CS 58000_01 Algorithm Design Analysis & Implementation (3 cr.)

Final Exam (Assignment As05)

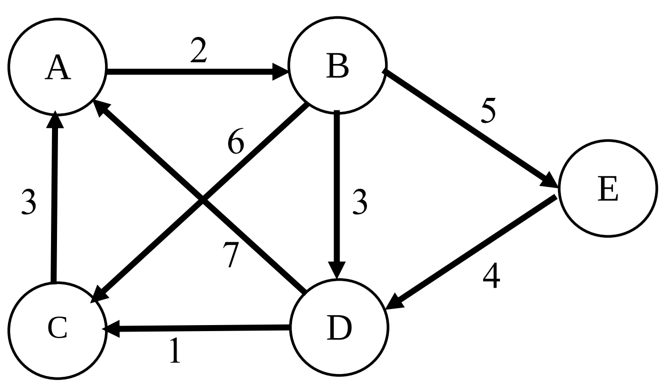
Student Name: ________________________________

First name middle name last name

This final exam (assignment As05) is due at 12:00 noon, Tuesday, December 12, 2023. Please submit your assignment to Brightspace (purdue.brightspace.com). No late turn-in is accepted. Please write your name on the first page of your assignment. (Take 10 off without giving your proper name on the first page). Your file name should be the first character of your first name, followed by your last name, such as PNg.docx. Please number your problem-answer clearly, such as Problem (1.a), (1.b), (1.c), (1.d), (1.e), Problem (2.a), (2.b), …, (4.d). The answers to the problems must be arranged in order. Please answer your questions using only a Word file (.docx file). No pdf file will be accepted. Without using a Word file (.docx file), the submitted problems' answers would not be graded or would take 10 points off. If you attach a pdf page of the solution to your Word file, please leave a few lines blank before your solution page; this allows me to place the cursor for writing comments on your Word file.

The total number of points for this Final Exam (Assignment As_05) is 170.

Given a connected weighted directed graph G(V, E) with its adjacency matrix, which is as follows:



	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	1
C	1	0	0	0	0
D	1	0	1	0	0
E	0	0	0	1	0

(a) Its adjacency matrix, $R^{(0)}$

Figure 1. A graph G(V, E) and its adjacency matrix, $R^{(0)}$.

Problem 1:[50 points]

Warshall's algorithm constructs the transitive closure through a series of n x n boolean matrices: *Compute all the elements of each matrix $R^{(k)}$ from its immediate predecessor $R^{(k-1)}$ in series (1.1), with each intermediate vertex numbered not higher than k.*

$$R^{(0)}, \ldots, R^{(k-1)}, R^{(k)}, \ldots, R^{(n)}. \qquad (1.1)$$

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	1	1
C	1	**1**	0	0	0
D	1	**1**	1	0	0
E	0	0	0	1	0

$R^{(1)}$ uses only A as the intermediate vertex.

C to A to B, then (C, B).

D to A to B, then (D, B)

	A	B	C	D	E
A	1	1	1	1	1
B	1	1	1	1	1
C	1	1	1	1	1
D	1	1	1	1	1
E	1	1	1	1	1

(a) Its transitive closure, $R^{(n)}$

Given graph G(V, E) with its adjacency matrix $R^{(0)}$, as shown in Figure (1), the matrix provides the reachability information. If an entry $r_{ij}^{(0)}$ has 1 in the i^{th} row and j^{th} column of the matrix $R^{(0)}$, then there is a directed edge as a directed path from an i^{th} vertex to j^{th} vertex. For example, there is 1 in row B and C column of the matrix $R^{(0)}$, then B reaches C in graph G. The path from B to C has no intermediate vertex. Since k = 0, no intermediate vertex will be served. According to Warshall's Algorithm, the matrix $R^{(k)}$ is the result computed from the immediate predecessor $R^{(k-1)}$. At most, k number of vertices can serve as the intermediate vertex for the reachability between any two vertices in the given graph G.

