(j)$, with $T(0) = 1$. Solve $T(n)$. (10%)

   Procedure EXTENDED-BOTTOM-UP-CUT-ROD(p, n) below lowers its time complexity by utilizing two auxiliary arrays $r[0 .. n]$ and $s[0 .. n]$ to keep solutions for sub-problems obtained so far, as listed below. (1) Fill in the two missing statements in the procedure and (2) give its time complexity. (6%)

```
EXTENDED-BOTTOM-UP-CUT-ROD(p, n)
  let r[0..n] and s[0..n] be new arrays
  r[0] = 0
  for j = 1 to n
      q = -∞
      for i = 1 to j
          if q < p[i] + r[j - i]
              1.
              2.
      r[j] = q
  return r and s
```

2. The problem of optimal parenthesization over a chain of matrix multiplications can be solved by a divide-and-conquer approach recursively. Let $m[i, j]$ denote the minimum number of scalar multiplications needed to compute $A_i \bullet A_{i+1} \bullet A_{i+2} \bullet \cdots \bullet A_j$, with A_k sized as $p_{k-1} \times p_k$ (for $i \le k \le j$), give the recurrence definition of the problem. (8%)

   The recurrence leads to exponential complexity but can be solved by dynamic programming much faster. What is the resulting time complexity and how do you get that complexity result? (6%)

3.

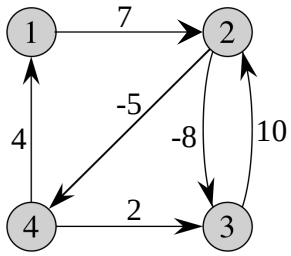
4. Given a set of 4 keys, with the following probabilities, determine the cost and the structure of an optimal BST (binary search tree), following the tabular, bottom-up method realized in the procedure of OPTIMAL-BST below to construct and fill $e[1..5, 0..4]$, $w[1..5, 0..4]$, and $root[1..4, 1..4]$.

i	0	1	2	3	4
p_i		0.12	0.08	0.23	0.17
q_i	0.06	0.13	0.07	0.05	0.09

(a) Fill in the two missing statements in the procedure. (6%)
(b) Construct and fill the three tables, and show the optimal BST obtained. (24%)

```
OPTIMAL-BST(p, q, n)
 1  let e[1..n+1, 0..n], w[1..n+1, 0..n],
          and root[1..n, 1..n] be new tables
 2  for i = 1 to n + 1
 3      e[i, i − 1] = q_{i−1}
 4      w[i, i − 1] = q_{i−1}
 5  for l = 1 to n
 6      for i = 1 to n − l + 1
 7          j = i + l − 1
 8          e[i, j] = ∞
 9          w[i, j] = w[i, j − 1] + p_j + q_j
10          for r = i to j
11              t = e[i, r − 1] + e[r + 1, j] + w[i, j]
12              if t < e[i, j]
13                  1.
14                  2.
15  return e and root
```

5.



6. Follow depth-first search (*DFS*), starting from Node 1, to traverse the graph shown in the preceding problem, with its edge weights all ignored. Mark (1) the type of every edge and (2) the discovery and the finish times of each node. (10%)