Take Final (30 Marks) **Oct 30th, 2016**

Dept. of Electrical Engineering, IIT Madras
EE4371 - Data Structures and Algorithms

- ▷ **Please write clear answers. Prefer a LyX or Latex file with well formatted maths equations.**
- ▷ **Code should be well commented and self-explanatory.**
- ▷ **Code should run!!**

1. **For this problem, please refer to https://en.wikipedia.org/wiki/Knapsack_problem in wikipedia. It discusses the knapsack problem and discusses both dynamic programming and greedy algorithmic approaches.** . . . . [10]

   > **Given a set of N objects of positive weights $\{w_i\}$, find the subset of these objects that maximizes their sum subject to**
   > $$\sum_{i=i}^{m} w_{k_i} \leq W_0 - \ln m$$
   > **for given $m \geq 0$ and $W_0 > 0$.**

   Formulate both a dynamic programming approach and a greedy algorithm approach for this problem. How does the factor, $-\ln m$, which penalises the use of larger number of items, affect your algorithm?
   Write code for this problem, which reads in input1.dat and writes out the solution found via both approaches on the screen. The table of intermediate calculations should be written out to output1.dat. Estimate the number of operations required using direct recursion and for using the two methods above.

2. **For this problem, please refer to the uploaded document nr.pdf. This contains a few pages from the free ebook Numerical Recipes in C (2nd Ed).** . . . . [8]
   An $m \times m$ square sparse matrix A is to be stored and used to compute $A\vec{x}$ and $\vec{x}^T A$ for arbitrary vectors $\vec{x}$. The matrix A has at most 6 non-zero entries on each row, though m could be $1,00,000$. Numerical Recipes in C has the code and the algorithms used to store such matrices in the section *Indexed Storage of Sparse Matrices*. Their technique uses arrays (and is one of several). Modify the algorithms to use pointers and linked lists to store A. The goal is to make computing both Ax and $x^T A$ as cheap as it is in the array approach. Compare the algorithms and discuss the benefits and problems of your approach compared to the Numerical Recipes approach.
   The algorithm should be implemented for the matrices in input2.dat (both A and x are given) and the value of $A\vec{x}$ and $\vec{x}^T A$ should be written out to output2.dat. (You can verify the correctness by just performing regular matrix multiplication in python.)

3. You can look at Directed Acyclic Graphs and at Spanning Trees in the textbook for this problem. .... [12]
A geneology is a directed graph connecting parents to their children. Assume strict exogamy operates and that none of the spouses are part of the family tree. Our geneology only tracks the those related by blood, and ditches the spouses. So if A is the son of B, A and B are in the geneology, but A's mother is skipped. It is also assumed that people in the geneology never have children with each other, but only with others outside the geneology. So each node in our geneology can trace its descent form the original ancestor vai a unique path.

(a) Define a data structure to store the information in the geneology. People are nodes in this graph, while edges are parent child relationships. The value of the edge is the age of the parent when child was born. Nodes contain age of the individual when he/she died (and an Id and their names). Will a binary tree be the appropriate structure? What about a $2-3$ tree? Or should it be a general tree or graph? Also develop the algorithm that will read the input file and store the graph with its relationships. Determine the original ancestor.

(b) Use the graph to determine the number of descendents of each member of the geneology. What algorithm will you use? How is edge information stored? What is the time complexity of your algorithm to obtain the desired information, if the number of generations below the node is k?

(c) We wish to determine how many individuals in the geneology lived to see their great-grandchildren. How will you traverse the graph? For each node, how will you obtain its list of great-grandchildren, and how will you determne if there was an overlap? Obtain the time complexity to obtain this information in terms of the nodes in the graph, n, and the number of generations, k.

Use the geneology in input3.dat, which gives an incidence matrix, with A_{ij} representing the age of person i when j was born. Additionally, the file contains a list of nodes, each row containing the name of the person and the age at which the person died.