# Technical University of Denmark

Written exam, December 11, 2020.

Course name: Algorithms and data structures II.

Course number: 02110.

Aids allowed: All aids allowed.

Exam duration: 4 hours

Weighting: Question 1: 29% - Question 2: 12% - Question 3: 15% - Question 4: 26% - Question 5: 18%

The weighting is only an approximative weighting.

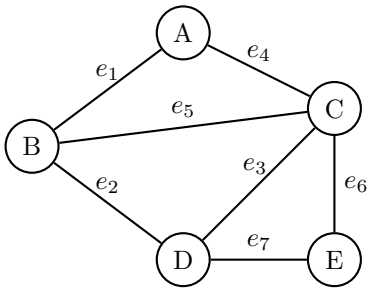
You can answer the exam in either Danish or English.

**All questions should be answered by filling out the box below the question.**

**As exam paper just hand in this and the following pages filled out.**

# Question 1

**Question 1.1 (9%)** Consider the unweighted graph $G$ below.




**Question 1.1.1** What is the size of a minimum cut in the graph? Give a minimum cut as a partitioning of the vertices. How many minimum cuts are there?

Solution:

Size of a minimum cut: _____

Minimum cut (partition of the vertices): _____

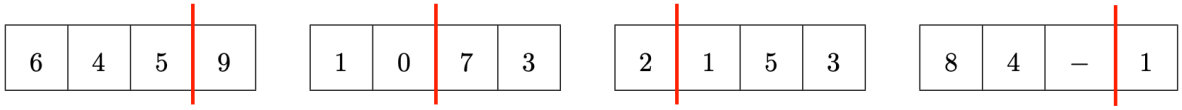
Number of minimum cuts: _____

**Question 1.1.2** Consider running the randomized Min-Cut algorithm on the graph $G$. Assume we first contract edge $e_3$. What is the probability that $B$ and $C$ are in the same contracted node after the next step? Explain your answer.

Solution:

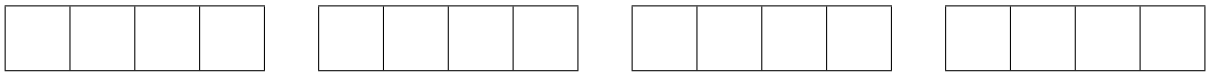
Probability: _____

**Question 1.3 (7%)** Consider a 2-level rotated array data structure below.




Show the result of applying the operation INSERT(7, 8) and then the operation DELETE(2) in that order. (Recall that INSERT($i$, $x$) inserts a new entry with value $x$ immediately *to the right* of entry $i$.)

Solution:



**Question 1.4 (6%)** A sequence of $n$ operations is performed on a data structure. The cost $T(i)$ of the $i$th operation is:

$$T(i) = \begin{cases} i^2 & \text{if } i \text{ is a power of 2} \\ 1 & \text{otherwise} \end{cases}$$

The worst case cost of an operation is:

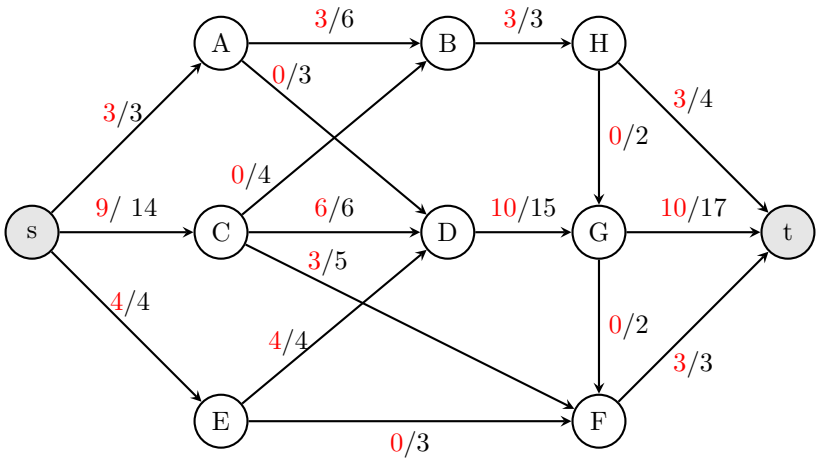
A $\Theta(1)$ B $\Theta(n)$ C $\Theta(n^2)$ D $\Theta(2^n)$ E $\Theta(2^{2n})$

The amortized cost of an operation is:

A $\Theta(1)$ B $\Theta(\sqrt{n})$ C $\Theta(n)$ D $\Theta(n^2)$ E $\Theta(2^n)$

# Question 2 (12%)

Consider the network $N$ below with capacities and flow on the edges. The red numbers are the flow values and the black are the capacities.




