Exam - June 2021

Algorithms and Data Structures

Instructions. This exam consists of **five questions** and you have time until 13:00 to submit your solution in digital exam. You can answer the questions directly on this paper, or use additional sheets of paper which have to be hand-in as a **single pdf file**. You are encouraged to mark the multiple choice answers as well as the labelling of graphs directly in this exam sheet.

- Before starting solving the questions, read carefully the exam guidelines at `https://www.moodle.aau.dk/mod/page/view.php?id=1173709`.
- Read carefully the text of each exercise. Pay particular attentions to the terms in bold.
- **CLRS** refers to the textbook T.H. Cormen, Ch. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition).
- You are allowed to refer to results in the textbook as well as exercise or self-study solutions posted in Moodle to support some arguments used in your answers.
- Make an effort to use a readable handwriting and to present your solutions neatly.

Question 1. 15 Pts

Identifying asymptotic notation. (Note: lg means logarithm in base 2)

(1.1) [5 Pts] Mark **ALL** the correct answers. $n^2\sqrt{n} + n^5 \lg n^5 + n \lg 2^n$ is

☐ **a)** $\Theta(n^5 \lg n)$ ☐ **b)** $\Theta(n)$ ☐ **c)** $\Theta(n^{2.5})$ ☐ **d)** $\Theta(n^5 \lg n^5)$ ☐ **e)** $\Theta(n^5)$

(1.2) [5 Pts] Mark **ALL** the correct answers. $n^2\sqrt{n} + n \log_3 2^n$ is

☐ **a)** $\Theta(n^5)$ ☐ **b)** $\Omega(n)$ ☐ **c)** $\Theta(n^{2.5})$ ☐ **d)** $\Omega(\sqrt{n})$ ☐ **e)** $O(n^5)$

(1.3) [5 Pts] Mark **ALL** the correct answers. $100 \cdot n^2 + n^2 \lg 8^n + \frac{n \lg n}{0.5} + \lg n^n$ is:

☐ **a)** $\Omega(n \lg n)$ ☐ **b)** $O(n^3)$ ☐ **c)** $O(n^2)$ ☐ **d)** $\Omega(n^2 \lg n)$ ☐ **e)** $O(n^2 \lg n)$

Question 2. 20 Pts

Consider the following recurrences

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 0 \\ n \cdot T(n-1) & \text{if } n > 0 \end{cases} \qquad Q(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 8 \cdot Q(n/2) + 2^n & \text{if } n > 1 \end{cases}$$

Answer the questions below concerning these two recurrences. For each question, play close attention to whether it concerns $Q(n)$ or $T(n)$.

(2.1) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** $Q(n)$ can be solved using Case 1 of the Master Theorem
- ☐ **b)** $Q(n)$ can be solved using Case 2 of the Master Theorem
- ☐ **c)** $Q(n)$ can be solved using Case 3 of the Master Theorem
- ☐ **d)** $T(n)$ can be solved using the Master Theorem

(2.2) [5 Pts] Mark **ALL** correct answers.

- ☐ **a)** $Q(n) = \Theta(2^n \lg n)$
- ☐ **b)** $Q(n) = O(n^3)$
- ☐ **c)** $Q(n) = \Theta(2^n)$
- ☐ **d)** $Q(n) = \Omega(n^{100})$

(2.3) [10 Pts] Prove that $T(n) = \Omega(2^n)$ using the substitution method.

Question 3. 25 Pts

Understanding of known algorithms.

(3.1) [5 Pts] Mark **ALL** the correct statements. Consider a modification to Quicksort, called MaxQuicksort, such that each time Partition is called, the maximum element of the sub-array to partition is found and used as a pivot.

- ☐ **a)** MaxQuicksort best-case running time is $\Theta(n^2)$
- ☐ **b)** If A is already sorted, then the running time of MaxQuicksort(A) is $\Theta(n \lg n)$
- ☐ **c)** MaxQuicksort(A) sorts the array A in **non-increasing** order
- ☐ **d)** MaxQuicksort worst-case running time is $O(n^3)$
- ☐ **e)** MaxQuicksort works in-place

(3.2) [4 Pts] Mark **ALL** the correct statements. Consider the array $A = [4, 3, 6, 2, 1, 5]$ and assume that $A.\mathit{heap\text{-}size} = A.\mathit{length}$.

- ☐ **a)** The binary tree interpretation of A satisfies the binary search tree property
- ☐ **b)** The result of Max-Heapify$(A, 1)$ is $[6, 3, 5, 2, 1, 4]$
- ☐ **c)** The result of Max-Heapify$(A, 1)$ is $[6, 3, 4, 2, 1, 5]$
- ☐ **d)** A satisfies the max-heap property

(3.3) [6 Pts] Consider the hash table $H = 97$, Nil, Nil, 14, Nil, Nil, Nil, 29, Nil, $75, 32$. Insert the keys $55, 8, 10$ in H using *open addressing* with the auxiliary function $h'(k) = k$.

Mark the hash table resulting by the insertion of these keys using linear probing.

- ☐ **a)** 97, Nil, Nil, 14, Nil, 10, 55, 29, 8, 75, 32
- ☐ **b)** 97, 55, 10, 14, Nil, Nil, Nil, 29, 8, 75, 32
- ☐ **c)** 97, 10, Nil, 14, Nil, Nil, 55, 29, 8, 75, 32
- ☐ **d)** none of the above

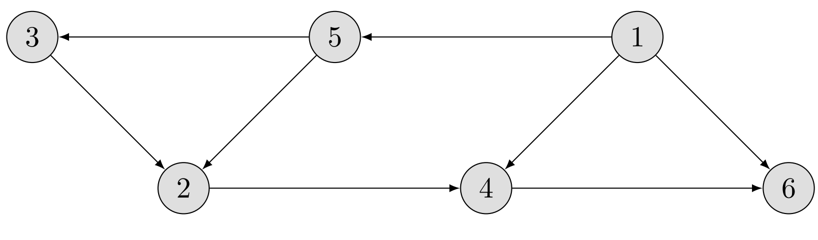
Mark the hash table resulting by the insertion of these keys using quadratic probing with $c_1 = 2$ and $c_2 = 4$.

- ☐ **a)** 97, Nil, Nil, 14, Nil, 10, 55, 29, 8, 75, 32
- ☐ **b)** 97, 55, 10, 14, Nil, Nil, Nil, 29, 8, 75, 32
- ☐ **c)** 97, 10, Nil, 14, Nil, Nil, 55, 29, 8, 75, 32
- ☐ **d)** none of the above

Mark the hash table resulting by the insertion of these keys using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \mod (m - 1))$.

- ☐ **a)** 97, Nil, Nil, 14, Nil, 10, 55, 29, 8, 75, 32
- ☐ **b)** 97, 55, 10, 14, Nil, Nil, Nil, 29, 8, 75, 32
- ☐ **c)** 97, 10, Nil, 14, Nil, Nil, 55, 29, 8, 75, 32
- ☐ **d)** none of the above

(3.4) [10 Pts] Consider the directed graph G depicted below.



(a) Write the intervals for the discovery time and finishing time of each vertex in the graph obtained by performing a depth-first search visit of G (see CLRS sec.22.3).
Remark: If more than one vertex can be chosen, choose the one with smallest label.

(b) Mark the corresponding "parenthesization" of the vertices in the sense of CLRS Theorem 22.7 resulting from the DFS visit performed before

☐ **a)** (1 (5 (2 (4 (6 6) 4) 2) (3 3) 5) 1) ☐ **b)** (1 (4 (6 6) 4) (5 (2 2) (3 3) 5) 1)

☐ **c)** (1 (4 4) (5 (2 2) (3 3) 5) (6 6) 1) ☐ **d)** none of the above

(c) Assign to each edge a label T (tree edge), B (back edge), F (forward edge), or C (cross edge) corresponding to the classification of edges induced by the DFS visit performed before.

(d) If G admits a topological sorting, then show the result of TOPOLOGICAL-SORT(G) (see CLRS sec.22.4). If it doesn't admit a topological sorting, briefly argue why.

Question 4. 20 Pts

Asymptotic runtime analysis.

Prof. Algo has been asked to analyse user interactions in a social network. Prof. Algo started by modelling the social network as a graph $G = (V, E)$ where each vertex represents a user of the network and there exists an edge $(u, v) \in E$ if and only if user v liked some content posted by user u. Additionally, G is equipped with a weight function $w\colon E \to \mathbb{N}$ such that, for $(u, v) \in E$, $w(u, v)$ is the number of likes given by user v to user u.

(a) [10 Pts] Interested in discovering groups of users having intense mutual interactions, Prof. Algo defines the concept of *k-ranked group* as a strongly connected component $C \subseteq V$ in the subgraph $G^k = (V, E^k)$ where $E^k = \{(u, v) \in E \mid w(u, v) \geq k\}$. Then, he provides the following algorithm to find all *k-ranked groups* of G.

```
RankedGroups(G, w, k)
Let G^k be an empty graph.
G^k.V = G.V
for each u ∈ G.V
    let G^k.Adj[u] be an empty list
    for each v ∈ G.Adj[u]
        if w(u, v) ≥ k
            List-Insert(G^k.Adj[u], v)
Strongly-Connected-Components(G^k)
```

Perform an asymptotic analysis of the worst-case running time of RankedGroups(G, w, k). Motivate your answer.

(b) [10 Pts] Prof. Algo defines the *influence* of an user $u \in V$ as *influence*$(u) = \sum_{v \in V} \delta(u, v)$, where $\delta(u, v)$ is the shortest path weight from u to v in G. Then, he provides the following algorithm which prints the vertices of the graph in non-decreasing order of influence.

```
PrintByInfluence(G, w)
Let T be an empty binary search tree
for each s ∈ V
    Dijkstra(G, w, s)
    s.key = 0
    for each v ∈ V − {s}
        s.key = s.key + v.d
    Tree-Insert(T, s)
Inorder-Tree-Walk(T.root)
```

Perform an asymptotic analysis of the worst-case running time of PrintByInfluence(G, w). Motivate your answer.

Question 5. 20 Pts

Solving computational problems.

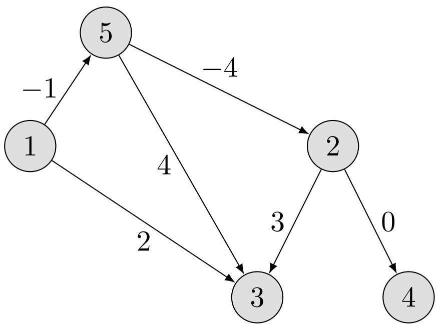
Given a directed **acyclic** graph $G = (V, E)$ with $V = \{1, \dots, n\}$ and weight function $w \colon E \to \mathbb{R}$, we consider the problem of finding a **longest** (maximally-weghted) simple path from i to j for all pairs of vertices $i, j \in V$.

(a) [10 Pts] Describe a **bottom-up** dynamic programming procedure ALLPAIRSLONGESTPATH(G, w) that returns an $n \times n$ matrix $L = (l_{ij})_{i,j \in V}$ where l_{ij} is the weight of a longest simple path from i to j.

(b) [10 Pts] Describe a procedure PRINTLONGESTPATH(G, w, i, j) that prints a longest simple path from i to j.

Remarks:

- The description of the algorithmic procedures must be given **both** by providing the pseudocode and by explaining in detail how it works.
- Specify in your solution whether the weighted graph (G, w) is assumed to be represented using adjacency matrix or adjacency lists.
- Try to execute your algorithm on the following example. You may catch some errors you did not foresee while designing your algorithm.



L	1	2	3	4	5
1	0	-5	3	-5	-1
2	$-\infty$	0	3	0	$-\infty$
3	$-\infty$	$-\infty$	0	$-\infty$	$-\infty$
4	$-\infty$	$-\infty$	$-\infty$	0	$-\infty$
5	$-\infty$	-4	4	-4	0

For instance, the longest path from vertex 1 to vertex 3 is the sequence $1, 5, 3$ having weight $-1+4 = 3$, while the longest path from 2 to 1 is the empty sequence with weight $-\infty$.