CSCD320 Homework6, Winter 2012, Eastern Washington University, Cheney, Washington.

Name: **EWU ID:** **Due:** 11:59pm, Feb. 26, 2012 (Sunday)

Please follow these rules strictly:

1. Write your name and EWUID on **EVERY** page of your submission.
2. Verbal discussions with classmates are encouraged, but each student must independently write his/her own solutions, without referring to anybody else's solution.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Blackboard system). Send in whatever you have by the deadline.
4. Submission must be computer typeset in the **PDF** format and sent to the Blackboard system. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this homework as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and nearly all the computer science and math professionals for paper writing.
5. Your submission PDF file must be named as: **firstname_lastname_EWUID_cscd320_hw6.pdf**
(1) We use the underline '_' not the dash '-'.
(2) All letters are in the lower case including your name and the filename's extend.
(3) If you have middle name(s), you don't have to put them into the submission's filename.
6. Sharing any content of this homework and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Problem 1 (30 points)**.** *We are given a set of activities:* $A = \{a_1, a_2, \ldots, a_n\}$*. Each activity* a_i *has a starting time* s_i *and an ending time* f_i*, meaning activity* a_i *happens during the time slot* $[s_i, f_i)$*. We want to select a subset* $S \subseteq A$*, such that:*

(1) the number of activities in S *is* **minimized***.*
(2) all the activities in S *do not overlap each other.*
(3) every activity in $A \setminus S$ *must overlaps at least one activity in* S*.* [1]

Design an efficient greedy algorithm that returns such a set S *and schedules the activities in* S*. Describe your algorithmic idea and explain why it works.*

Problem 2 (10 points)**.** *Suppose you are given a text of* 87 *characters drawn from the alphabet* {`a, b, c, d, e, f`} *and the frequency of each letter is:*

a	*b*	*c*	*d*	*e*	*f*
2	*20*	*25*	*6*	*25*	*9*

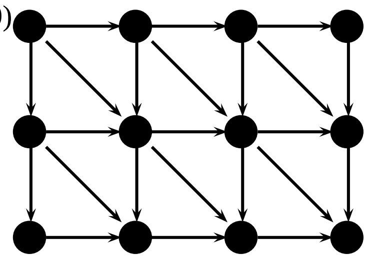
1. *Create a Huffman tree for this text (you may have multiple different Huffman tree for this text, but anyone is fine).*
2. *Show the Huffman code of each letter.*

[1] $A \setminus S = \{a \mid a \in A \cap a \notin S\}$

3. *Computer the size of the Huffman code compressed version of this text in bits.*

4. *Calculate the compression ratio:*
$$\frac{\textit{compressed text size}}{\textit{raw text size}}$$
assuming we use 8-bit ASCII code for storing the raw text.

Problem 3 (30 points)**.** *Consider a directed graph arranged into rows and columns. Each vertex is labeled (j, k), where row j lies between 0 and M, and column k lies between 0 and N. The start vertex is $(0, 0)$ and the destination vertex is (M, N). Each vertex (j, k) has an associated weight $W(j, k)$. There are edges from each vertex (j, k) to at most three other vertices: $(j + 1, k)$, $(j, k + 1)$, and $(j + 1, k + 1)$, provided that these other vertices exist.*



An example graph with $M = 2$ and $N = 3$.

Count the number of different paths from (0, 0) to (M, N) that have minimum total weight.

1. *Provide a recursive formula that computes the exact value that is specified in the problem.*

2. *Express the recursive formula as a bottom-up table-driven dynamic programming algorithm, and determine its running time.*

Problem 4 (15 points)**.** *Search and learn two existing algorithms that use the greedy strategy, in addition to those that we have discussed the class. For each algorithm, in your own language, concisely and clearly describe:*

1. *the problem statement*

2. *the greedy strategy and why it works ?*

3. *the source of your finding. For example, the url of the webpages, the title and page of a book, the title/author/year of an article, etc.*

Problem 5 (15 points)**.** *Search and learn two existing algorithms that use the dynamic programming strategy, in addition to those that we have discussed the class ("rod-cutting" and "matrix-chain"). For each algorithm, in your own language, concisely and clearly describe:*

1. *The problem statement*

2. *Why recursion-based idea does not work ?*

3. *Why dynamic programming works ?*

4. *The source of your finding. For example, the url of the webpages, the title and page of a book, the title/author/year of an article, etc.*