**Question 2.1 (7%)** Is this a maximum $s-t$ flow? If not give an augmenting path (write the nodes of the path).

Solution:

Augmenting path (if no augmenting path exists write NONE): _____

Value you can augment with on path: _____

**Question 2.2 (5%)** Give a $s-t$ minimum cut in the network $N$.

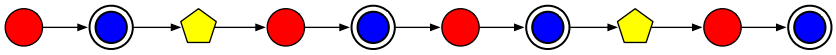
Solution:

Minimum cut (write the partition of the nodes): _____

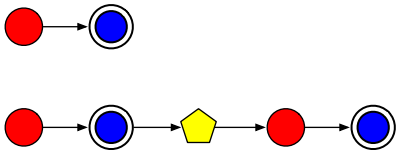
Value of the minimum cut: _____

# Question 3 (15%)

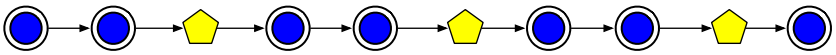
Paul loves pearls and has a lot of strings of pearls with pearls of different colors. Paul especially likes strings of pearls that start and end in the same way. Let $S$ be a string of pearls of length $n$. If the first $i$ pearls ($i < n$) and the last $i$ pearls $S$ are the same he says that the string of pearls has a *pretty end* of length $i$. More formally, if $S = p_1 p_2 \cdots p_n$ then there is a pretty end of length $i < n$ if and only $p_1 \cdots p_i = p_{n-i+1} \cdots p_n$. For example, the string of pearls



has two pretty ends: one of length 2 and one of length 5.



**Question 3.1 (5%)** Compute the length of all pretty ends of the string $S = bbybbybbyb$:



Solution:

$S$ has pretty ends of length(s): _____

**Question 3.2 (10%)** Give an efficient algorithm for computing the length of all pretty ends of a string $S$ of length $n$. Analyze the running time of your algorithm in terms of $n$ and argue that it is correct.

Solution:

Solution 3.2 continued:

# Question 4 (26%)

Consider the following simple 2-player card game. At the beginning of the game, the $n$ cards are dealt face up in a long row. Each card is worth a different number of points. After all the cards are dealt, the players take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, you can decide which of the two cards to take. The winner of the game is the player that has collected the most points when the game ends.

**Question 4.1 (6%)** Having never taken an algorithms class, your friend Donald follows the obvious greedy strategy—when it's his turn, Donald always takes the card with the higher point value.

Prove that you should not also use the greedy strategy when you play against Donald. That is, give an example of a sequence where you can win, but only if you do not follow the same greedy strategy as Donald.

Solution:

**Question 4.2 (5%)** Let $R = [r_1, r_2, \ldots, r_n]$ be the row of cards, that is $R[i]$ is the $i$th card from the left. In this exercise we want to compute the maximum number of points that you can collect playing against Donald.

Let $L[i, j]$ be the maximum number of points that you can get if you play against Donald on the sequence of cards $R[i \ldots j]$ if it is your turn first.

E.g. if $R = [5, 3, 4, 7, 9, 1, 2, 6]$ then $L[3, 5] = 13$ as $R[3 \ldots 5] = [4, 7, 9]$ and you get $9 + 4 = 13$ points if you pick 9 in your first turn, but only $4 + 7 = 11$ if you pick 4 first.

Fill out the table below for the row $R = [7, 4, 3, 5, 6]$.

| $i \setminus j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

**Question 4.3 (5%)** Which of the following recurrences correctly computes $L[i, j]$:

A
$$L[i,j] = \begin{cases} 0 & \text{if } i > j \\ R[i] & \text{if } i = j \\ \max\{R[i], R[j]\} & \text{if } i + 1 = j \\ R[i] + \max\{L[i+2, j], L[i, j-1]\} & \text{if } R[i] > R[j] \text{ and } i < j - 1 \\ R[j] + \max\{L[i+1, j-1], L[i, j-2]\} & \text{if } R[i] < R[j] \text{ and } i < j - 1 \end{cases}$$

