Technical University of Denmark

Written exam, December 9, 2021.

Course name: Algorithms and data structures II.

Course number: 02110.

Aids allowed: Written aids are permitted.

Exam duration: 4 hours

Weighting: Question 1: 28% - Question 2: 8% - Question 3: 14% - Question 4: 22% - Question 5: 17% - Question 6: 11 %.

The weighting is only an approximative weighting.

You can answer the exam in either Danish or English.

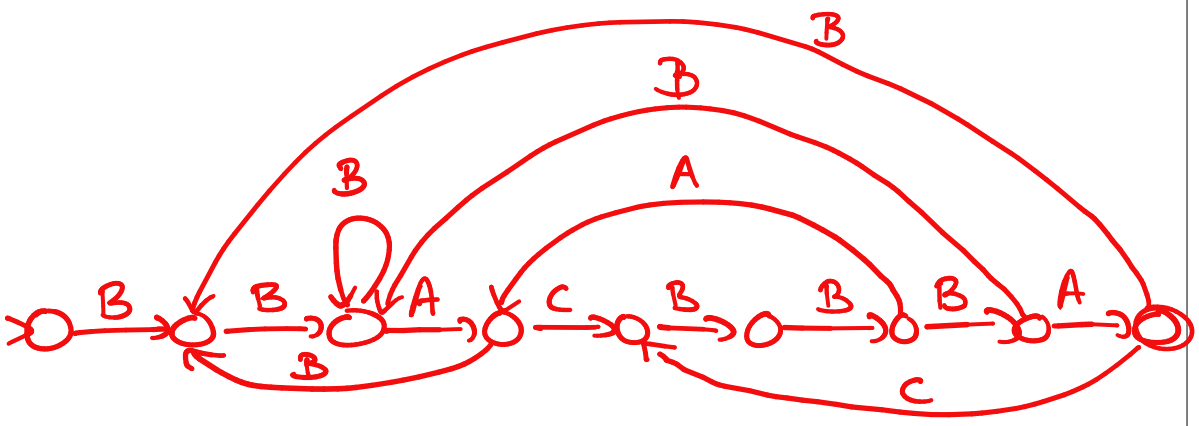
All questions should be answered by filling out the box below the question.

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

Question 1 (28%)

Question 1.1 (7%) Draw the string matching automaton for the string `BBACBBBA`:

Solution:

