CSI 3105 ASSIGNMENT 3 Fall, 2014

Due Date: Monday, November 17, 2014, by 5:30 pm.. You can hand in your assignment directly to me in class that day, or else you can bring it to my office (STE 5106) before 5:30 (put it under my door if I am not there). Assignments from 1 minute to 24 hours late will lose 10%. Assignments received after 5:30 pm on November 18th will receive a grade of 0.

On this and all future assignments: You must be sure to include your name and student ID on the assignment, as well as the course code and the assignment number, and to staple your assignment. In your assignment, please be sure to explain all of your solutions CLEARLY, or you will lose marks. To be complete, in any worst case analysis you do you must state the input size, the basic operations being counted, and a general input of your fixed input size which gives the worst case (you should state these things in your solution even if some of them are given in the questions). Whenever possible, simplify your answers by using any appropriate formulas found in Appendix A. Also, you will be marked on the efficiency of any algorithms you design.

This assignment will be marked out of 60.

---

1. [20 marks]

Suppose you are given an undirected graph G=(V,E) with nodes labelled 1, 2, 3,…,n, and a specific node k in V. You wish to answer the following question: Is node k adjacent to all other nodes in the graph? Discuss CLEARLY what the complexity would be to answer this question for each of the following data structures for G (which were discussed in class). Note that while discussing the complexity, you will need to give some description about HOW you would solve this problem in each case (this description does not have to be detailed pseudocode, but instead should be a clear description in English regarding what part of the data structure you need to scan and why--enough that you can talk about the complexity).

a) Edge List
b) Adjacency Matrix
c) Adjacency List.

2. [10 marks]

Using the algorithm as taught in class, solve the following 0-1 knapsack problem where the set S consists of 4 items. Be sure to show your table, as done in class, and state your final solution. You do not need to show your other work.

W = 9

Item number	1	2	3	4
w_i	2	3	4	5
p_i	1	4	3	4

3. [10 marks total]
Consider the Matrix Chain Multiplication problem, for which we designed a dynamic programming algorithm. Suppose that we found that this algorithm was too painful for words, and so we decided to try to design a greedy algorithm for this problem. Let the matrix chain consist of matrices A1*A2*A3*...*An, where each matrix Ai has dimensions d_{i-1} x d_i (this is the same notation we used in class).

a) [5 marks] Consider the following idea for a greedy algorithm:

> First multiply the two matrices whose common dimension d_i is the smallest, and continue in the same way.

Show that this greedy algorithm does not always work by providing an example of a matrix chain problem with not more than 4 matrices for which the greedy algorithm described fails. You must also provide the two solutions, i.e. the one from the greedy algorithm, and the optimal solution, and demonstrate that the algorithm fails for your example by giving the number of multiplications for each of your solutions.

b) [5 marks]
Design a different greedy algorithm for this problem, and describe it clearly. Show that it also fails by providing a counter example, in the same manner as a).

4. [20 marks total]
Suppose you have an array S indexed from 1 to n which contains n numbers, not in any particular order, and you wish to count how many times a given number x occurs in S. Consider the recursive algorithm below for this which finds the number of occurrences of x in the index range i...j in S. Of course, solving the problem would involve an initial call to the algorithm for the range 1..n:

```
int CountOccur (int i,j)
{
int mid, count;
if j < i
  return  0;
else {
   mid =⌊(j + i)/2⌋
   if S[mid] = x
      Count = 1 + CountOccur(i,mid-1) + CountOccur(mid+1,j);
   else
      Count =  CountOccur(i,mid-1) + CountOccur(mid+1,j);
   return Countt;
   }
}
```

a) [15 marks]
Design a dynamic programming version of the above recursive algorithm and write the pseudocode for your algorithm. Your algorithm must be a dynamic programming

version of this algorithm. For this algorithm, you should have a table which has an entry for every index pair i and j, $1 \leq i \leq n$, $1 \leq j \leq n$. For index pair i and j your table should contain the number of times x appears in S in the range i...j. And of course, you must allow for the fact that you will need entries for $j<i$ sometimes too (so you may need to enlarge your table for this—think about this).

b) [3 marks]
Illustrate your algorithm on the following example:
S = [2, 3, 6, 2, 7, 8, 2, 9], x =2.

c) [2 marks]
State the complexity of your algorithm, where you are counting the number of table entries calculated.