(1.a) For the given graph G(V, E), what are the elements (vertices) for computing $R^{(k)}$ for $0 < k$?

(1.b) Given the graph G(V, E) with its adjacency matrix $R^{(0)}$, for k = 3, what are the vertices in G that could serve as the intermediate vertex for defining a directed path of any two vertices?

(1.c) Given the graph G(V, E) with its adjacency matrix $R^{(0)}$, what is the largest value of n of the matrix $R^{(n)}$ that has to be computed for reachability?

(1.d) Using its adjacency matrix $R^{(0)}$ to compute matrix $R^{(n)}$, which column and row are considered?

(1.e) What is the time efficiency for Warshall's algorithm?

Problem 2: [40 points]

For designing a dynamic programming algorithm for the knapsack problem, given n items of known weights w_i with values v_i,

(w_1, v_1) (w_2, v_2), …, (w_n, v_n)

a knapsack of capacity holding maximum weight W, the following formula is used to find the most valuable F(n, W) subset of the items that fit into the knapsack.

$$F(i, j) = \begin{cases} \text{Max}\{ \mathbf{F(i-1, j)}, \; v_i + F(i-1, j-w_i) \}, & \text{if } j - w_i \geq 0. \\ \\ F(i-1, j), & \text{if } j - w_i < 0. \end{cases} \qquad \text{….2.1}$$

with the initial conditions which are as follows:

$$F(0, j) = 0 \text{ for } j \geq 0 \text{ and } F(i, 0) = 0 \text{ for } i \geq 0. \qquad \text{…. 2.2}$$

Given an instance data ID:

item	**weight**	**value**
1	2	\$ 12
2	1	\$ 10
3	3	\$ 20
4	2	\$ 15

Applying formulas (2.1) and (2.2), the following dynamic programming table is filled. The maximal value is F(4, 5) = 37.

capacity $j \leq W=5$		F(i, 0) = 0	**capacity j**				
item i ----- weight j	i \ j	0	1	2	3	4	5
F(0, j) = 0	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	37

(2.a) For this table, how many entries are to be computed? In general, for n items and W Knapsack capacity, how many entries are to be computed?

(2.b) When W is extremely large compared to n, this algorithm is worse than the brute-force algorithm $\theta(2^n)$, which simply considers all subsets. Explain this.

Background Description for the problem (2.c):

To improve the efficiency, the fact that it is unnecessary to determine the entries in the i^{th} row for every w between 1 to W. For computing F(n, W), the only entries needed in the $(n-1)^{st}$ row are the ones F(n-1, W) and F(n-1, W- w_n).

Continue to work backward from n to determine which entries in $(n-1)^{th}$ row are needed. The process stops when n = 1 or $w \leq 0$.

After determining which entries are needed in the i^{th} row, decide which entries are needed in the $(i-1)^{st}$ row using the fact that

F(i, w) is computed from F(i-1, w) and F(i-1, w-w_i).

(2.c) For the given instance data ID, given the following dynamic programming table, determine which entries are needed during the process that reaches the goal (i.e., the maximal value is F(4, 5) = 37). Mark X on those entries that are needed to get the maximal value.

capacity $j \le W=5$		F(i, 0) = 0	**capacity j**				
item i ----- weight j	i \ j	0	1	2	3	4	5
F(0, j) = 0	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0					
$w_2 = 1, v_2 = 10$	2	0					
$w_3 = 3, v_3 = 20$	3	0					
$w_4 = 2, v_4 = 15$	4	0					

(2.d) How many needed entries containing nontrivial values (i.e., not those in row 0 or column 0) must be computed?

Problem 3: [40 points]

Consider the **single-source shortest-paths problem:**

Given a vertex s (called the **source)** in a weighted connected graph G = (V, E),

The application of Dijkstra's Algorithm finds the **shortest paths** to all its other vertices.

The single-source shortest-paths problem looks for a family of paths; each path leads from the source to a different vertex in the graph, and some paths may have edges in common.

