Technical University of Denmark

Written exam, December 12, 2014.

Course name: Algorithms and data structures.

Course number: 02110.

Aids allow: All written materials are permitted.

Exam duration: 4 hours

Weighting: Question 1: 6% - Question 2: 24% - Question 3: 15% - Question 4: 18% - Question 5: 22% - Question 6: 15%.

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

Question 1

In the following two questions, there might be more than one statement that is true (there might also be 0).

Question 1.1 Let T be a red-black tree with R red nodes and B black nodes (including the NIL leaves). Which of the following statements are true:

A $B \leq R$ **B** $R \leq B$ C $R \geq 1/2B$ **D** $R \leq 1/2B$ E $B \leq 1/2R$

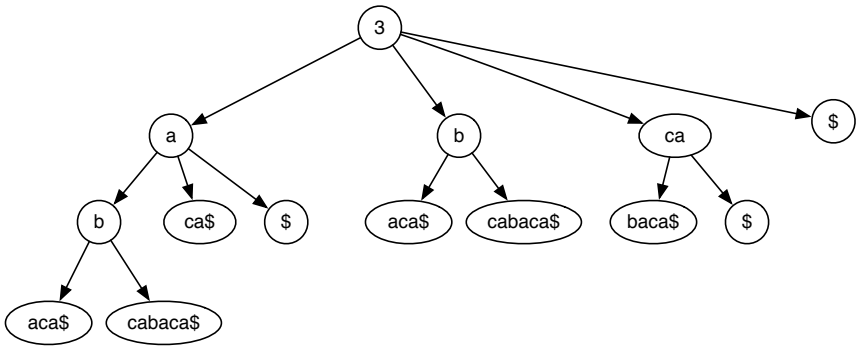
Question 1.2 Let T be a 2-3-4 tree with n nodes and T_R the corresponding red-black tree. Let R be the number of red nodes in T_R and B the number of black nodes in T_R. Which of the following statements are true:

A $n = R + B$ **B** $R \leq n$ **C** $B \leq n$ **D** $B \geq n$ E $R = n$

Here we assume that we count the NIL leaves in both the 2-3-4 tree and in the red-black tree.

Question 2

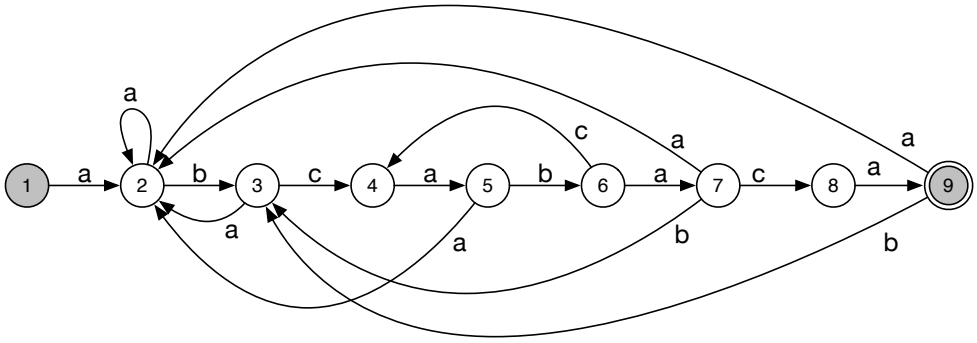
Question 2.1 Draw the suffix tree for the string `abcabaca$` (don't replace the labels by indexes into the string, just write the labels on the edges):



Question 2.2 Compute the prefix function as used in the Knuth-Morris-Pratt algorithm for the string `abcabaca`:

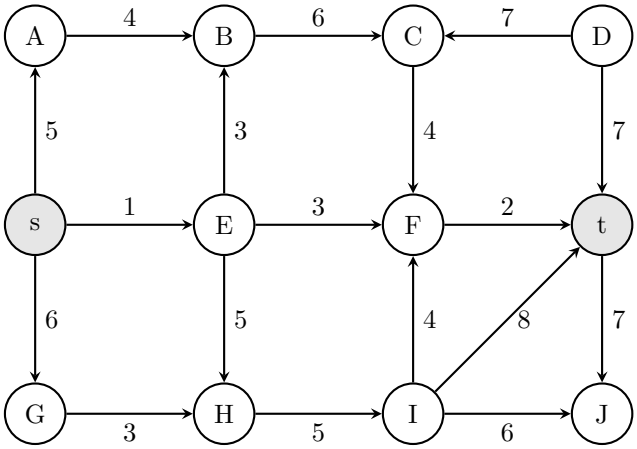
i	a	b	c	a	b	a	c	a
π	0	0	0	1	2	1	0	1

Question 2.3 Draw the string matching finite automata for the string `abcabaca`:

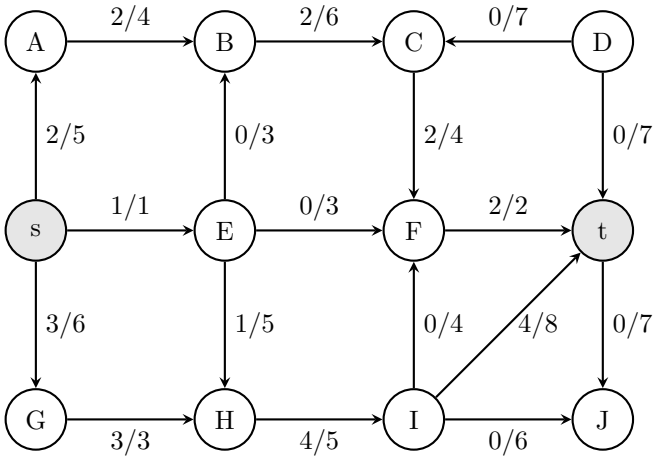


Question 3

Consider the network N below with capacities on the edges.



Question 3.1 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: 6

minimum cut: $\{s, A, B, C, F, G\}, \{D, E, H, I, J, t\}$

Question 3.2 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
$sEFt$	1
$sGHIt$	3
$sABCFt$	1
$sABCFEHIt$	1

Question 4

Given two sequences $S = s_1 \cdots s_m$ and $T = t_1 \cdots t_n$, we want to find a string U that contains S and T as subsequences. We call such a string U for a *supersequence* of S and T. A shortest supersequence is a supersequence of minimal length. If $S = AABAB$ and $T = BBABB$ then $U_1 = AABBABB$, $U_2 = ABBABAB$ and $U_3 = AABABBABB$ are all examples of supersequences of S and T. Both U_1 and U_2 are shortest supersequences.

Let $L(i, j)$ denote the length of the shortest supersequence of $s_1 \cdots s_i$ and $t_1 \cdots t_j$.

Question 4.1 Fill out the table below for $L(i, j)$ when $S = CCB$ and $T = BCBC$.

$L(i,j)$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	2	3	4
2	2	3	3	4	4
3	3	3	4	4	5

Question 4.2 Which of the following recurrences correctly computes $L(i, j)$:

A
$$L(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ and } j = 0 \\ L(i-1,0) & \text{if } j = 0 \text{ and } i > 0 \\ L(0,j-1) & \text{if } i = 0 \text{ and } j > 0 \\ L(i-1,j-1)+1 & \text{if } s_i = t_j, i > 0, \text{ and } j > 0 \\ \min\{L(i-1,j), L(i,j-1)\}+1 & \text{if } s_i \neq t_j, i > 0, \text{ and } j > 0 \end{cases}$$

B
$$L(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ L(i-1,j-1) & \text{if } s_i = t_j, i > 0, \text{ and } j > 0 \\ \min\{L(i-1,j), L(i,j-1)\}+1 & \text{if } s_i \neq t_j, i > 0, \text{ and } j > 0 \end{cases}$$

C
$$L(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ L(i-1,j-1)+1 & \text{if } s_i = t_j, i > 0, \text{ and } j > 0 \\ \min\{L(i-1,j), L(i,j-1)\}+1 & \text{if } s_i \neq t_j, i > 0, \text{ and } j > 0 \end{cases}$$

Question 4.3 Write the pseudocode for an algorithm based on dynamic programming and the recurrence from Question 4.2 that finds the length of the shortest supersequence of two sequences S and T of length n and m. Analyze the space usage and running time of your algorithm in terms of n and m.

```
COMPUTE-LENGTH()
   L := n × m table

   for i = 0 to n do
      for j = 0 to m do
         if i = 0 then
            L[i, j] := j
         else if j = 0 then
            L[i, j] := i
         else if s_i = t_j then
            L[i, j] := L[i-1, j-1]] + 1
         else
            L[i, j] := min(L[i - 1, j], L[i, j - 1]) + 1
         end if
      end for
   end for

   return L[n, m]
```

Since we store the results to the subproblems in a table L of size $n \times m$ we use $\Theta(nm)$ space. The outer loop always make n iterations and the inner loop always makes m iterations per iteration of the outer loop. All other operations can be done in constant time, and thus the total time of the algorithm is $\Theta(nm)$.

Question 5

A zombie breakout has occurred in the country 1D and you have been asked by the prime minister to help find solutions to stop the infection. You are given a map of the country, with coordinates of all cities. There are X infected cities and Y uninfected cities. The map also contains information about all roads between cities. A road goes directly to from one city to another and all roads are directed. There are R roads. A city is reachable from the capital if you can follow one or more roads from the capital to the city. You can assume that the travel time from the capital to any reachable city is no more than a day.

Question 5.1 The zombies only travel on the roads. The capital is still uninfected and the prime minister wants to know how many roads that must be destroyed to keep the capital uninfected. Give an algorithm to compute the minimum number of roads that has to be destroyed to cut off the capital from the infected cities. Analyze the running time of your algorithm in terms of X, Y, and R. Remember to argue that your algorithm is correct.

Solution Model the problem as a min s-t cut problem in a directed graph $G = (V, E)$.There is a vertex v_a for each city a and there is an edge from vertex v_a to vertex v_b if and only if there is a road from city a to city b. There is also a vertex s that is connected to all vertices representing infected cities. That is, there is an edge from s to v_a if a is infected. Thus $V = \{v_a | a \text{ is a city}\} \cup \{s\}$ and $E = \{(v_a, v_b) | \text{ there is a road from city } a \text{ to city } b\} \cup \{(s, v_a) | a \text{ is infected}\}$. The capacities on all edges from s are set to ∞ and all other capacities are set to 1. Finding the minimum number of roads that needs to be destroyed to cut off the capital corresponds to finding the minimum capacity s-t cut in the graph, where t is the capital.

Correctness. If there is a set of Z roads cutting of the capital form the infected cities, then these edges is a s-t cut in the graph of size Z. If there is a s-t cut in the graph of size $Z < \infty$ then destroying these roads will cut off the capital from all the infected cities: since all edges from s have capacity ∞ none of these edges can be in the cut, and therefore all infected cities will be on the same side of the cut as s. Since the size of min cut is less than ∞ (removing all edges except from the ones from s will give an s-t cut and this cut has size R), finding a min cut in the graph solves the problem.

Analysis We have $|V| = X + Y$ and $|E| = R + X$. If we use the Edmonds-Karp algorithm to find a min-cut we get a running time of $O(|V||E|^2) = O((X + Y)(R + X)^2)$. If we use the Ford-Fulkerson algorithm we get a running time of $O(|f^*||E|) = O(|E|R) = O((R + X)R)$, since a min cut in the graph has size at most R.[1]

[1]Note: Since the analysis of the Ford-Fulkerson algorithm also applies to the Edmonds-Karp algorithm the running time of Edmonds-Karp is $O(\min(\{(X + Y)(R + X)^2, (R + X)R\}) = O((R + X)R)$.

Question 5.2 Researchers in the center in the capital have found a vaccine. To get the vaccine to the people in the uninfected cities we need to send 10 doctors to each uninfected city. The doctors can only travel on the roads. They can go through infected cities but at most 50 doctors can go through each infected city per day. Give an algorithm to decide whether we can get the vaccine out to all uninfected cities in one day. Analyze the running time of your algorithm in terms of X, Y, and R. Remember to argue that your algorithm is correct.

Solution Model the problem as a max flow problem in a directed graph $G = (V, E)$. There is two vertices v_a^{in} and v_a^{out} for each city a. There is an edge from v_a^{in} to v_a^{out}. The capacity of this edge is 50 if a is infected and ∞ otherwise. There is an edge from vertex v_a^{out} to vertex v_b^{in} if and only if there is a road from city a to city b. These edges have capacity ∞. There is also a vertex t that is connected with all vertices representing uninfected cities. That is, there is an edge from v_a^{out} to t if a is uninfected. The capacities of these edges are 10.

Thus $V = \{v_a^{\text{in}} | a \text{ is a city}\} \cup \{v_a^{\text{out}} | a \text{ is a city}\} \cup \{t\}$ and
$E = \{(v_a^{\text{in}}, v_a^{\text{out}}\} \cup \{(v_a^{\text{out}}, v_b^{\text{in}})| \text{ there is a road from city } a \text{ to city } b\} \cup \{(v_a^{\text{out}}, t) | a \text{ is uninfected}\}$.

We use a max flow algorithm to find the s-t maximum flow in the graph where s is the capital. If the maximum flow is $10Y$ then we can get the vaccine to all uninfected city in one day.

Correctness. If there is a way to get 10 doctors to each uninfected city then there is a maximum flow of $10Y$: If x doctors travel on a road then assign a flow of x to the corresponding edges, and if x doctors go through a city a assign a flow of x to the edge $(v_a^{\text{in}}, v_a^{\text{out}})$. Also assign a flow of 10 to each edge going from an uninfected city to t. For all infected cities we can assume the doctors entering the city leaves again and therefore we have flow conservation. For each uninfected city a we can assume that all except 10 doctors entering the city leave again. Since we assigned a flow of 10 on the edge from v_a^{out} to t we have flow conservation in these nodes as well. All capacity constraints are satisfied (at most 50 doctors go through an infected city and exactly 10 stays in each uninfected city).

If there is a maximum flow of size $10Y$ then there is a way to get 10 the doctors to each uninfected city: We know there always is an integral flow when the capacities are integral. We can send the doctors according to the flow. Since the capacity on the edges $(v_a^{\text{in}}, v_a^{\text{out}})$ is 50 for all infected cities at most 50 doctors go through each infected city. Since the capacity is 10 on all edges into t, we get exactly 10 doctors staying in each uninfected city if the flow is $10Y$.

Analysis. $|V| = 2(X + Y)$ and $|E| = R + Y$. If we use the Ford-Fulkerson algorithm we get a running time of $O(|f^*||E|) = O(10Y|E|) = O((R + Y)Y)$, since the maximum flow in the graph has size at most $10Y$.

If we use the Edmonds-Karp algorithm to find a max-flow we get a running time of $O(|V||E|^2) = O((X + Y)(R + Y)^2)$. [2]

[2]Note: Since the analysis of the Ford-Fulkerson algorithm also applies to the Edmonds-Karp algorithm the running time of Edmonds-Karp is $O(\min(\{(X + Y)(R + Y)^2, (R + Y)Y\}) = O((R + Y)Y)$.

Question 6

To find a vaccine against the zombie virus, researchers have been looking at genes from infected and uninfected people. Assume we are given k strings $S_1, S_2, \ldots, S_k$ that we know are infected with virus, and one string U that is virus free. All the strings are over an alphabet of size $O(1)$. The total length of the strings is $n = |U| + \sum_{i=1}^{k} |S_i|$. An *indicator string* V is a string that is a substring of *all* the strings $S_1, S_2, \ldots, S_k$, but *not* a substring of U.

Given the strings $S_1, S_2, \ldots, S_k$, U, and a positive integer ℓ we want to determine whether or not there exists an indicator string V of length *at least* ℓ.

Fx. is $V = BA$ an indicator string for the set of strings below.

	1	2	3	4	5	6	7	8	9	10	11	12
$S_1 =$	A	B	C	A	B	A	C	C	A			
$S_2 =$	C	C	A	B	C	A	A	B	A			
$S_3 =$	B	A	B	C	A	A	B	C	C	A		
$S_4 =$	C	C	A	B	C	A	C	A	B	A		
$S_5 =$	A	B	C	A	A	B	A	C	C	A		
U =	A	B	B	C	A	C	C	A	B	C	C	A

Question 6.1 Find an indicator string V of at length at least 3 for the strings above and give the position of its first occurrence in each of the virus infected strings.

$V =$ *ABCA*

Positions: $S_1 : 1, S_2 : 3, S_3 : 2, S_4 : 3, S_5 : 1$.

Question 6.2 Give an algorithm that given the strings $S_1, \ldots, S_k$ and U of total length n, and an integer ℓ, finds an indicator string of length at least ℓ or reports that no such indicator string exists.

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

Solution Construct the suffix tree for $S_1\$_1S_2\$_2\cdots\$_{k-1}S_k\$_kT\$_{k+1}$, where $\$_i$, for $i = 1, \ldots, k+1$, are characters not in the alphabet. Mark for each node v and each i, $1 \leq i \leq k$, wether there is a leaf in the subtree of v there is a suffix of S_i, and if there is a suffix of T in v's subtree. This is marked bottom up in a traversal of the tree, where the markings in a node is the merge of the markings of its children. Also compute the string depth of each node during the traversal (the string depth of v is the length of the string obtained by concatenating the characters on the path from the root to v).

If a node v has a marking for all S_i but not for T and the string depth of v is at least ℓ then the string obtained by concatenating the characters on the path from the root to v is an indicator string of length at least ℓ. If we find such a string during the traversal of the tree we report the string. Otherwise we report "no indicator string of length at least ℓ exists".

Analysis The suffix tree can be constructed in $O(n)$ time, since the size of the alphabet is constant. The marking takes time $O(k)$ for each child of a node. Thus the marking takes $O(nk)$ time in total. Computing the string depth during a traversal takes constant time for each node and thus the running time of the algorithm is $O(nk)$.

Correctness Let I be an indicator string of length at least ℓ. Find the corresponding position in the suffix tree by following the path corresponding to from the root. If this is on an edge go to the vertex below. The string corresponding to this vertex exists in all S_i and thus all S_i must be represented in the subtree of v. I is not in T and therefore no leaf that is a suffix of T is in the subtree of v. Similarly, if there is a no of string depth at least ℓ with leaves in the subtree corresponding to all S_i but not T then the substring corresponding to v occurs in all S_i but not in T and is therefore an indicator string.