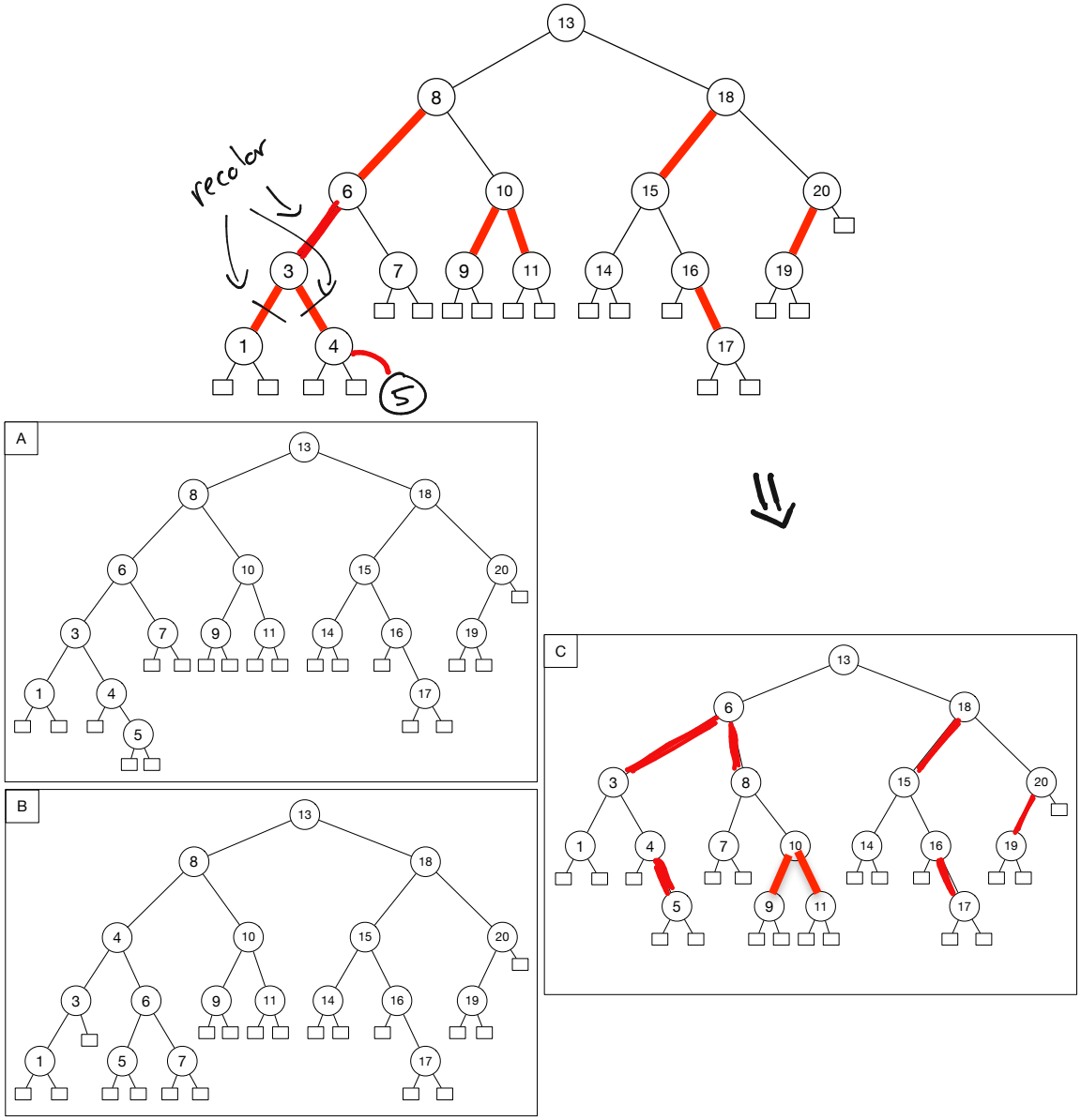


Question 1.2 (7%) Compute the prefix function as used in the Knuth-Morris-Pratt algorithm for the string `BBACBBBA`:

Solution:

i	1	2	3	4	5	6	7	8
$P[i]$	B	B	A	C	B	B	B	A
$\pi[i]$	0	1	0	0	1	2	2	3

Question 1.3 (7%) How does the red-black tree below look after inserting 5? Choose the correct tree and color the edges the color they have after the insertion. (You can either color the edge, write R or B next to each edge, or clearly mark the red and black edges in some other way.)



Question 1.4 (7%) Compute the Fenwick tree F for the array A below:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$A[i]$	–	2	4	1	1	6	7	5	3	2	3	2	4	7	8	2	1

Solution:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$F(i)$	-	2	6	1	8	6	13	5	29	2	5	2	11	7	15	2	58