The algorithm is ***not*** interested in *a single shortest path* that starts at the source and visits all the other vertices.

Given a connected weighted directed graph G(V, E) in Figure 1.

(3.a) Construct its weighted matrix.

	A	B	C	D	E
A					
B					
C					
D					
E					

(b) Its weight matrix

Description of the problem:

In Dijkstra's Algorithm, it states

for every vertex u in $V - V_T$ that is adjacent to u* **do {**

if $d_{u*} + w(u^*, u) < d_u$

{ $d_u \leftarrow d_{u*} + w(u^*, u)$;

$p_u \leftarrow u^*$

… } //end if

} //end for

(3.b) Complete the following two blanks.

Let the vertex A be the source.

$$d_{u^*} + w(u^*, u) < d_u$$

Tree vertices	Remaining vertices	Illustration
		$V_T = \{\emptyset\}$ $V - V_T = \{A, B, C, D, E\}$
A(-, 0)	min{B(A, 2)}	$V_T = \{A\}$ $V - V_T = \{B, C, D, E\}$
B(A, 2)	min{D(B, 3+2), E(B, 5+2), C(B, 6+2)}	$V_T = \{A, B\}$ $V - V_T = \{C, D, E\}$
D(B, 5)	______________________________	$V_T = \{A, B, D\}$ $V - V_T = \{C, E\}$
C(D, 6)	______________________________	$V_T = \{A, B, D, C\}$ $V - V_T = \{E\}$
E(B, 7)		$V_T = \{A, B, D, C, E\}$ $V - V_T = \{\emptyset\}$

(3.c) From the solution of (3.b), How many shortest paths from the single source, vertex A, do you obtain? List the shortest paths to all its other vertices from a given source, vertex A, with their weights.

(3.d) Consider the statement $d_{u*} + w(u*, u) < d_u$. Choose one of the Items for each of the blanks.

Items = { d_{u*}, d_u , u*, u, w(u*, u) }.

(i) ___________ A vertex in V- V_T and is currently considered.

(ii) ___________ A vertex in V_T and is previously selected.

(iii) ___________ the weight for edges between u* and u

(iv) ___________ the weight of a path from a vertex in V_T to the distinctive source.

(v) ____________ the weight of a path from a vertex in V- V_T to the distinctive source.

Problem 4: [40 points]

Given a connected weighted directed graph G(V, E) in Figure 1, the application of Prim's Algorithm constructs a minimum spanning tree through a sequence of expanding subtrees.

Let Y(X, w) be the term "X reaches Y with weight w." That means "there is a directed edge from X to Y with weight w."

(4.a) Complete the following blank with answers written in the form Y(X, w), where X is in V_T and Y is in V- V_T.

Tree vertices	Remaining vertices	Illustration
		$V_T = \{\emptyset\}$ $V - V_T = \{A, B, C, D, E\}$
A(-, -)	**B(A, 2),** C(-, ∞), D(-, ∞), E(-, ∞)	$V_T = \{A\}$ $V - V_T = \{B, C, D, E\}$
B(A, 2)	**D(B, 3),** E(B, 5), C(B, 6)	$V_T = \{A, B\}$ $V - V_T = \{C, D, E\}$
D(B, 3)	______________________	$V_T = \{A, B, D\}$ $V - V_T = \{C, E\}$
C(D, 1)	E(B, 5)	$V_T = \{A, B, D, C\}$ $V - V_T = \{E\}$
E(B, 5)		$V_T = \{A, B, D, C, E\}$ $V - V_T = \{\emptyset\}$

(4.b) What is the minimum spanning tree of weighted, directed graph G(V, E) given in Figure 1? What is the total weight of this minimum spanning tree?

(4.c) Give why Prim's Algorithm does not have to check whether to form a cycle by adding a newly found edge Y(X, w) to the immediate spanning tree.

(4.d) Does Prim's algorithm always yield a minimum spanning tree?