# Technical University of Denmark

Written exam, December 13, 2018.

Course name: Algorithms and data structures II.

Course number: 02110.

Aids allowed: Written aids are permitted.

Exam duration: 4 hours

Weighting: Question 1: 32% - Question 2: 15% - Question 3: 15% - Question 4: 26% - Question 5: 12%

The weighting is only an approximative weighting.

You can answer the exam in either Danish or English.

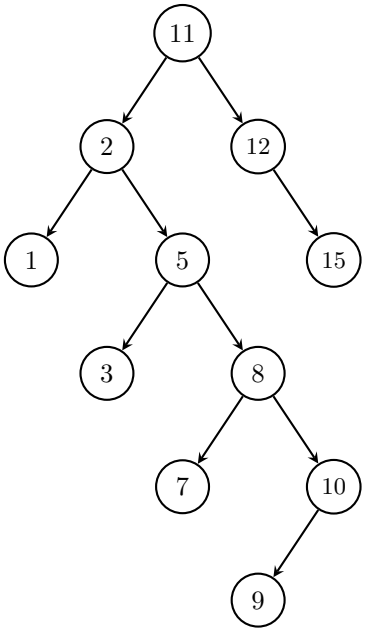
**All questions should be answered by filling out the box below the question.**

**As exam paper just hand in this and the following pages filled out. If you need more space you can use extra paper that you hand in together with the exam paper.**

Study number: _____ Name: _____

# Question 1

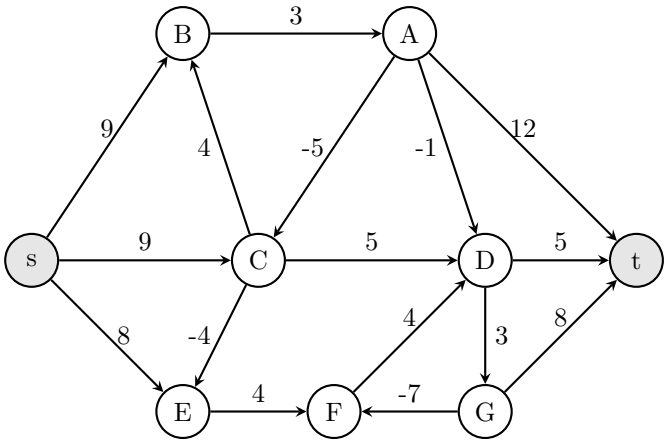
**Question 1.1 (8%)** Show how the splay tree below looks after deleting the element 10.




Solution:

**Question 1.2 (8%)** Find a shortest path from $s$ to $t$ in the graph below. Mark the edges on the shortest path and write the length of the path.

Solution:




Length of shortest path: _____

**Question 1.3 (8%)** Draw the string-matching finite automaton for the string `BBCABBAB`:

Solution:

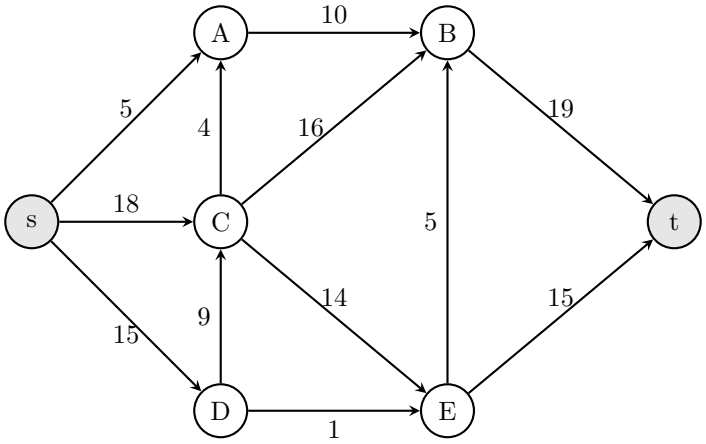
**Question 1.4 (8%)** Compute the prefix function as used in the Knuth-Morris-Pratt algorithm for the string `BBCABBAB`:

