**CMPE 300 ANALYSIS OF ALGORITHMS** 10.11.2017

###### MIDTERM

1. Consider the fractional knapsack problem.
2. Describe the problem
3. Give the *formal definition* of the problem
4. Write a greedy algorithm in pseudocode for solving this problem
5. Show the execution of the algorithm briefly on an example data and show the output
6. Analyze the worst-case complexity of the algorithm
7. A variation of this problem is the 0/1 knapsack problem. Explain it briefly and show with an example that the algorithm in part (c) may not find an optimal solution to this variation of the problem
8. Consider the graph below. Suppose that the start node is A.
9. Apply the depth-first search algorithm. Show clearly how the nodes are visited and explain in detail. List the order of node visits. Show the DFS-tree.

(From a node when there are alternative nodes to go, choose the one with the smallest letter. For instance, among A, B, and C, choose A.)

1. Apply the breadth-first search algorithm. Show clearly how the nodes are visited and explain in detail. List the order of node visits. Show the BFS-tree.

(From a node when there are alternative nodes to go, choose the one with the smallest letter. For instance, among A, B, and C, choose A.)

1. Analyze the complexity of the DFS algorithm. Assume that the graphs are stored in adjacency list representation. First, trace the execution of the algorithm for the graph given above in detail, showing the execution of the basic operations explicitly. Then, analyze the complexity clearly.

(The analysis must be general; not for the example graph. But the trace of the example graph must conform with your general analysis.)

1. Consider the following algorithm:

function Example (L[low:high], size)

 if (size≥1) then

 for i←low to high do

 Data1 ← L[i]

 if (L[i] ≤ L[high]) then

 Data2 ← L[size-1-i]

 endif

 endfor

 call Example (L[low:(size/2)-1], size/2)

 endif

end

*(continued on next page)*

The algorithm is initially called as “call Example(L[0:n-1], n)”. Assume that the data size *n* is a power of two.

The basic operation is the two assignment statements. Analyze the average complexity of the algorithm. Assume that each element of the array L is a distinct integer between 1 and *n*.

During the analysis, you must use the technique of dividing the algorithm into steps and you must use conditional expectation explicitly in some part of the analysis. Otherwise, no points will be given. Show all the analysis *clearly*.

*Notes:*

* Where pseudocode is required, the syntax of the pseudocode must be strictly followed. No points will be given if the syntax is not followed or any other language (e.g. C) is used.
* Questions 1,3:35 points, 2:30 points
* Time: 1:30 hours
* Close notes and books

Beni, gözlerin götürür

Gözlerin

Aşkla, acıyla…

Kuşatmışlar

Sesimi, soluğumu

Kesilmiş

Tuz-ekmek payım

Vurgunum

Ve darda,

Gözaltındayım.

Dal, kor keser

Penceremde açarsa

Kuş, vurulur

Üzerimden uçarsa.

Ve hal böyle böyle,

Yol bu yöndeyken

Gelir,

Ki her gelişinde

Daha da içten

Gelir,

Soluk soluğa

Benim olursun.

Amansız sarmasında

Kollarımın

Esrik,

Çığlık çığlığa

Erir, kar gibi vücudun…

Nicedir,

Kahpe ağzında

Bir salgın,

Bir deprem gibi künyemiz

Nicedir,

Başımıza zindan dünyamız

Biz ki

Yarınıyız halkın,

Umudu, yüzakıyız,

Hıncı, namusu…

Şafakları,

Taaa şafakları

Hey canım,

Kalbim

Dinamit kuyusu