B
$$L[i,j] = \begin{cases} 0 & \text{if } i > j \\ R[i] & \text{if } i = j \\ \max\{R[i], R[j]\} & \text{if } i + 1 = j \\ \max\{R[i] + L[i+1, j-1], R[j] + L[i, j-2]\} & \text{if } R[i+1] < R[j-1] < R[j] \text{ and } i < j - 1 \\ \max\{R[i] + L[i+1, j-1], R[j] + L[i+1, j-1]\} & \text{if } R[j-1] < R[i+1] < R[j] \text{ and } i < j - 1 \\ \max\{R[i] + L[i+2, j], R[j] + L[i, j-2]\} & \text{if } R[j-1] < R[i] < R[i+1] \text{ and } i < j - 1 \\ \max\{R[i] + L[i+2, j], R[j] + L[i+1, j-1]\} & \text{if } R[i] < R[j-1] < R[i+1] \text{ and } i < j - 1 \end{cases}$$

C
$$L[i,j] = \begin{cases} 0 & \text{if } i > j \\ R[i] & \text{if } i = j \\ \max\{R[i] + L[i+1, j-1], R[j] + L[i, j-2]\} & \text{if } R[i+1] \le R[j], R[i] \le R[j-1] \text{ and } i < j \\ \max\{R[i] + L[i+1, j-1], R[j] + L[i+1, j-1]\} & \text{if } R[i+1] \le R[j], R[i] > R[j-1] \text{ and } i < j \\ \max\{R[i] + L[i+2, j], R[j] + L[i, j-2]\} & \text{if } R[i+1] > R[j], R[i] \le R[j-1] \text{ and } i < j \\ \max\{R[i] + L[i+2, j], R[j] + L[i+1, j-1]\} & \text{if } R[i+1] > R[j], R[i] > R[j-1] \text{ and } i < j \end{cases}$$

**Question 4.3 (10%)** Write pseudocode for an algorithm *based on dynamic programming and the recurrence you chose in Question 4.2* that given a row $R$ computes the maximum number of points you can get playing against Donald if you take the first turn. Analyze the space usage and running time of your algorithm in terms of $n$.

Solution:

Solution 4.3 continued:

# Question 5 (18%)

The yearly light fest is coming up. Every year all $n$ departments at the university puts different colored light chains on a tree outside the department. Each department $i$ needs $k$ light chains of different colors and they each have a list $L_i$ of the colors they want to use. Each department should get at most one light chain of each color and they should only receive light chains that has a color that is on their list. Your friends are in charge of distributing the light chains to the departments. The university has a limited number of light chains $c_j$ of each color $j$ and their job is to make all the departments happy. Suppose that $k = 2$, there are $n = 4$ departments and $m = 4$ colors with $c_1 = c_2 = c_3 = c_4 = 2$, that is there are two light chains of each color. The departments lists are $L_1 = L_2 = \{1, 2, 3\}$ and $L_3 = L_4 = \{1, 3, 4\}$. Then one possible way to distribute the light chains such that each department gets at least 2 light chains of different colors are:

- department 1 gets light chains of color $c_1$ and $c_2$,
- department 2 gets light chains of color $c_2$ and $c_3$,
- department 3 gets light chains of color $c_3$ and $c_4$,
- department 4 gets light chains of color $c_1$ and $c_4$,

## Question 5.1 (10%)

Give an efficient algorithm that takes as input the number of light chains of each color $c_1, c_2, \ldots, c_m$, the lists $L_i$ for each of the $n$ departments, and the parameter $k$, and decides whether there is a way to distribute the light chains such that each department gets $k$ light chains of different colors. Analyze the time and space usage of your algorithm in terms of $n$, $m$, and $k$. Remember to argue that your algorithm is correct.

Solution:

Solution 5.1 continued:

## Question 5.2 (8%)

You show your friends a solution computed by your algorithm from Question 5.1, and to your big surprise they tell you that this solution is not at all feasible. It turns out that there are three different types of light chains (star shaped, heart shaped and round) and no department should get light chains of only a single type. Each color only comes in one shape (fx. are all the red light chains heart shaped), but every type is made in many different colors. For example, suppose color 1 is star shaped, color 2 and 3 are heart shaped, and color 4 is round. Then the solution from before does not work as department 2 only gets heart shaped light chains. However, we could give both department 1 and 2 light chains of color 1 and 2, and department 3 and 4 light chains of color 3 and 4.

Give an efficient algorithm that decides whether there exists a solution satisfying all the conditions from Question 5.1, plus the new requirement about types. Analyze the time and space usage of your algorithm in terms of $n$, $m$, and $k$. Remember to argue that your algorithm is correct.

Solution:

Solution 5.2 continued: