Technical University of Denmark

Written exam, December 14, 2017.

Course name: Algorithms and data structures II.

Course number: 02110.

Aids allowed: All written materials are permitted.

Exam duration: 4 hours

Weighting: Question 1: 30% - Question 2: 15% - Question 3: 20% - Question 4: 18% - Question 5: 17%

The weighting is only an approximative weighting.

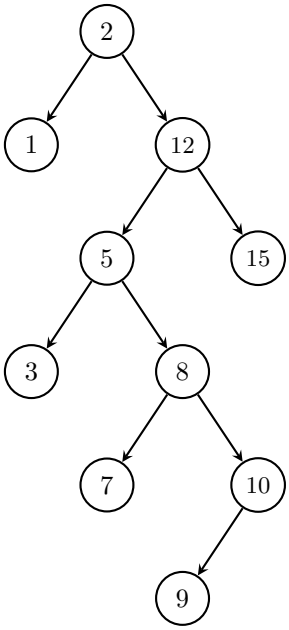
You can answer the exam in either Danish or English.

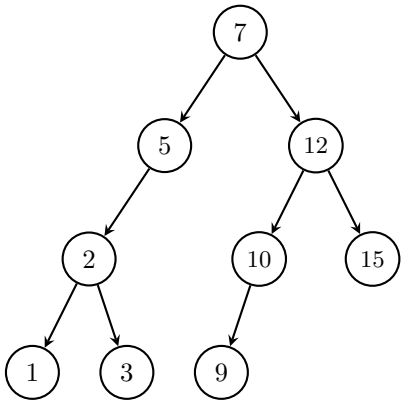
All questions should be answered by filling out the blank space below the question. As exam paper just hand in this and the following pages filled out. If you need more space you can use extra paper that you hand in together with the exam paper.

Study number: _____ Name: _____

Question 1

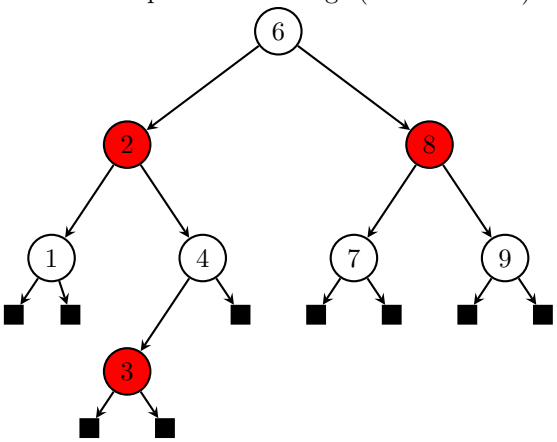
Question 1.1 (8%) Show how the splay tree below looks after deleting the element 8.

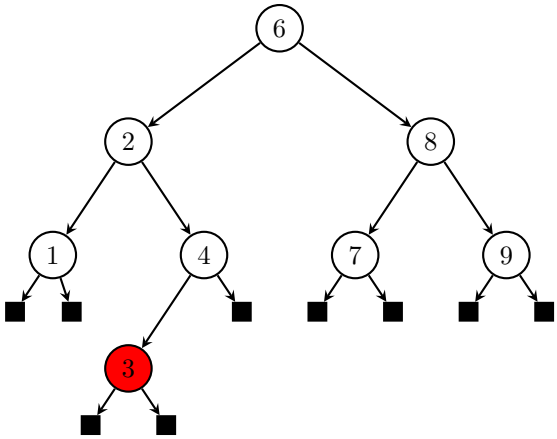


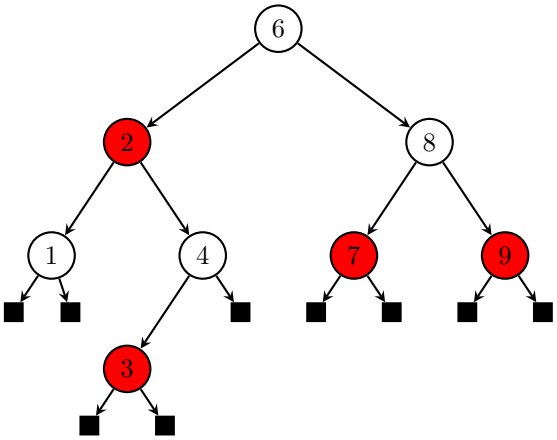


Question 1.2 (7%) Give two different colorings of the tree below that makes it a legal red-black tree. You can either use colors or write B for black and R for red next to the nodes.

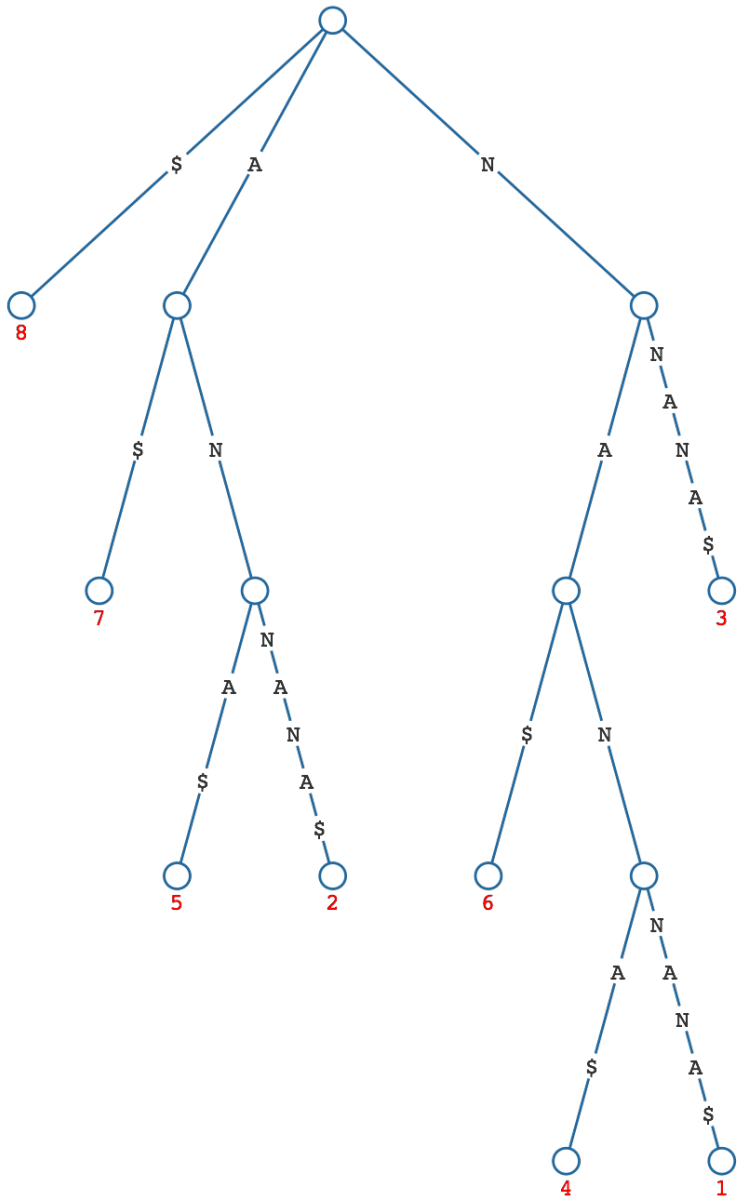
There are 3 possible colorings (white =black):







Question 1.3 (8%) Draw the suffix tree for the string `NANNANA$` (don't replace the labels by indexes into the string, just write the labels on the edges):

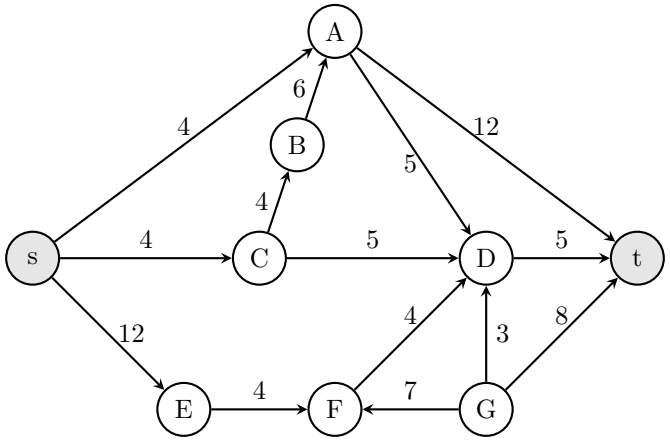


Question 1.4 (7%) Compute the prefix function as used in the Knuth-Morris-Pratt algorithm for the string `NANNANA`.

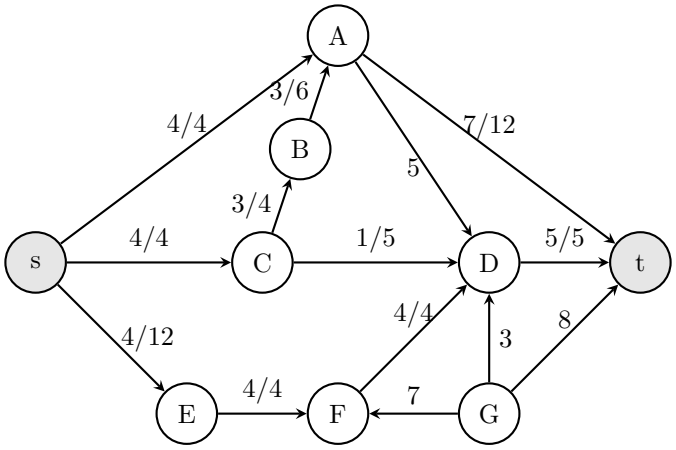
i	1	2	3	4	5	6	7
$P[i]$	`N`	`A`	`N`	`N`	`A`	`N`	`A`
$\pi[i]$	0	0	1	1	2	3	2

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 flow, and give a minimum $s-t$ cut (give the partition of the vertices).



value of flow: 12

minimum cut: $\{sE\}, \{ABCDFGT\}$

Question 2.2 (7%) Use Edmonds-Karp's 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
sAt	4
$sCDt$	4
$sEFDt$	1
$sEFDCBAt$	3

Question 3

The rabbit Billy lives with his family in a rabbit hole.

Question 3.1 The rabbit Billy is very shy and he loves carrots. He knows that there are carrots somewhere on the path going from the rabbit hole into the woods. But he does not know how far away the carrots are. Billy is very shy and is afraid to leave home alone, so he comes up with the following strategy. In the first round he takes 1 step and then goes back to the hole. In the next round he takes 2 steps and then goes back, on the third round he takes 4 steps, etc. That is, in each round he takes twice as many steps as in the previous round. Assume the carrots are n steps away.

Question 3.1.1 (2%) How many rounds does it take before Billy finds the carrots?

$$\lceil \log n \rceil + 1$$

Question 3.1.2 (5%) What is the total number of steps Billy takes before he finds the carrots? (An asymptotic answer is fine). Explain your answer.

Total number of steps is $O(n)$:

$$2(1 + 2 + 4 + \cdots 2^{\lceil \log n \rceil}) = 2 \sum_{i=1}^{\lceil \log n \rceil + 1} 2^{i-1} = 2 \cdot (2^{\lceil \log n \rceil + 1} - 1) = \Theta(n).$$

Question 3.1.3 (5%) What is the amortized number of steps that Billy takes in a round? (An asymptotic answer is fine). Explain your answer.

There are $\lceil \log n \rceil + 1$ and the total number of steps is $O(n)$. By definition the amortized number of steps per round is $O(n/\log n)$.

Question 3.2 Billy is both shy and very forgetful. Last week he hid k carrots in k different bushes. Now he is hungry, but he forgot in which of the b bushes around the rabbit hole he hid the carrots. To find a carrot he now does the following. In each round he goes to a random bush and checks if there is a carrot. If not he runs back to the rabbit hole and tries again. Since he is very forgetful he might go to the same bush again in the next round.

Question 3.2.1 (4%) What is the expected number of bushes that Billy visits before he finds the first carrot? Explain your answer.

The expected number of trials before a succes is $1/p = b/k$.

Or

$$\sum_{i=1}^{\infty} i \cdot \frac{k}{b} \cdot \left(\frac{b-k}{b}\right)^{i-1}$$

Question 3.2 (4%) After eating the first carrot Billy is still hungry, so he keeps looking for two more carrots. He does not remember where he already has found carrots, so he might go to these bushes again. What is the expected number of bushes that Billy visits before he has found 3 carrots? Explain your answer.

$$\frac{b}{k} + \frac{b}{k-1} + \frac{b}{k-2}$$

or

$$\sum_{j=0}^{2} \sum_{i=1}^{\infty} i \cdot \frac{k-j}{b} \cdot \left(\frac{b-k+j}{b}\right)^{i-1}$$

Question 4

The Dean of studies at University of Rabbitland wants to make a new student board for the whole university. The student board should have exactly 1 student representing each of the different classes at campus. A student can represent a class if the student is in that class. A student can be in more than one class, but he or she can only represent one of these classes on the board. For example, Billy the Rabbit is taking both the class "Advanced Carrot Structures" and the class "Introduction to Jumping Algorithms". If Billy is chosen as the "Advanced Carrot Structures" representative, then someone else must represent "Introduction to Jumping Algorithms". There are k classes and n students at University of Rabbitland. The sum of students in all the classes is m (recall that m can be larger than n since a student can be in many classes).

Question 4.1 (10%) Describe an algorithm that, given the lists of students participating in each class, returns a set of k student representatives for the board satisfying all the requirements. If it is not possible to assign members to the board, then the algorithm should return "Not possible".

Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

Model as flow problem. A node for each student and a node for each class. An edge from a student node to a class node with capacity iff the student participates in that class. A source node s and an edge with capacity 1 from s to each of the student nodes. A sink node t and an edge from each class node to t with capacity 1.

If the maximum flow is k return the students that have flow one into them. Otherwise return not possible.

$|V| = n+k+2$ and $|E| = m+n+k$. Maxflow $= k$. The running time of Ford-Fulkerson is $O(k(m+n+k))$.

Question 4.2 (8%) The dean is very happy with your algorithm to put together the board, but he realizes that he forgot something important. The board should consist of exactly the same number of Master students, Bachelor students and Diploma students. That is, the board must consist of $k/3$ Master students, $k/3$ Bachelor students, and $k/3$ Diploma students. Luckily, the number of classes k (and thus the number of members of the board) is a multiplicative of 3.

Describe an algorithm that, given the lists of students participating in each class, returns a set of k student representatives for the board satisfying all the requirements. If it is not possible to assign members to the board, then the algorithm should return "Not possible".

Analyze the asymptotic running time of your algorithm. Remember to argue that your algorithm is correct.

Add 3 nodes to the graph: v_m, v_b and v_d. Remove all edges from s. Add an edge with capacity $k/3$ from s to each of the nodes v_m, v_b and v_d. Add an edge from v_m to a student if the student is a master student, from v_b if the student is a bachelor student, and from v_d if the student is a diploma student.

There are 3 extra nodes and 3 extra edges. The max flow is still k. The running time is as before.

Question 5

A sequence $x_1 < x_2 < \ldots < x_n$ of integers is *inflating* if $x_i - x_{i-1} < x_{i+1} - x_i$ for all $1 < i < n$. For example, the sequence $3, 5, 8, 13$ is an inflating sequence, since $5 - 3 < 8 - 5 < 13 - 8$, whereas $3, 4, 6, 7$ is not, since $6 - 4 > 7 - 6$.

Given a sequence $X = [x_1, \ldots, x_n]$ of integers, where $x_1 < x_2 < \ldots < x_n$, we want to find the longest inflating *subsequence* of X. For example, the sequence $6, 8, 12$ is a longest inflating subsequence of $3, 6, 8, 10, 12$.

For $1 \leq i \leq j \leq n$, let $D(i, j)$ be the length of the longest inflating subsequence of $x_1, \ldots, x_j$, where x_i and x_j are respectively the second last and last elements in the subsequence. Let $D(i, i) = 1$ for all i.

Question 5.1 (4%) Fill out the table below for the sequence $X = [1, 4, 5, 7, 9]$.

$D(i,j)$	1	2	3	4	5
1	1	2	2	2	2
2		1	2	2	3
3			1	3	3
4				1	2
5					1

Question 5.2 (4%) Which of the following recurrences correctly computes $D(i, j)$:

A $$D(i,j) = \begin{cases} 1 & \text{if } i = j \\ 1 + \max\{D(i,\ell) \mid 1 \leq \ell \leq j \text{ and } x_i - x_\ell < x_j - x_i\} & \text{otherwise} \end{cases}$$

B $$D(i,j) = \begin{cases} 1 & \text{if } i = j \\ 1 + \max\{D(\ell,i) \mid 1 \leq \ell < i \text{ and } x_i - x_\ell < x_j - x_i\} & \text{otherwise} \end{cases}$$

C $$D(i,j) = \begin{cases} 1 & \text{if } i = j \\ 1 + \max\{D(\ell,i) \mid 1 \leq \ell \leq i \text{ and } x_i - x_\ell < x_j - x_i\} & \text{otherwise} \end{cases}$$

Question 5.3 (9%) Write pseudocode for an algorithm *based on dynamic programming and the recurrence from Question 5.2* that computes the maximum total score you can achieve. The input to your algorithm is an array $X[1 \ldots n]$ of increasing integers.

Analyze the space usage and running time of your algorithm in terms of n.

```
COMPUTE-INFLATING(X[1...n])
   D := n × n table
   max := 0
   for i=1 to n do
      D[i, i] := 1
   for i = 1 to n do
      for j = i to n do
         local_max: = 0
         for l = 1 to i do
            if (X[i]-X[l] < X[j]-X[i] and D(l,i)>local_max) then
               local_max := D(l,i)
            end if
         end for
         D(i,j) := local_max + 1
         if D(i,j) > max then
            max = D(i,j)
         end if
      end for
   end for
   return max
```

Since we store the results to the subproblems in a table D of size $n \times n$ we use $\Theta(n^2)$ space. There are three nested for loops. All other operations can be done in constant time, and thus the total time of the algorithm is $\Theta(n^3)$.