Solution:

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $P[i]$ | B | B | C | A | B | B | A | B |
| $\pi[i]$ | | | | | | | | |

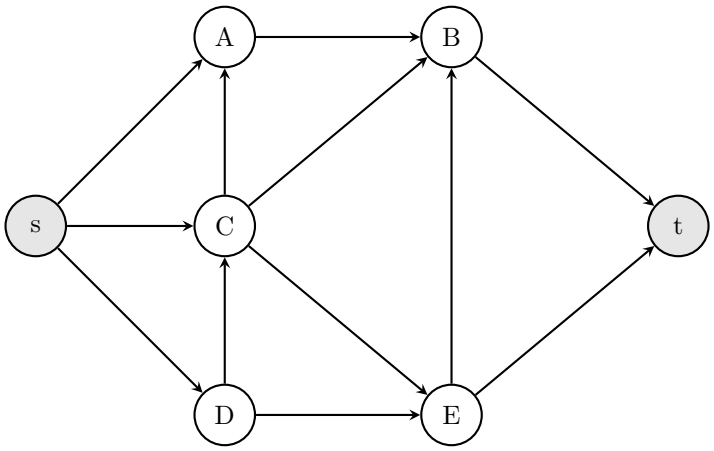
# Question 2

Consider the network $N$ below with capacities on the edges.




**Question 2.1 (8%)** Give a maximum flow from $s$ to $t$ in the network $N$ (write the flow for each edge along the edges on the graph below), give the value of the maximum flow, and give a minimum $s-t$ cut (give the partition of the vertices).

Solution:




value of flow: _____

minimum cut: _____

**Question 2.2 (7%)** Use the *Scaling Max-Flow Algorithm* to compute a maximum flow in the network $N$. For each augmenting path write the nodes on the path and the value you augment the path with in the table below.

| augmenting path | value |
|---|---|
| | |

Solution to this question requires knowledge of suffix trees, which is not part of the curriculum in 2023.



# Question 3

In this question we consider strings $S_1, \ldots, S_k$ over an alphabet of constant size. Each string has length $n$. A *unique substring* of $S_i$ is a substring $a_i$ such that $a_i$ *does not* occur as a substring of $S_j$ for $j \neq i$. For each string $S_i$ we want to find a *shortest* unique substring $a_i$ of $S_i$.

Example: if $S_1 = ABABBACABC$ and $S_2 = ACCBABAABB$ then $a_1 = BC$ is a unique substring of $S_1$ and $a_2 = CB$ is a unique substring of $S_1$. The string $BBA$ is also a unique substring of $S_1$, but it is not a shortest unique substring, since $AC$ is a shorter unique substring.

## Question 3.1 (4%)

For each of the strings $S_1, \ldots, S_4$ below find a *shortest unique substring*.

$$S_1 = AABBAACBAC$$
$$S_2 = CBBCCBACBB$$
$$S_3 = AABBCACBBA$$
$$S_4 = CCCBBCCCCC$$

Solution:

$a_1 =$ _____

$a_2 =$ _____

$a_3 =$ _____

$a_4 =$ _____

## Question 3.2 (11%)

Give an efficient algorithm that given $k$ different strings $S_1, \ldots, S_k$ each of length $n$ from an alphabet of size $O(1)$, finds a shortest unique substring of each string. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

Solution:

# Question 4

You're organizing the First Annual DTU Cake Meeting, to be held on three days in January. There will be lots of cake for everyone and a large number of bakers have applied to bake cakes for the event. You need to hire bakers according to the following constraints.

1. Each candidate baker has given you a list of types of cakes they can bake.
2. Each baker can bake at most four cakes during the entire event.
3. At most one of each type of cake (chocolate cake, strawberry cake, ...) can be produced by all the bakers in total.

