CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651: Algorithm Design Techniques

Winter 2022

MidTerm - Close book exam - 2:30 hours

Instructor: Professor B. Jaumard

First Name	**Last Name**	**ID#**

	a.	**b.**	**c.**	
Question 1	5 (reasonable attempt) 10 (some good idea) 15 (exact recurrence relation)	5 points		20 points
Question 2	5	4	16	25
Question 3	15	10		25
Question 4	13	12		25
Question 5	25			25
Total				

Question 1. Dynamic Programming (20 points.)

Samantha is cooking from her garden, which is arranged in grid with n rows and m columns. Each cell $(i, j)(1 \leq i \leq n, 1 \leq j \leq m)$ has an ingredient growing in it, with tastiness given by a positive value T_{ij}. To prepare a dinner, Samantha stands at a cell (i, j) and pick one ingredient from each quadrant relative to that cell. The tastiness of her dish is the product of the tastiness of the four ingredients she chooses. Help Samantha find an $O(nm)$ dynamic programming algorithm to maximize the tastiness of her dish.

The four quadrants relative to a cell (i, j) are defined as follows:

TL - top-left = all cells (a, b) such that $a < i, b < j$,

BL - bottom-left = all cells (a, b) such that $a > i, b < j$,

TR - top-right = all cells (a, b) such that $a < i, b > j$,

BR - bottom-right = all cells (a, b) such that $a > i, b < j$.

Because Samantha needs all four quadrants to be non-empty, she can only stand on cells (i, j) where $1 < i < n$ and $1 < j < m$.

(a) Define TL_{ij} to be maximum tastiness value in the top-left quadrant of cell (i, j):

$$\text{TL}_{ij} = \max\{T_{ab} : 1 \leq a \leq i, 1 \leq b \leq j\}.$$

Find a dynamic programming algorithm to compute TL_{ij}, for all $1 < i < n$ and $1 < j < m$, in $O(nm)$ time. While writing the algorithm, explain clearly how you proceed with the initialization of the values. What is the space complexity requirement?

(b) Use the idea in part (a) to obtain an $O(nm)$ algorithm to find the tastiest dish. Again, what is the space complexity requirement?

Exercise and Solution taken from: `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/exams/MIT6_046JS15_quiz1sols.pdf`

Question 2. Graph algorithm (25 points.)

On the the figure below is flow network (G, c) with s = source node at the top left corner and t = sink at the lower right. On each arc ℓ, the first value is a flow value φ_ℓ, followed by the link capacity value between square brackets. Verify for yourself that the flow satisfies the conservation conditions.

(a) Draw the residual graph (G^R_φ, c_φ).

(b) Using the residual graph, explain how to check whether the flow is optimal. If the flow is not optimal, explain what are the steps of the Ford-Fulkerson algorithm to perform in order to compute an optimal flow. Compute an optimal flow.

Answer (b) 4 points There is no path from source to sink in the residual graph $\rightsquigarrow$ flow is optimal according to Fored-Fulkerson's algorithm

(c) Assume that the link capacity of $(7, \text{sink})$ increases by two units. Using the algorithm of Ford Fulkerson, compute an optimal flow (indicate its components on each arc and its optimal value), starting form an optimal flow of the previous question.

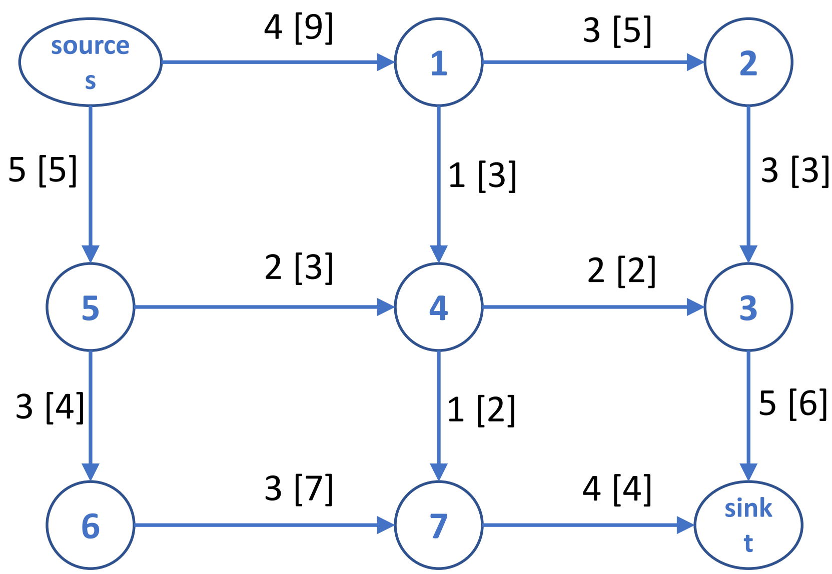


Figure 1: Original flow

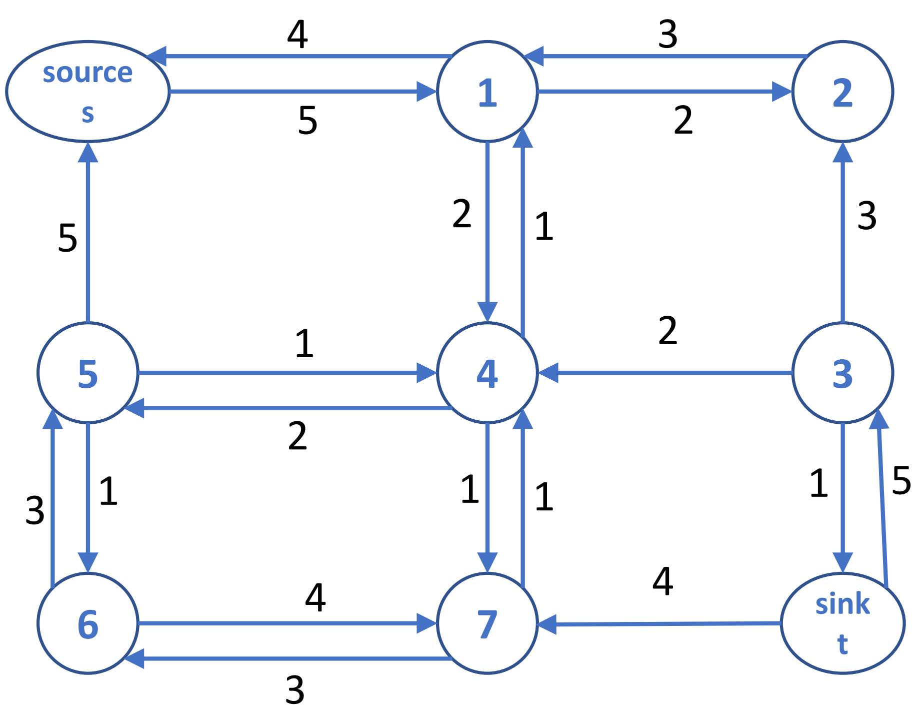
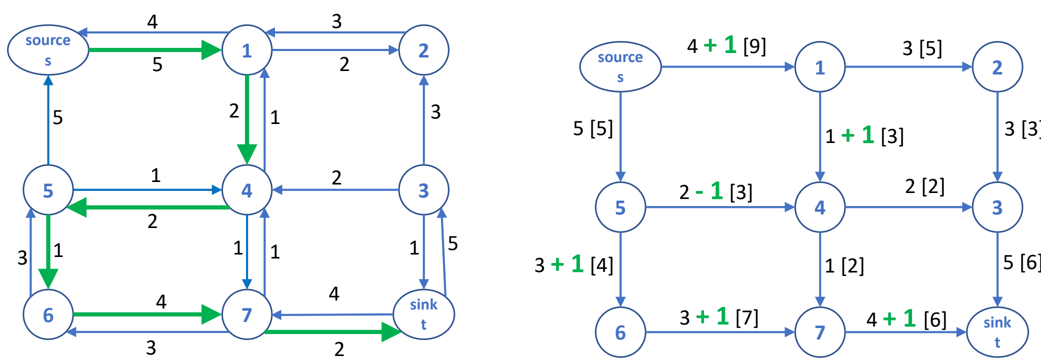
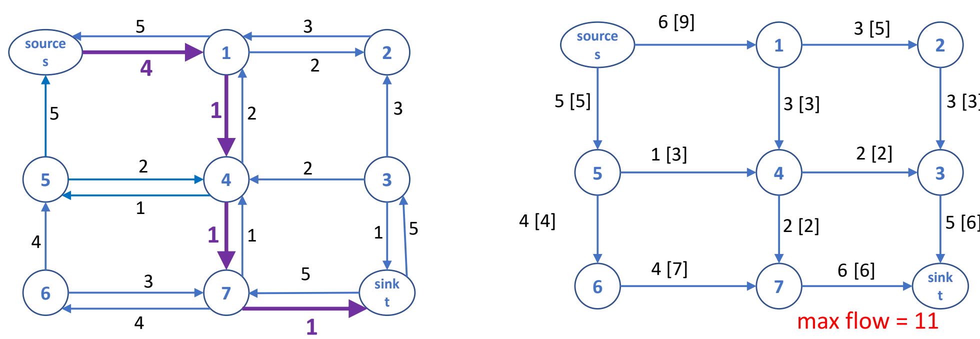


Figure 2: Question (a): Residual graph **(5 points)**



(a) Residual Graph. (b) Updated Flow.

Figure 3: Question (c): Residual graph #1 **(5 points)**, Resulting updated flow on original graph **(3 points)**



(a) Residual Graph. (b) Updated Flow.

Figure 4: Question (c): Residual graph #2 **(5 points)**, Resulting updated flow on original graph **(3 points)**

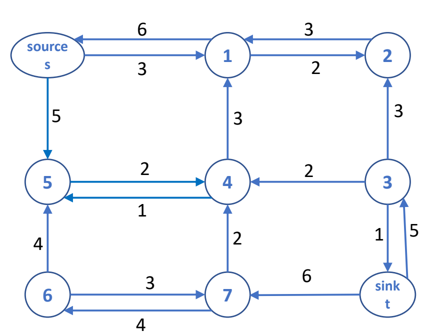


Figure 5: Question (c): Residual graph #3: no path from source to sink

Question 3. Amortized Analysis (25 points.)

Design a data structure to maintain a set S of n distinct integers that supports the following two operations:

1. INSERT(x, S): insert integer x into S.
2. REMOVE-BOTTOM-HALF(S): remove the smallest $\lfloor n/2 \rfloor$ integers from S.

(a) Describe your data structure and your algorithm. Give the worse-case time complexity of the two operations.

(b) Carry out an amortized analysis with an arbitrary sequence of operations combining INSERT(x, S) and REMOVE-BOTTOM-HALF(S). Show that it is possible to do it in such a way as to run INSERT(x, S) in amortized $O(1)$ time, and REMOVE BOTTOM-HALF(S) run in amortized 0 time.

Solution:
3 points. Use a linked list to store those integers.
1 point. If you propose a data structure which is not a good fit for the requested complexities
0 point. If you did not suggest any data structure.

If you indicated a stack or multiple stacks, you did not get points for algorithms/complexities, as this is a quite bad answer for a problem that requires some kind of sorting. Indeed, in order to identify the bottom-half, no sorting is required, but only the computation of the median.

INSERT(x, S). **5 points. 3 points for the algorithm, 2 points for the complexity.**

REMOVE-BOTTOM HALF(S). **7 points. 5 points for the algorithm, 2 points for the complexity.**

Amortized analysis. **10 points.**

Exercise and Solution taken from: `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/exams/MIT6_046JS15_quiz1sols.pdf`

Question 4. (25 points)

A party of n people have come to dine at a fancy restaurant and each person has ordered a different item from the menu. Let $D_1, D_2, \cdots, D_n$ be the items ordered by the diners. Since this is a fancy place, each item is prepared in a two-stage process. First, the head chef (there is only one head chef) spends a few minutes on each item to take care of the essential aspects and then hands it over to one of the many sous-chefs to finish off. Assume that there are essentially an unlimited number of sous-chefs who can work in parallel on the items once the head chef is done. Each item D_i takes h_i units of time for the head chef followed by s_i units of time for the sous-chef (the sous-chefs are all identical). The diners want all their items to be served at the same time which means that the last item to be finished defines the time when they can be served. The goal of the restaurant is to serve the diners as early as possible.

Consider a greedy algorithms that order the items according to one of the different criteria.

(a) Identify the ordering that yields an optimum solution and provide the proof of its optimality

- **4 points for the identification of the proper criterion**
- **7 points for the proof of optimality**

(b) For two other cases, describe a counter example that shows that the order does not yield an optimum solution.

- **7 points for each counter example**

1. Order the items in increasing order of $h_i + s_i$.
2. Order the items in decreasing order of $h_i + s_i$.
3. Order the items in increasing order of h_i.
4. Order the items in decreasing order of h_i.
5. Order the items in increasing order of s_i.
6. Order the items in decreasing order of s_i.

Exercise and Solution taken from:
`https://courses.engr.illinois.edu/cs374/sp2020/B/labs/lab_12_a.pdf`

Question 5. Recurrence relation. (25 points)

Solve the following recurrence relation using the technique with the characteristic equation. Provide the complete analytic expression of the solution.

$$d_1 = 1 \quad \text{and} \quad d_n = d_{n/2} + 1, \qquad (n \geq 2).$$

You must not use the master theorem to solve it, but the method that requires going through the solution of the characteristic equation.

Solution:
See Lecture 1.

Linearization. 5 points

$$d_1 = 1 \quad \text{and} \quad d_n = d_{n/2} + 1, \qquad (n \geq 2)$$

Nonlinear recurrence equation !

Linearized using a change of variable: $n = 2^k \quad \hookrightarrow \quad a_k = d_{2^k}$

$$a_k = a_{k-1} + 1 \qquad k \geq 1, \qquad a_0 = 1$$

This is a linear non homogeneous recurrence equation with $f(n) = 1$

Step 1. 5 points Homogeneous part

1a. Homogeneous part: $a_k - a_{k-1} = 0$

1b. Characteristic Equation: $r - 1 = 0$, $r = 1$

1c. $a_k^{(h)} = C_1(r)^k = C_1(1)^k = C_1$ is the generic solution for the homogeneous part

Step 2. 5 points Particular solution

2a. A particular solution: $a_k^{(p)} = Bk + C$

2b. Constant values of the particular solution: $a_k^{(p)}$ must satisfy: $a_k^{(p)} = a_{k-1}^{(p)} + 1$,
i.e., $Bk + C = B(k-1) + C + 1$
$\rightsquigarrow B = 1$, we take $C = 0$. $a_k^{(p)} = k$

Step 3. 1 point General solution

3. $a_k = a_k^{(h)} + a_k^{(p)} = k + C_1$

Step 4. 4 points Initial conditions

4a. Satisfying initial condition $a_0 = 1 = 0 + C_1$, $C_1 = 1$.

4b. Finally $a_k = k + 1$.

Step 5. 5 points Coming back to n indexing.

4c. $d_{2^k} = k + 1$, recall that $n = 2^k$, i.e, $k = \log_2 n$

4d. $d_n = \log_2 n + 1 = O(\log n)$