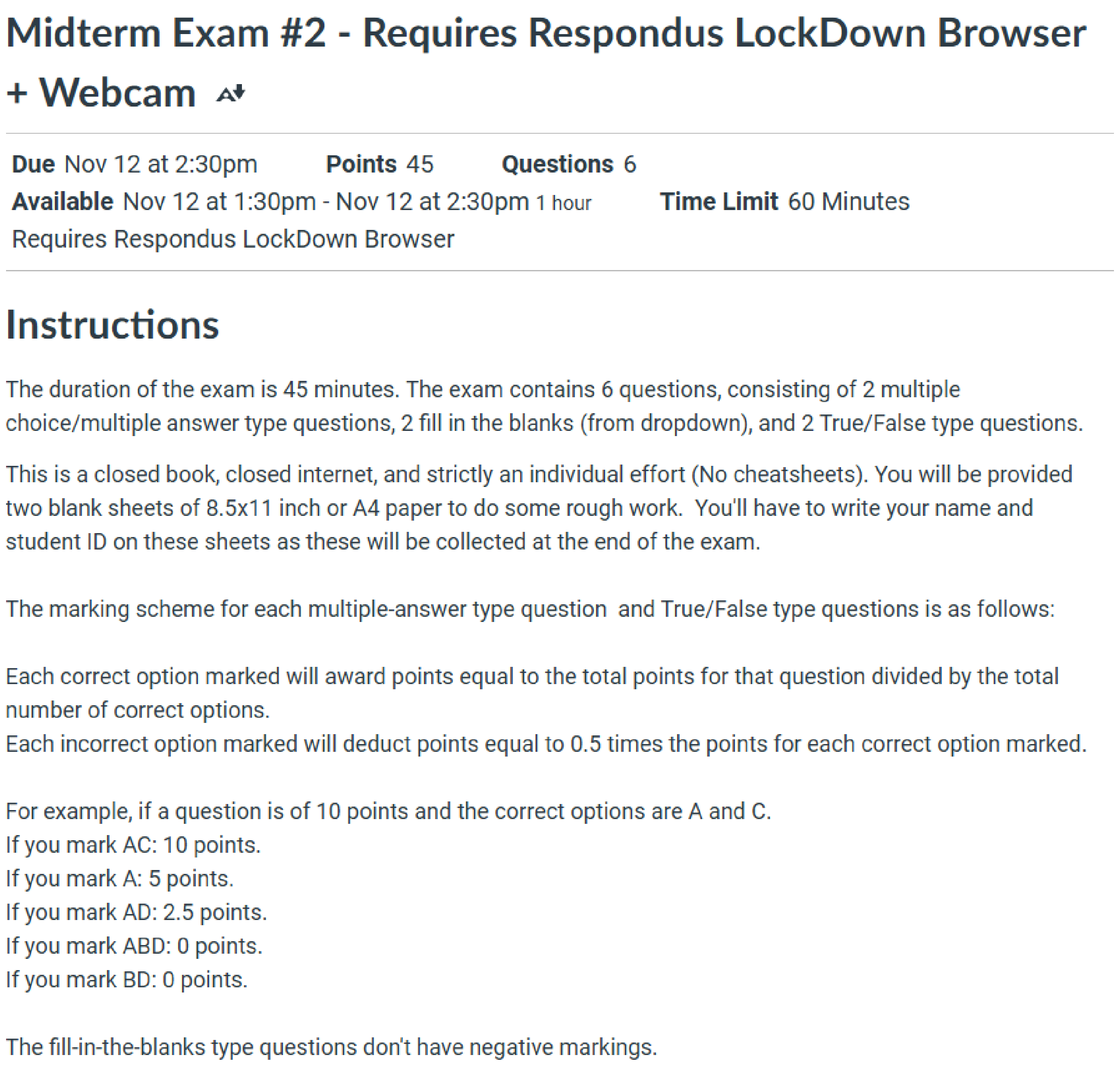
Foundations of Algorithms by Arunabha Sen, ASU

Midterm 2 (Fall 2024)

Instructions



Midterm Exam #2 - Requires Respondus LockDown Browser + Webcam

Due Nov 12 at 2:30pm **Points** 45 **Questions** 6
Available Nov 12 at 1:30pm - Nov 12 at 2:30pm 1 hour **Time Limit** 60 Minutes
Requires Respondus LockDown Browser

Instructions

The duration of the exam is 45 minutes. The exam contains 6 questions, consisting of 2 multiple choice/multiple answer type questions, 2 fill in the blanks (from dropdown), and 2 True/False type questions.

This is a closed book, closed internet, and strictly an individual effort (No cheatsheets). You will be provided two blank sheets of 8.5x11 inch or A4 paper to do some rough work. You'll have to write your name and student ID on these sheets as these will be collected at the end of the exam.

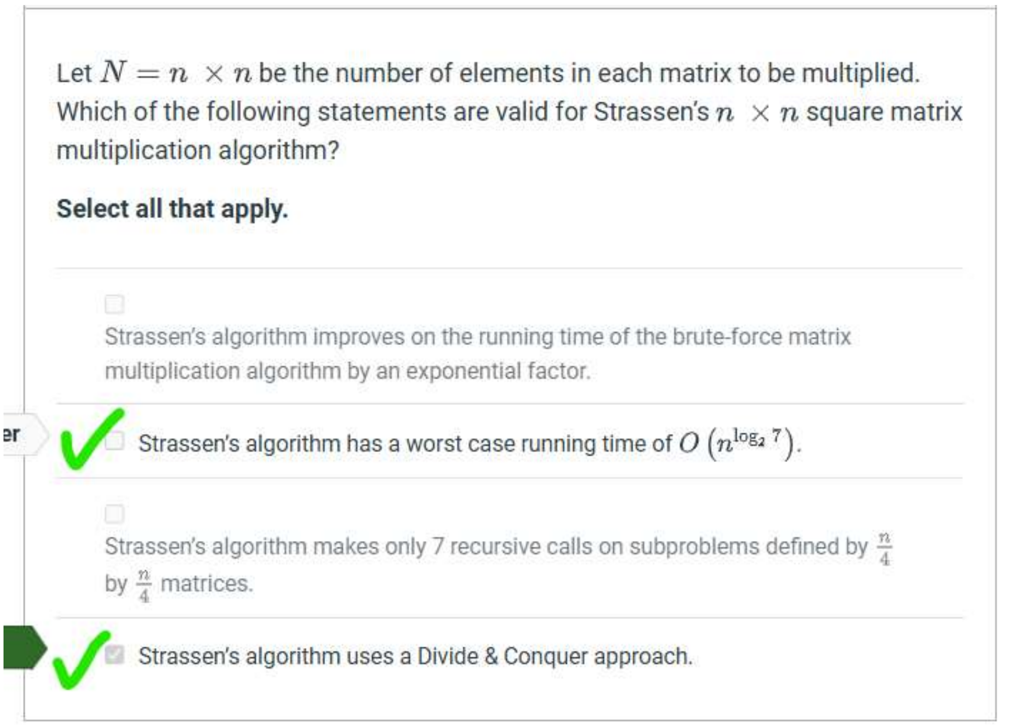
The marking scheme for each multiple-answer type question and True/False type questions is as follows:

Each correct option marked will award points equal to the total points for that question divided by the total number of correct options.
Each incorrect option marked will deduct points equal to 0.5 times the points for each correct option marked.

For example, if a question is of 10 points and the correct options are A and C.
If you mark AC: 10 points.
If you mark A: 5 points.
If you mark AD: 2.5 points.
If you mark ABD: 0 points.
If you mark BD: 0 points.

The fill-in-the-blanks type questions don't have negative markings.

Q-1 Strassen's Multiplication Algorithm [Refer Module 4 -> Lesson 3 -> Video 3: Fast Matrix Multiplication]