## Question 4.1 (10%)

Suppose there are $b$ candidate bakers and $t$ different types of cake. Give an algorithm that computes the maximum number of cakes that can be produced according to the constraints. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

Solution:

Solution 4.1 continued:

## Question 4.2 (8%)

It turns out that there were way too many cakes and not enough variation in style on each day. So you impose the following new constraints (in addition to constraint 1, 2, and 3 from above).

4. There must be produced exactly $k$ cakes each day, and thus $3k$ cakes altogether.
5. Each baker can bake at most two cakes each day (and still at most 4 cakes in total).

Give an efficient algorithm that either assigns a baker and a cake to each of the $3k$ cake slots, or correctly reports that no such assignment is possible. Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

Solution:

Solution 4.2 continued:

## Question 4.3 (8%)

Your friend that is helping you organize the event thinks that the assignment process is too complicated: there are too many different bakers and too many restrictions. He thinks it is more important that you get a possibility to taste all the different types of cake during the event. So he asks you to design an algorithm that given the lists of the types of cakes each baker can bake checks if $q$ bakers are enough to get *at least one* of each type of cake. There are now no restrictions on the number of cakes that a baker can bake or on the total number of cakes (that is, constraints 2, 3, 4 and 5 does not have to be fulfilled).

**Question 4.3.1** Show that the problem of checking if $q$ bakers are enough is in NP.

Solution:

**Question 4.3.2** Prove that the problem of checking if $q$ bakers are enough is NP-complete (remember to argue that your reduction is correct).

Solution:

# Question 5

An exponential multistack consists of an infinite series of stacks $A_0, A_1, A_2, \ldots$, where the $i$th stack $A_i$ can hold up to $2^i$ elements. When pushing an element onto the expontial multistack we first try to push it onto the first stack $A_0$. If this is full, we first pop all elements from $A_0$ and push them onto the next stack $A_1$ to make room. Whenever we try to push an element onto a full stack $A_i$, we first recursively move all elements from $A_i$ to $A_{i+1}$. Moving a single element from one stack to the next takes $O(1)$ time.

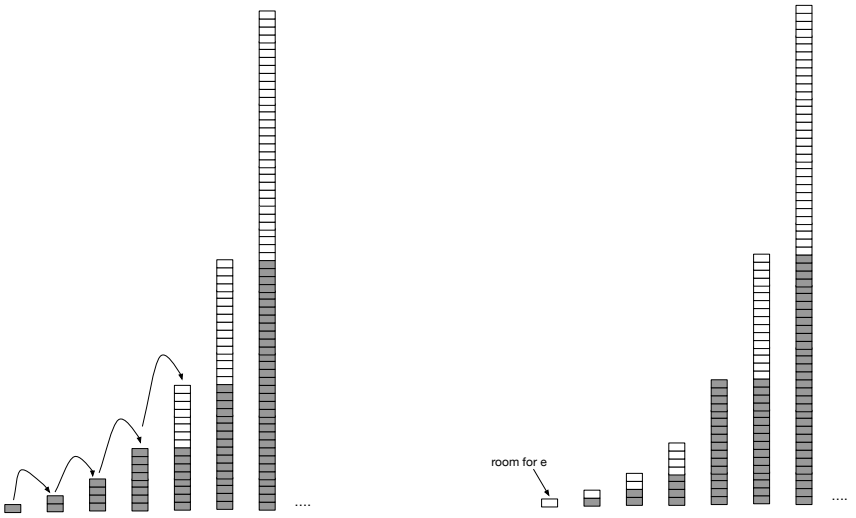



Figure 1: Making room for one new element $e$ in the expontial multistack.

## Question 5.1(2%)

What is the worst case time for pushing an element onto an exponential multistack containing $n$ elements?

Solution:

## Question 5.2 (3%)

What is the maximum number of times a single element has been moved in an exponential multistack containing $n$ elements? Argue why your answer is correct.

Solution:

## Question 5.3 (10%)

Assume we start with an empty exponential multistack. Prove that the amortized cost of a push operation is $O(\log n)$, where $n$ is the maximum number of elements in the exponential multistack.

Solution: