CS 456: Advanced Algorithms

Homework #03

Assigned Date : Friday, October 31, 2014
Due Date : Tuesday, November 18, 2014 @ 12:29:59 p.m.

Instructions

- This is an individual assignment. **Do your own work**. Acts of academic misconduct (plagiarism, use of illegal solutions manuals, etc.) will be strictly monitored and will be subjected to one or more of the penalties outlined in the course syllabus.
- Your answers should be produced using a word processing application.
- Handover a **printed**, **stapled** copy of your solutions to the instructor at the beginning of class on due date. Make sure to include your name and the last 3 digits of your SIUE ID in the first page of your solutions sheet.
- A moodle dropbox will become available the day before the due date as a secondary submission option. Ensure to submit a PDF document if you decide to use the moodle dropbox. **DO NOT** email your solutions to the instructor.
- Make proper arrangements, after consulting the instructor, to deliver your solutions **BEFORE** the due date, if you have a planned absence on the due date.
- Answer all questions
- Your assignment is due on **Tuesday, November 18, 2014 @ 12:29:59 p.m.**
- Total points: [**80 points**] for undergrads, [**110 points**] points for grads

Questions

Q1. Dynamic Programming [**20 points**]

(a) A burglar breaks into a museum and finds n items of weights $w_1, w_2, \ldots, w_n$ of values $v_1, v_2, \ldots, v_n$. However, the burglar's knapsack has a maximum weight capacity of W. The burglar wants to select the most valuable subset of items he can fit into his knapsack under its weight constraint. At the same time, none of the items can be broken down into smaller pieces, thus either he has to decide to take a particular item or simply leave it.

i. [**10 points**] Derive a recurrence relation $K(i,j)$ that defines the value of the most valuable subset of the first i items that fit into the knapsack of capacity j.

ii. [**5 points**] Write a pseudocode algorithm that takes $v[n], w[n], W$ and returns the subset of selected items $s \in n$. Prove that the runtime efficiency of your algorithm is $O(nW)$.

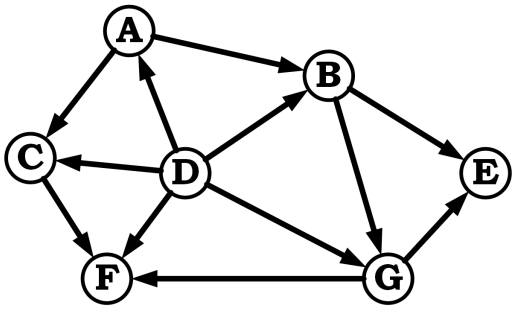
iii. [**5 points**] Use your recurrence to solve the following instance of the knapsack problem. Assume $W = 6$.

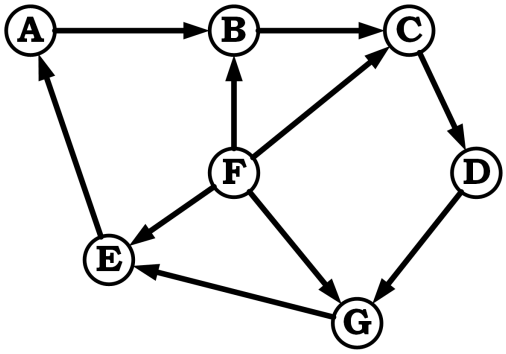
item	weight	value
1	3	$25
2	2	$20
3	1	$15
4	4	$40
5	5	$50

Q2. Topological Sorting [**20 points**]

(a) [**10 points**] Prove that the topological sorting problem has a solution if and only if it has a DAG.

(b) [**10 points**] Use the DFS-based algorithm to solve the topological sorting problem for the following digraphs:



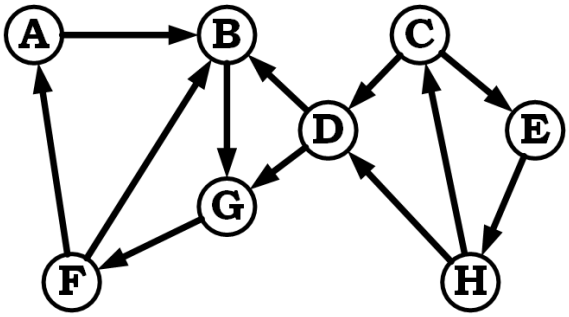


Q3. Strongly connected components [**15 points**]

(a) [**5 points**] Write the pseudocode for an algorithm to detect the strongly connected components of a given digraph

(b) [**5 points**] What is the time efficiency of your algorithm? Show your work.

(c) [**5 points**] Determine the strongly connected components of the following graph



Q4. DFS, BFS [**15 points**]

(a) [**5 points**] Prove that a BFS of an undirected graph can not have back edges or forward edges
(b) [**5 points**] Prove that a BFS of a digraph has no forward edges
(c) [**5 points**] Prove that a DFS of an undirected graph can not have cross edges or forward edges

Q5. [**10 points**] Write a pseudocode algorithm that takes a graph $G = (V, E)$ and returns a boolean value to indicate whether the graph has cycles. Prove that the time efficiency of algorithm is $O(V)$

Graduate Students Only (No Extra Credit)

Q6. [**20 points**] A subsequence is *palindromic* if it is the same whether read left to right or right to left. For instance, "`A nut for a jar of tuna`" is a palindrome.

(a) [**10 points**] Derive a recurrence relation $L(i,j)$ where $(1 \leq i \leq j \leq n)$ that defines the longest palindromic subsequence s of a given sequence $x[i, \cdots n]$.
(b) [**10 points**] Write an pseudocode algorithm that takes a sequence $s[i, \ldots, n]$ and returns the length of the longest palindromic subsequence.. Prove that the running time of your algorithm is $O(n^2)$.

Q7. [**10 points**] A directed graph $G = (V, E)$ **singly connected** if $u \rightsquigarrow v$ implies that G contains **at most** one simple path from u to v for all vertices $u, v \in V$. Give an efficient algorithm to determine whether a given graph is singly connected.