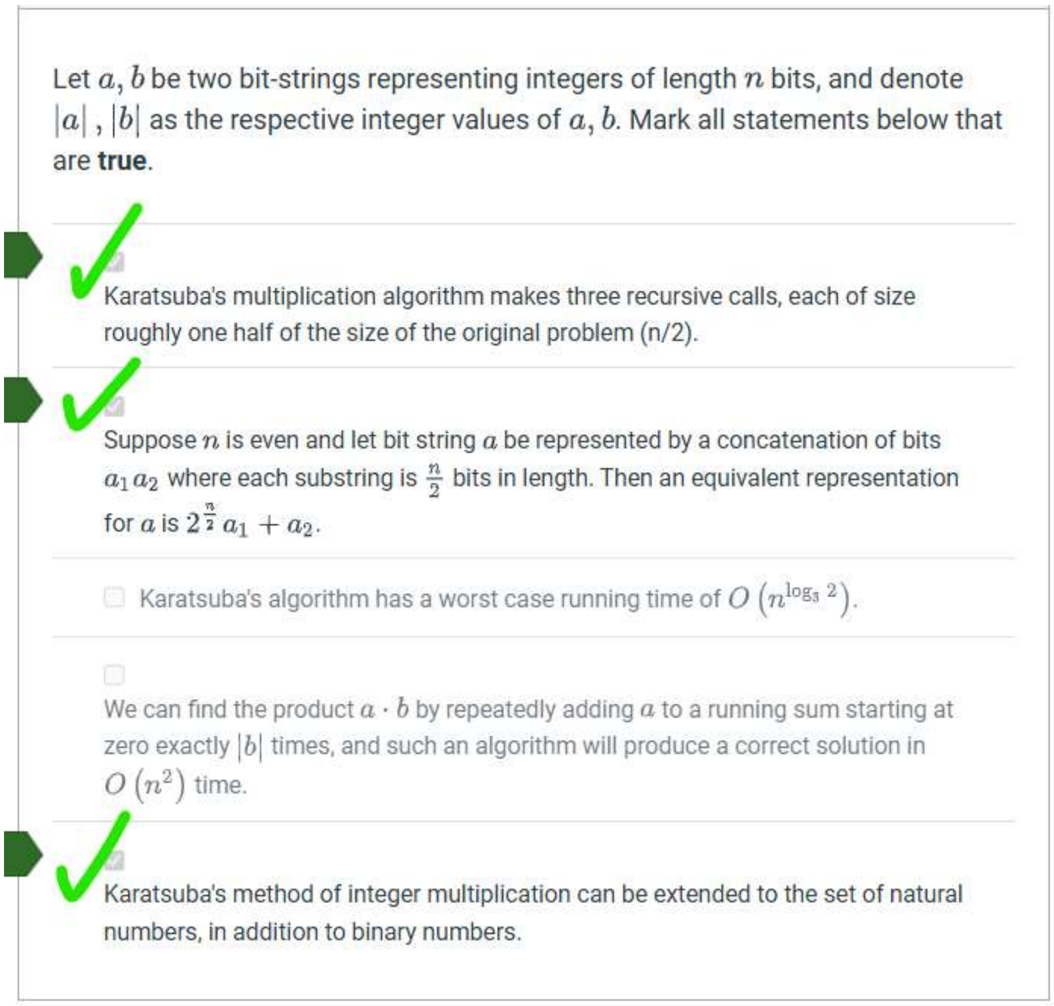


Let $N = n \times n$ be the number of elements in each matrix to be multiplied. Which of the following statements are valid for Strassen's $n \times n$ square matrix multiplication algorithm?

Select all that apply.

- [ ] Strassen's algorithm improves on the running time of the brute-force matrix multiplication algorithm by an exponential factor.
- [✓] Strassen's algorithm has a worst case running time of $O\left(n^{\log_2 7}\right)$.
- [ ] Strassen's algorithm makes only 7 recursive calls on subproblems defined by $\frac{n}{4}$ by $\frac{n}{4}$ matrices.
- [✓] Strassen's algorithm uses a Divide & Conquer approach.

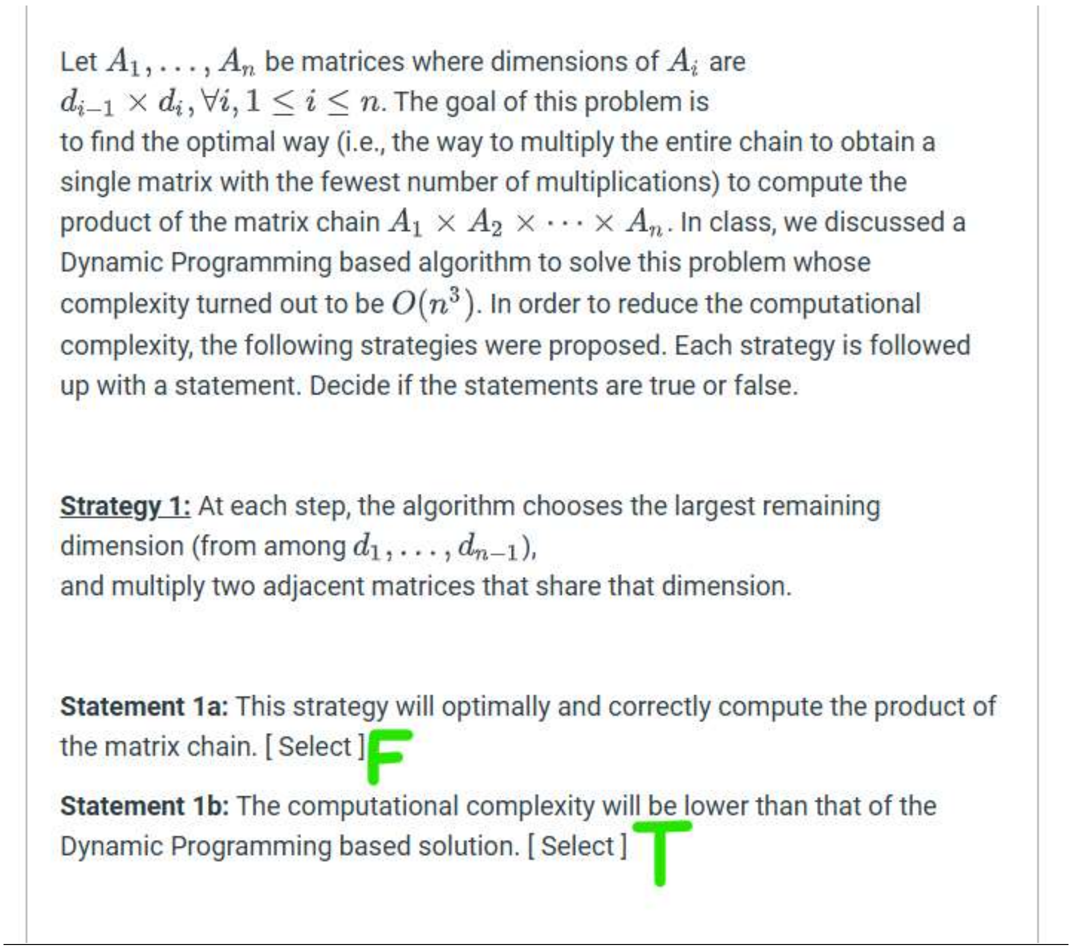
Q-2 Karatsuba Multiplication [Refer Module 4 -> Lesson 3 -> Video 2: Karatsuba Multiplication]



Let a, b be two bit-strings representing integers of length n bits, and denote $|a|$, $|b|$ as the respective integer values of a, b. Mark all statements below that are **true**.

- [✓] Karatsuba's multiplication algorithm makes three recursive calls, each of size roughly one half of the size of the original problem (n/2).
- [✓] Suppose n is even and let bit string a be represented by a concatenation of bits $a_1 a_2$ where each substring is $\frac{n}{2}$ bits in length. Then an equivalent representation for a is $2^{\frac{n}{2}} a_1 + a_2$.
- [ ] Karatsuba's algorithm has a worst case running time of $O\left(n^{\log_3 2}\right)$.
- [ ] We can find the product $a \cdot b$ by repeatedly adding a to a running sum starting at zero exactly $|b|$ times, and such an algorithm will produce a correct solution in $O\left(n^2\right)$ time.
- [✓] Karatsuba's method of integer multiplication can be extended to the set of natural numbers, in addition to binary numbers.

Q-3 MCM

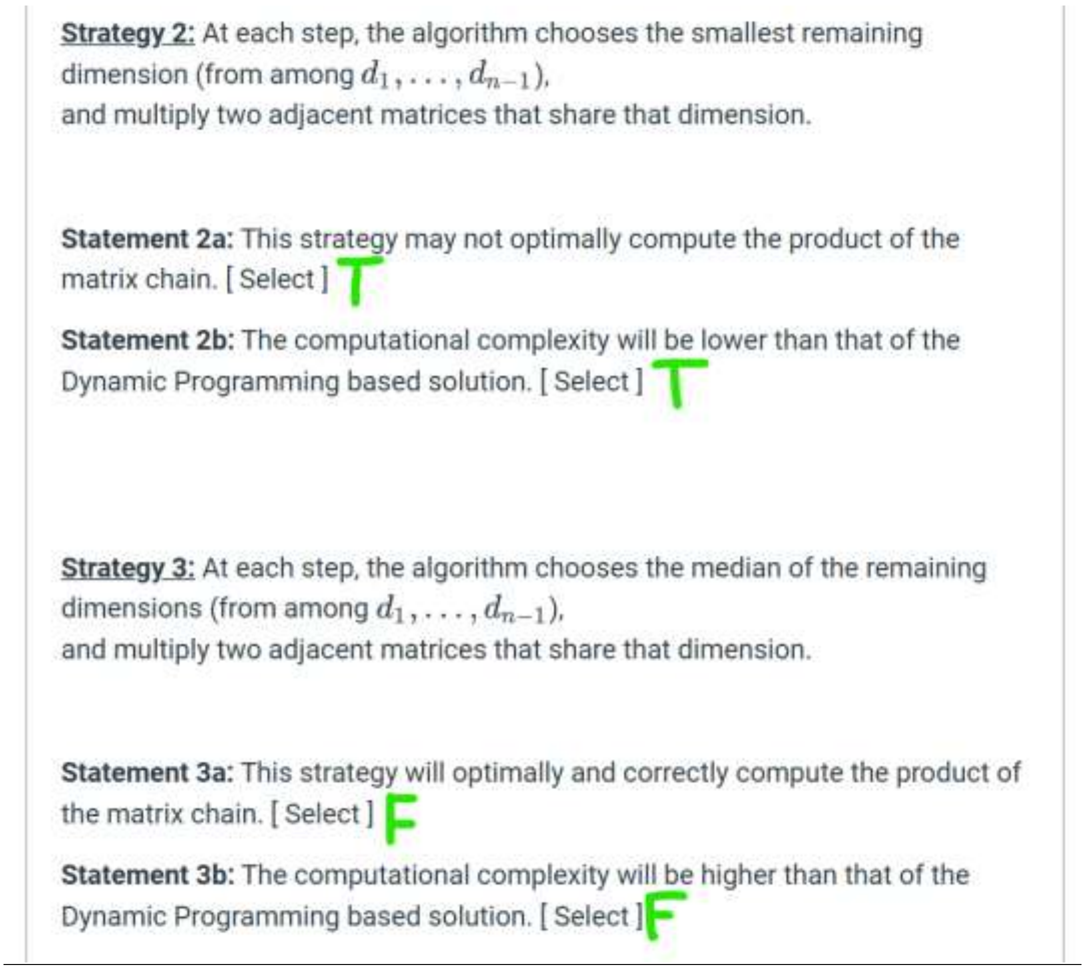


Let $A_1, \ldots, A_n$ be matrices where dimensions of A_i are $d_{i-1} \times d_i, \forall i, 1 \leq i \leq n$. The goal of this problem is to find the optimal way (i.e., the way to multiply the entire chain to obtain a single matrix with the fewest number of multiplications) to compute the product of the matrix chain $A_1 \times A_2 \times \cdots \times A_n$. In class, we discussed a Dynamic Programming based algorithm to solve this problem whose complexity turned out to be $O(n^3)$. In order to reduce the computational complexity, the following strategies were proposed. Each strategy is followed up with a statement. Decide if the statements are true or false.

Strategy 1: At each step, the algorithm chooses the largest remaining dimension (from among $d_1, \ldots, d_{n-1}$), and multiply two adjacent matrices that share that dimension.

Statement 1a: This strategy will optimally and correctly compute the product of the matrix chain. [Select] F

Statement 1b: The computational complexity will be lower than that of the Dynamic Programming based solution. [Select] T



Strategy 2: At each step, the algorithm chooses the smallest remaining dimension (from among $d_1, \ldots, d_{n-1}$), and multiply two adjacent matrices that share that dimension.

Statement 2a: This strategy may not optimally compute the product of the matrix chain. [Select] T

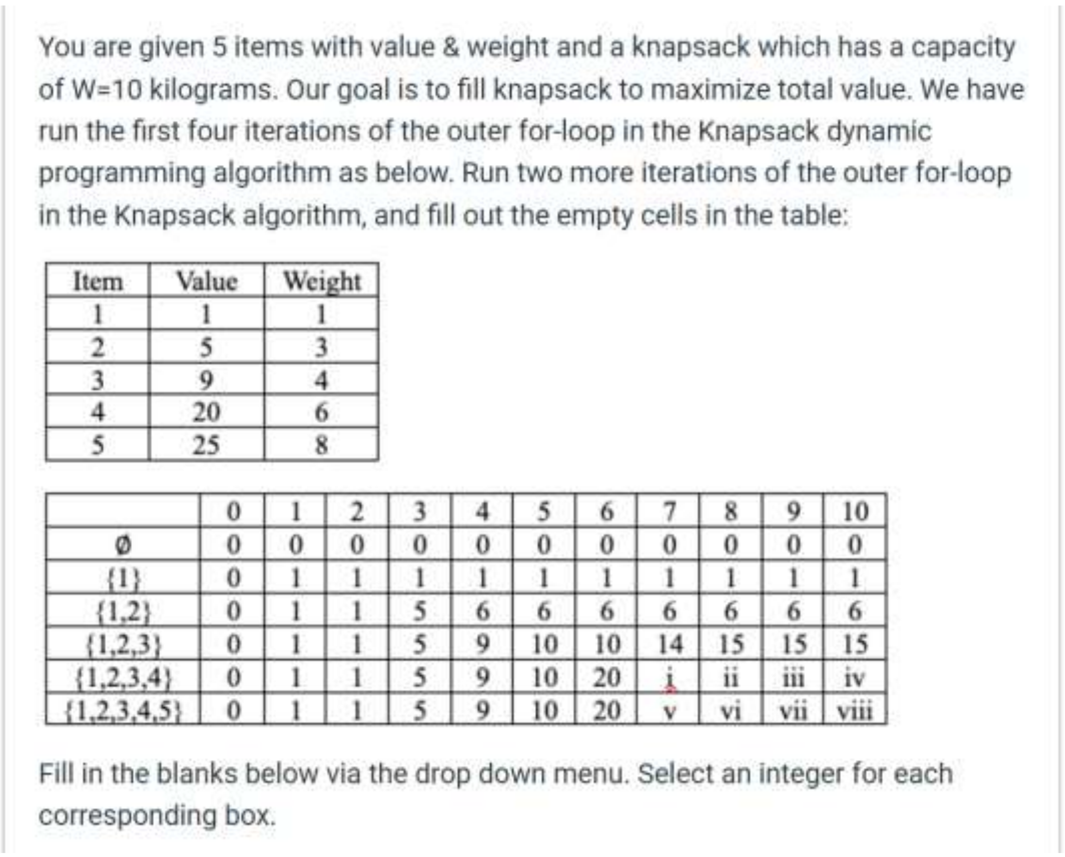
Statement 2b: The computational complexity will be lower than that of the Dynamic Programming based solution. [Select] T

Strategy 3: At each step, the algorithm chooses the median of the remaining dimensions (from among $d_1, \ldots, d_{n-1}$), and multiply two adjacent matrices that share that dimension.

Statement 3a: This strategy will optimally and correctly compute the product of the matrix chain. [Select] F

Statement 3b: The computational complexity will be higher than that of the Dynamic Programming based solution. [Select] F

Q-4 01-Knapsack DP [Refer Module 5 -> Lesson 2 -> The Knapsack Problem]



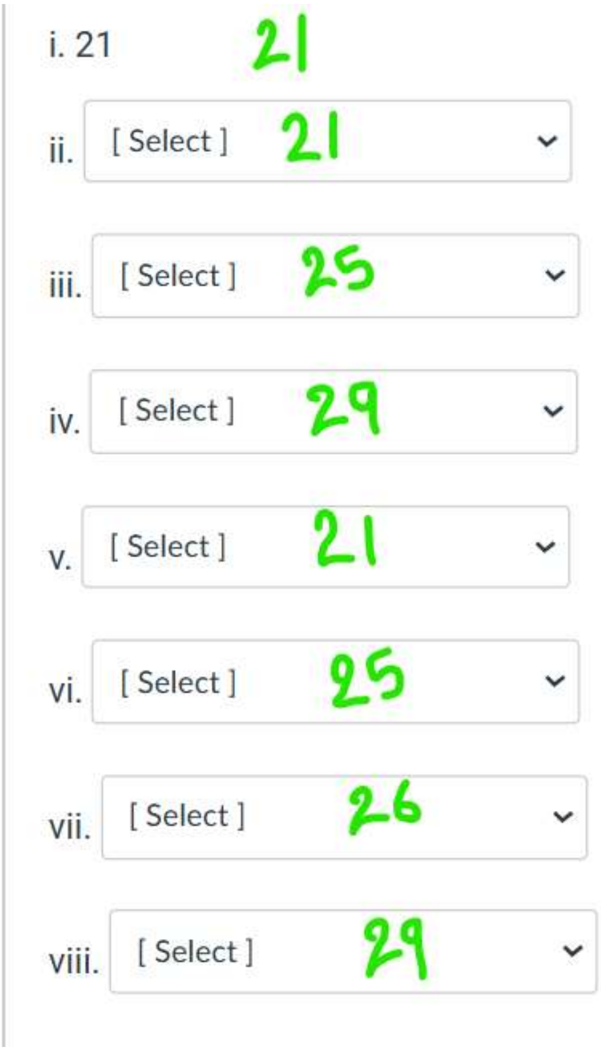
You are given 5 items with value & weight and a knapsack which has a capacity of W=10 kilograms. Our goal is to fill knapsack to maximize total value. We have run the first four iterations of the outer for-loop in the Knapsack dynamic programming algorithm as below. Run two more iterations of the outer for-loop in the Knapsack algorithm, and fill out the empty cells in the table:

Item	Value	Weight
1	1	1
2	5	3
3	9	4
4	20	6
5	25	8

	0	1	2	3	4	5	6	7	8	9	10
Ø	0	0	0	0	0	0	0	0	0	0	0
{1}	0	1	1	1	1	1	1	1	1	1	1
{1,2}	0	1	1	5	6	6	6	6	6	6	6
{1,2,3}	0	1	1	5	9	10	10	14	15	15	15
{1,2,3,4}	0	1	1	5	9	10	20	i	ii	iii	iv
{1,2,3,4,5}	0	1	1	5	9	10	20	v	vi	vii	viii

Fill in the blanks below via the drop down menu. Select an integer for each corresponding box.

Ans:



i. 21 21

ii. [Select] 21

iii. [Select] 25

iv. [Select] 29

v. [Select] 21

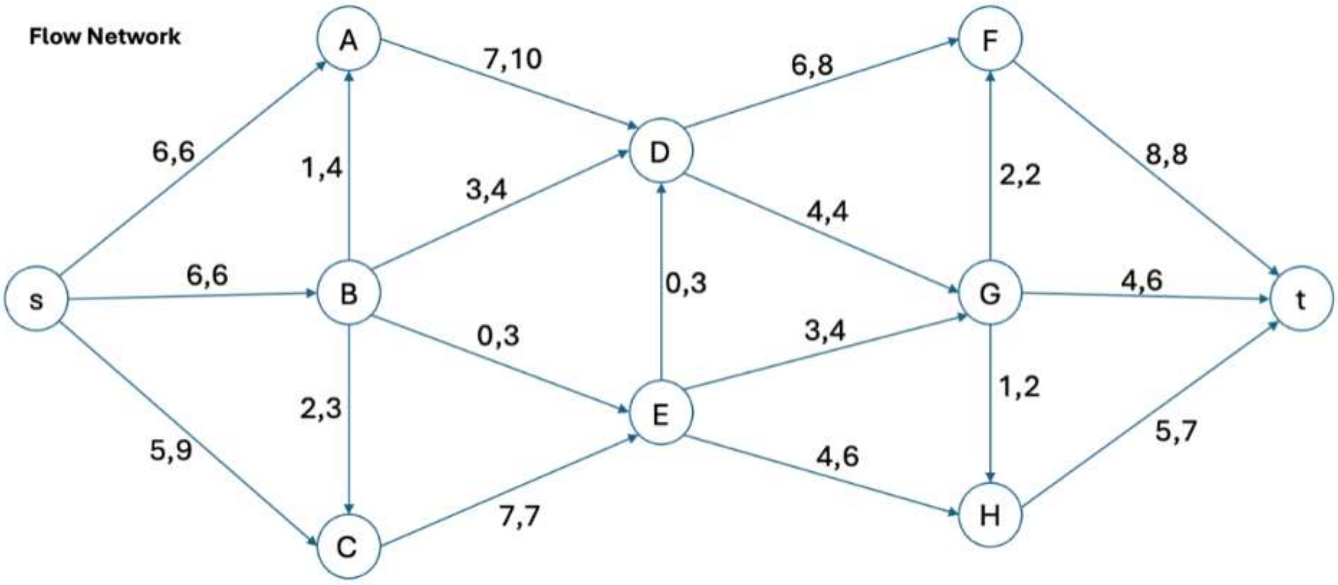
vi. [Select] 25

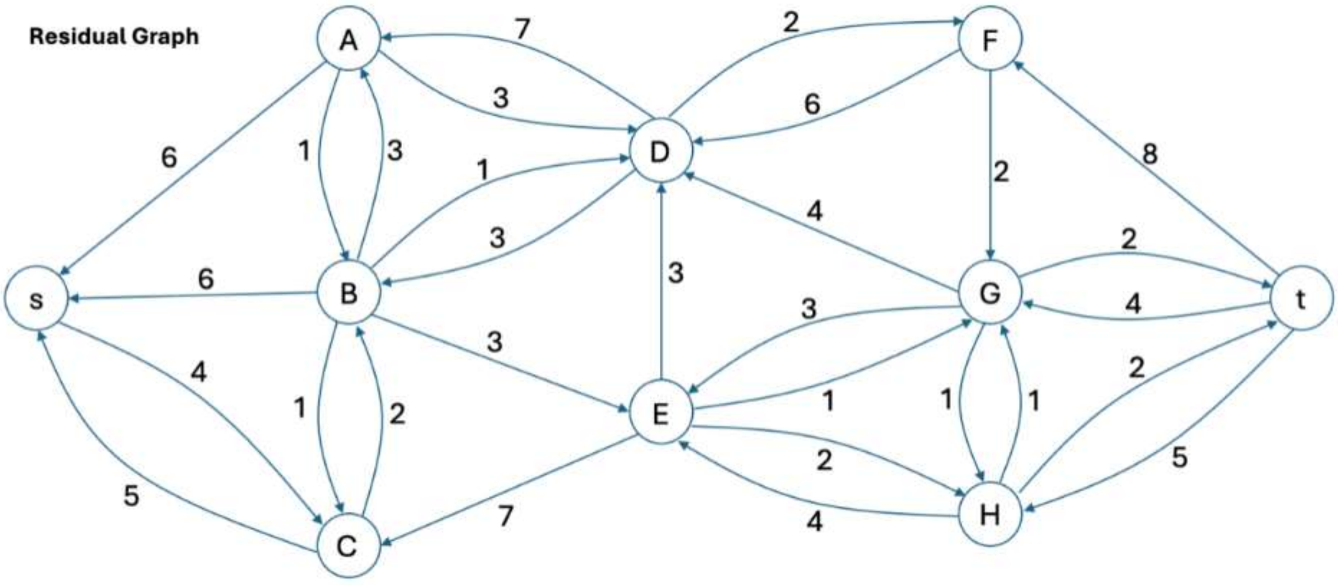
vii. [Select] 26

viii. [Select] 29

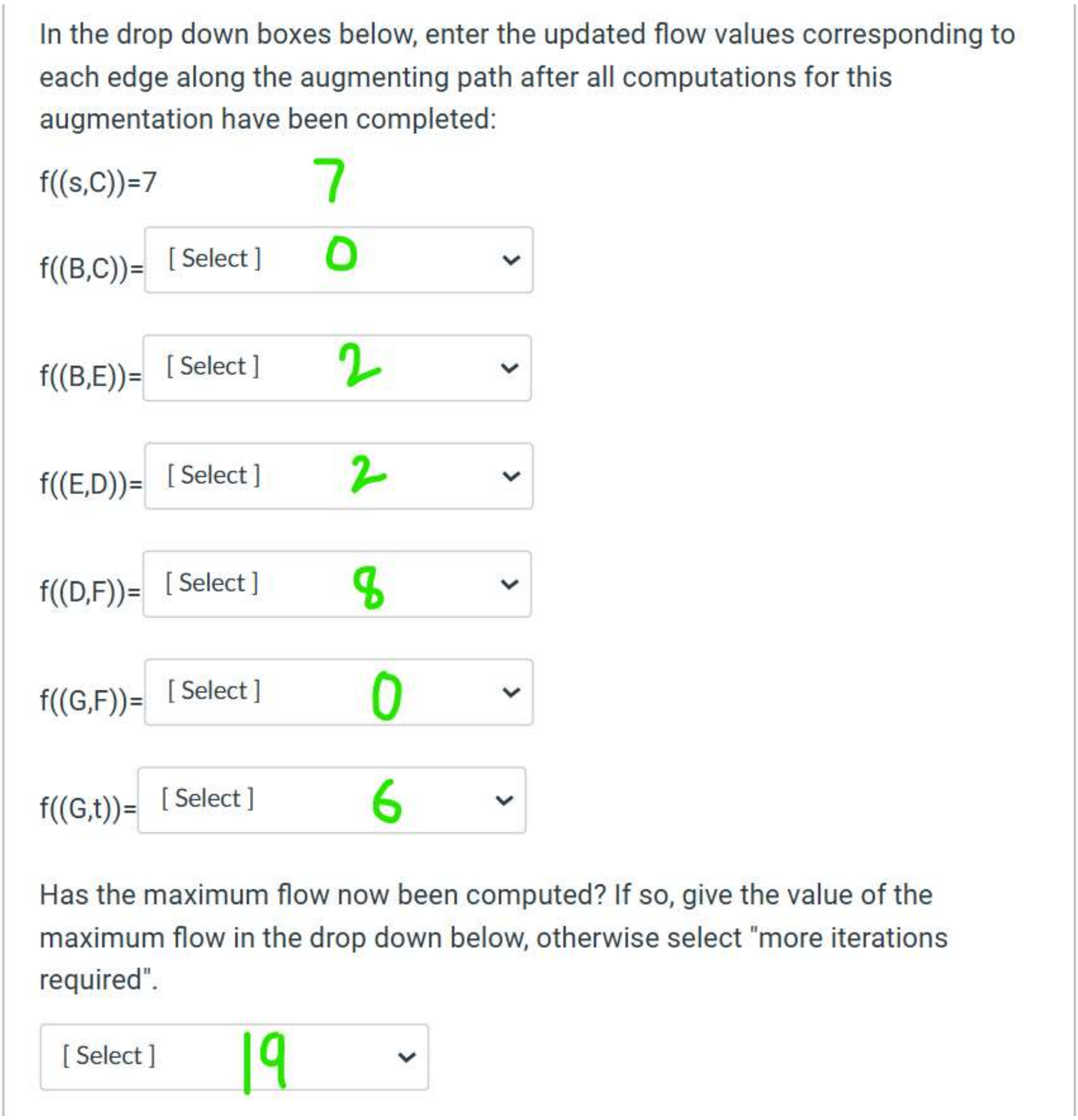
Q-5 Network Flow [Refer Module 6: Network Flows]

> You are given the following flow network with source node s and sink node t as it stands in the middle of an execution of the Ford-Fulkerson algorithm after five paths have been augmented. You are given the corresponding residual graph at the same iteration as well. Recall that we are trying to calculate the maximum flow value for this network. Suppose the next path the algorithm chooses to augment is the chain of nodes $\mathrm{P{=}s,C,B,E,D,F,G,t}$. Give the updated flow values for each edge present in P after the augmentation is completed. Note that edge weights in the flow network are given as f, c where f is the flow value assigned to that edge and c is the capacity of the edge. The edge weights of the residual graph represent residual capacities only. Also, recall the function $\mathrm{f((x,y))}$ denotes the flow of edge (x, y).





Ans:



In the drop down boxes below, enter the updated flow values corresponding to each edge along the augmenting path after all computations for this augmentation have been completed:

f((s,C))=7 7

f((B,C))= [Select] 0

f((B,E))= [Select] 2

f((E,D))= [Select] 2

f((D,F))= [Select] 8

f((G,F))= [Select] 0

f((G,t))= [Select] 6

Has the maximum flow now been computed? If so, give the value of the maximum flow in the drop down below, otherwise select "more iterations required".

[Select] 19