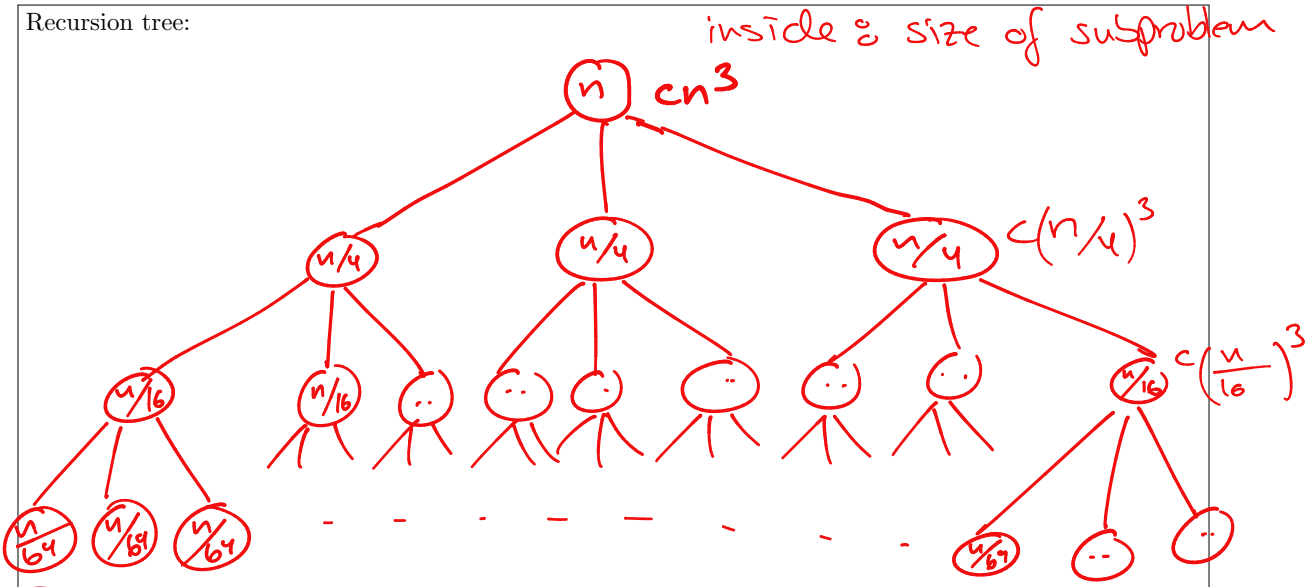
Question 2 (8%)

Let $T(n)$ denote the running time of an algorithm and let $T(n)$ be defined by the following recurrence.

$$T(n) = \begin{cases} 3 \cdot T(n/4) + cn^3 & \text{if } n > 4 \\ c & \text{otherwise} \end{cases}$$

Question 2.1 Draw the recursion tree for the recurrence. You should draw at least the first 4 levels. Write the size of the subproblem represented by each node, the time used in a single subproblem on each level, and the total time used on each level drawn in the tree.

Recursion tree:



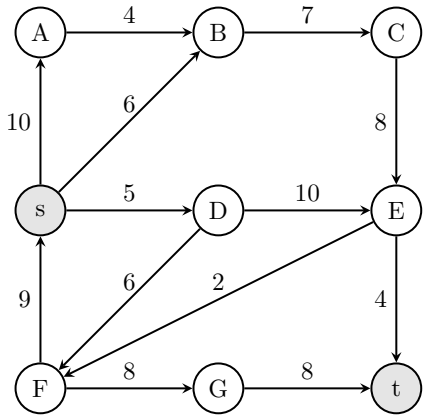
Time
level 0: single node cn^3, total cn^3
level 1: single node $c(\frac{n}{4})^3$, total $3 \cdot c(n/4)^3 = (\frac{3}{64})cn^3$
level 2: single node $c(\frac{n}{16})^3$, total $9\,c(n/16)^3 = (\frac{3}{64})^2 cn^3$
level 3: single node $c(\frac{n}{64})^3$, total $3^3 cn(\frac{n}{64})^3 = (\frac{3}{64})^3 cn^3$

Question 2.2 The total time used on level j can be expressed as:

A $(\frac{9}{64})^j cn^3$ B $(\frac{3}{64})^j cn^3$ C $(\frac{3}{12})^j cn^3$ D $(\frac{3}{4})^j cn^3$ E $\left(\frac{3cn^3}{4}\right)^j$

Question 3 (14%)

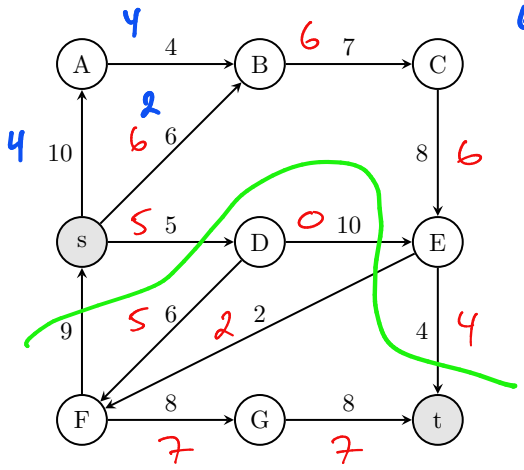
Consider the network N below with capacities on the edges.



Question 3.1 (7%) 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).

several correct. Eg. blue also correct,

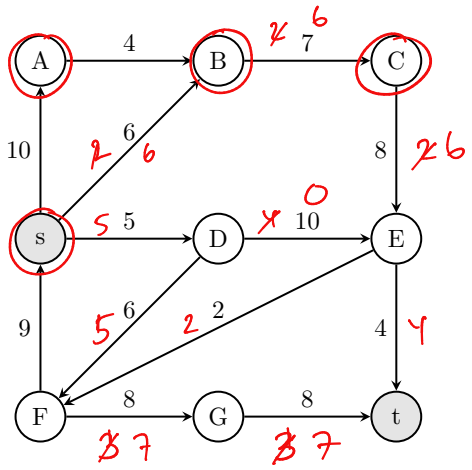
Solution:



value of flow: 11

minimum cut: {s, A, B, C, E} {D, F, G, t}

Question 3.2 (7%) Use the *Edmonds-Karp Max-Flow Algorithm* to compute a maximum flow in the network N (the network is shown again below). For each augmenting path write the nodes on the path and the value you augment the path with in the table below.

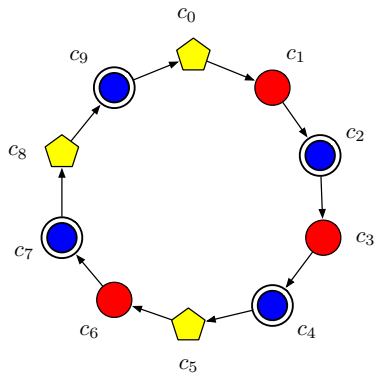


Solution:

augmenting path	value
sDEt	4
sDFGt	1
sBCEFGt	2
sBCEDFGt	4

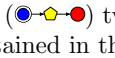
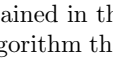
Question 4 (22%)

Some of your friends have a company "Pretty Pearls" where they sell bracelets made of pearls. Each pearl has a color from a set of colors C. A bracelet B can be described as a circular sequence of colors $c_0c_1 \cdots c_{n-1}$, where pearl c_{n-1} is considered to precede c_0 in the sequence. For example the bracelet B



is of size 10, and the set of colors is $C = \{b, r, y\}$. The bracelet can be described as the sequence $B = yrbrbyrbyb$.

They have made a webshop, where customers can type in a pattern in the form of a sequence of colors from C, and then ask if a specific bracelet contains the pattern. Formally, the pattern $P = p_0p_1 \cdots p_{m-1}$ is contained in the bracelet $B = c_0c_1 \cdots c_{n-1}$ if and only if $m \leq n$ and there exists a $j \in \{0, \ldots, n-1\}$ such that $p_i = c_{j+i \mod n}$ for all $i \in \{0, \ldots, m-1\}$.

For example, the bracelet B above contains the pattern byr () twice: both beginning at position 4 and at position 9, whereas the pattern rry () is not contained in the bracelet.

Your friends ask you if you can help them to construct an algorithm that can solve the problem for them.

Question 4.1 (11%) Give an algorithm that given a bracelet B of length n and a pattern P of length m computes wether the bracelet contains the pattern. Analyse the space and running time of your algorithm. Remember to argue that your algorithm is correct.

Solution:

Use KMP with pattern P and string B·B.

Time O(m+n).

Correctness: Show alg finds pattern ⇕ pattern is contained.

Solution 4.1 continued:

⇓ ∘ Find starting and ending in same B ⇒ contained ✓

If start in first B and end in second then .....

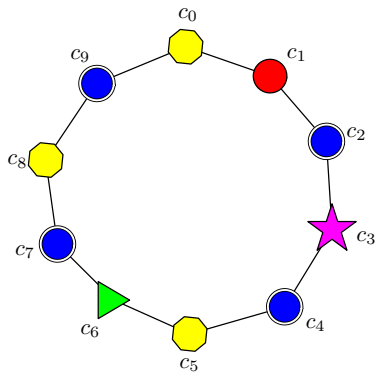
⇑ ∘ ...

Question 4.2 (11%) Sometimes they modify the bracelets by exchanging a pearl in the design. And at the same time they would like to answer questions of the form: How many blue and red pearls are there in the bracelet from pearl i to pearl j (both inclusive).

Help your friends by giving a data structure that given a bracelet $B = c_0c_1 \cdots c_{n-1}$ of length n supports the following operations:

- ChangeColor(i, c): Change the color of pearl i to c.
- CountBlueRed(i,j): Returns the total number of blue and red pearls in the bracelet from pearl i to pearl j (including both i and j).

Note that i can be larger than j. In that case, you should return the total number of blue and red colors in the sequence $c_ic_{i+1} \cdots c_{n-1}c_0 \cdots c_j$. For example in the bracelet below (B = *yrbpbygbyb*), the query CountBlueRed(3, 5) returns 1 and the query CountBlueRed(6, 1) returns 3.



Remember to analyse the space usage and query time of your data structure.

Solution:

Construct a partial sums data structure e.g. a Fenwick tree over an array A, where $A[i] = \begin{cases} 1 & \text{if } c_{i-1} \in \{b, r\} \\ 0 & \text{o.w.} \end{cases}$

Solution 4.2 continued:

```
ChangeColor(i,c): if c ∈ {b,r}:
                      If sum(i+1) - sum(i) = 0:
                          add(i+1, 1)
                      If c not in {b,r}:
                          If sum(i+1)-sum(i) = 1:
                              add(i+1,-1)

CountBlueRed(i,j):
    if i ≤ j :
        return sum(j+1) - sum(i)

    else (i > j):
        return sum(n) - sum(i)
               + sum(j-1)
```

Question 5 (17%)

Your friends needs to buy new pearls for the company. They have a budget of B and they found a place where they can buy large bags of pearls. There are k different bags $\{1, 2, \ldots, k\}$. For each bag i they know the number n_i of pearls in the bag and the price p_i of the bag. They want to maximize the number of pearls they can get without exceeding the budget B.

Your friends suggest the following strategy: Keep picking the bag with the minimum average price for a pearl from the remaining bags, until you cannot pick the next one without exceeding the budget. That is, compute $c_i = p_i/n_i$ for all bags and pick the bag with the minimum c_i until you cannot pick the next one without exceeding B.

Question 5.1 (4%) Give an example that shows that the algorithm suggested by your friends does not find the optimal solution.

Solution:

B = 14

	1	2	3
p_i	10	7	7
n_i	10	6	6

OPT : bag 2 and 3 , # pearls = 12

Greedy : 1 , # pearls = 10

Question 5.2 (3%) You decide to help your friends find an algorithm that can return the maximum number of pearls they can buy. Let $L[i, b]$ be the maximum number of pearls you can get with budget b using a subset of the bags $\{1, \ldots, i\}$. Which of the following recurrences correctly computes $L[i, b]$:

A
$$L[i,b] = \begin{cases} 0 & \text{if } p_i > b \\ n_i & \text{if } p_i = b \\ \max\{L[i-1,b], L[i-1,b-p_i]+n_i\} & \text{otherwise} \end{cases}$$

~~B~~
$$L[i,b] = \begin{cases} 0 & \text{if } i < 1 \\ L[i-1,b] & \text{if } p_i > b \\ \max\{L[i-1,b], L[i-1,b-p_i]+n_i\} & \text{otherwise} \end{cases}$$

C
$$L[i,b] = \begin{cases} 0 & \text{if } i < 1 \\ L[i-1,b] & \text{if } p_i > b \\ \max\{L[i-1,b], L[i-1,b-n_i]+p_i\} & \text{otherwise} \end{cases}$$

Question 5.3 (10%) Write pseudocode for an algorithm *based on dynamic programming and the recurrence you chose in Question 5.2* that given a list $P = [p_1, \ldots, p_k]$ of the prices and a list $S = [n_1, \ldots, n_k]$ of the sizes of the k bags computes the maximum number of pearls you can get. Analyze the space usage and running time of your algorithm in terms of k and B.

Solution:

```
L : 2 dimensional array (k+1)x(Bx1)
Initialize L to 0.

for i = 1 to k :
    for b = 1 to B:
        if p[i] > b :
            L[i][b] = L[i-1, b]

        else
            L[i][b] = max{L[i-1,b],
                          L[i-1, b-pi]+ni}

Return L[k][B]
```

Space = O(kB)
Time = O(kB)

Solution 5.3 continued:

Question 6 (11%)

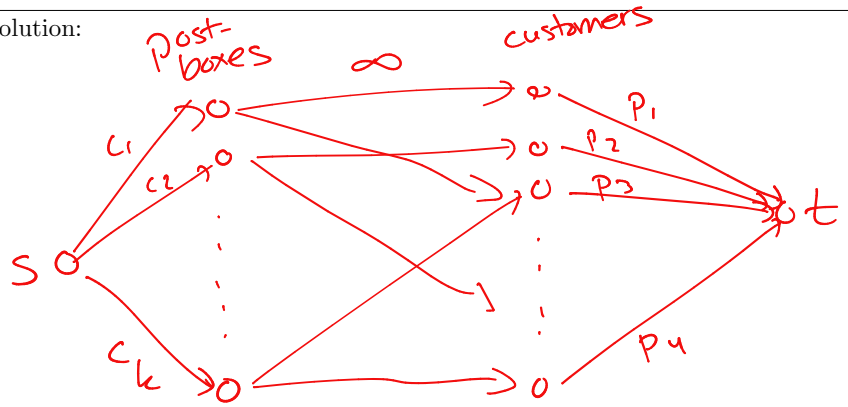
In the city of Algotown the company "Efficient Parcel Service" is busy bringing out packages before Christmas. They have installed k postboxes $\{1, \ldots, k\}$ around the city, where their n customers can pick up their packages. Each customer j has a list L_j of approved postboxes, where they are willing to pick up packages. Let p_j denote the number of packages that customer j should get. Each postbox i has capacity c_i, which is the maximum number of packages it can contain.

All packages start at the central depot, and at night they are delivered to the postboxes. The company asks you to find an algorithm that can help them with the distribution for a given night. They want to deliver as many packages as possible, while ensuring that all customers only need to go to a postbox on their list (a customer j can go to many postboxes to pickup packages as long as they are all on their list L_j).

Give an efficient algorithm that finds the maximum number of packages that can be delivered.

The input to the algorithm is the free capacity of the postboxes $c_1, c_2, \ldots, c_k$, and for each customer j the number of packages p_j and the list L_j of approved postboxes. Analyze the time and space usage of your algorithm in terms of k, n, and the total length of the lists $L = \sum_{j=1}^{n} |L_j|$. Remember to argue that your algorithm is correct.

Solution:



Node u_i for each postbox $i \in \{1, \ldots, k\}$
Node v_j for each customer $j \in \{1 \ldots n\}$.
Edge from s to u_i of capacity c_i.
Edge from v_j to t of capacity p_j.
Edge from u_i to $v_j \Leftrightarrow i \in L_j$.

Solution 6.1 continued:

value of max flow f^* = maximum number of packages that can be delivered.

Correctness:
Show

flow of value X $\Leftrightarrow$ X packages can be delivered.

#edges = $k + n + L$, #nodes = $k + n + 2$

f^* = max flow $\leq \min\{\sum p_i, \sum c_j\}$.

Scaling algorithm: $O((k+n+L)^2 \log C)$
where $C = \max_i \{c_i\}$.

Edmonds-Karp:
$O((k+n)(k+n+L)^2)$

Ford-Fulkerson: $O((k+n+L) f^*)$
(pseudo-polynomial)

Choose Scaling or EK.