Technical University of Denmark

Written examination, May 21, 2019.

Course name: Algorithms and Data Structures 1

Course number: 02105

Aids: Written aids. Calculators are **not** permitted.

Duration: 4 hours.

Weights: Exercise 1 - 24 %, Exercise 2 - 20 %, Exercise 3 - 16 %, Exercise 4 - 15 %, Exercise 5 - 25 %. The weights are approximate. The grade is based on an overall assessment.

All exercises should be answered by filling out the areas below the description. As your solution to the exam, just hand in this page and the following pages. If you really need more space, you may use extra pieces of paper and hand these in along with your solution.

Asymptotic bounds should be as tight as possible. Unless otherwise specified, the base of all logarithms is 2 and $\log^k n$ means $(\log n)^k$.

Name: ____

Student ID: ____________________________________

1 Complexity

1.1 (6 %) For each statement below, mark whether or not it is correct.

	Yes	No
$2+2n^2+2\log^2 n = \Theta(n^2)$	□	□
$\frac{2\cdot\sqrt{n}}{10^7} + 2\cdot\log(n^7) = O(\sqrt{n})$	□	□
$\log(2^n)+\log(2n) = \Omega(n)$	□	□
$80^2\cdot n + \frac{n^2}{10^7} = \Theta(n)$	□	□
$\log(\log n) + 5n^{1/3} + 6n^{1/5} = \Omega(n^{1/4})$	□	□

1.2 (6 %) Arrange the following functions in increasing order according to asymptotic growth, that is, if $g(n)$ immediately follows $f(n)$ then $f(n) = O(g(n))$.

$$2\cdot 3^n \qquad 18^4 \qquad \sqrt{n}\log n \qquad 54n^3 \qquad n^2 \qquad \log(n^3)$$

Solution: ______________________________

1.3 (12 %) State the running time for each of the following algorithms. Write your solution in O-notation as a function of n.

```
ALG1($n$)
$c = 0$
$j = n - 4$
while $j \leq n$ do
    $j = j + 1$
    for $i = 1$ to $n$ do
        $c = j + i$
    end for
end while
```

Solution: ______________

```
ALG2($n$)
if $n \leq 1$ then
    return 1
else
    return $1 + $ALG2$(1) + $ALG2$(1)$
end if
```

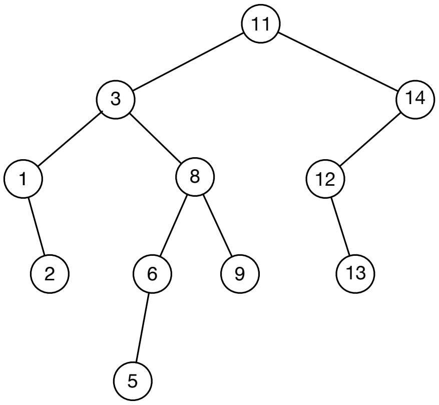
Solution: ______________

```
ALG3($n$)
$c = 0$
for $i = 1$ to $\lceil \log n \rceil$ do
    for $j = i$ to $n$ do
        $c = i + j$
    end for
end for
```

Solution: ______________

2 Data Structures and Algorithms

Consider the following binary search tree T.



2.1 (4 %) Write the ordering of the vertices from a preorder, postorder, and inorder traversal of T.

PREORDER : ____

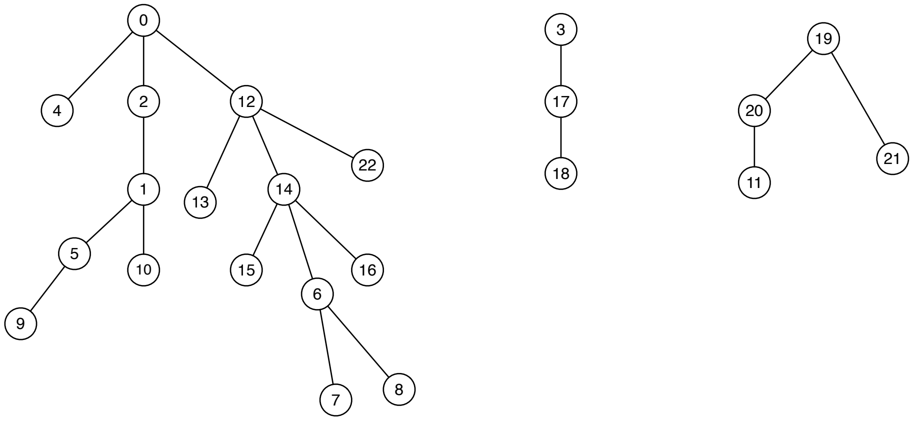
POSTORDER : ____

INORDER : ____

2.2 (4 %) Show T after deleting the vertex with key 3.

Solution:

Consider the following forest of trees representing a family of sets in a union find data structure constructed using the quick union algorithm.



2.3 (4 %) State the results of the following FIND(·) operations.

FIND(2) : ____________

FIND(18) : ____________

FIND(0) : ____________

FIND(8) : ____________

FIND(11) : ____________

FIND(14) : ____________

2.4 (4 %) Assume that we now use path compression on the forest above. Show the forest of trees after a FIND(8) operation.

Solution:

2.5 (4 %) We want to support the operation Top(k) on each of the following data structures. Let n be the total number of elements in the data structure. Given a parameter $k \leq n$, Top(k) returns the k largest elements in the data structure. For instance, if the data structure consists of the elements $\{6, 32, 18, 7, 2\}$, Top(2) should return 32 and 18 since these are the 2 largest elements among the 5 elements. State for each of the following data structures, the running time of an efficient algorithm for the Top(k) operation in O-notation as a function of k and/or n.

Single-linked list: ____________________

Unsorted array: ____________________

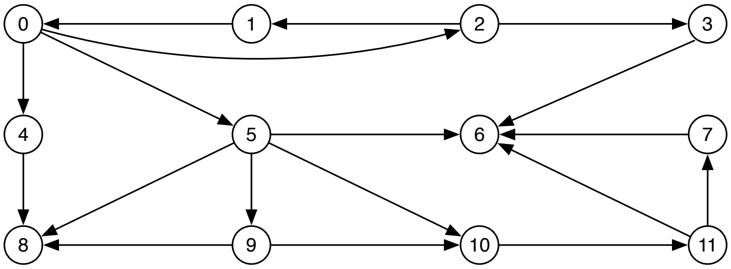
Sorted array: ____________________

Max-heap: ____________________

Binary search tree: ____________________

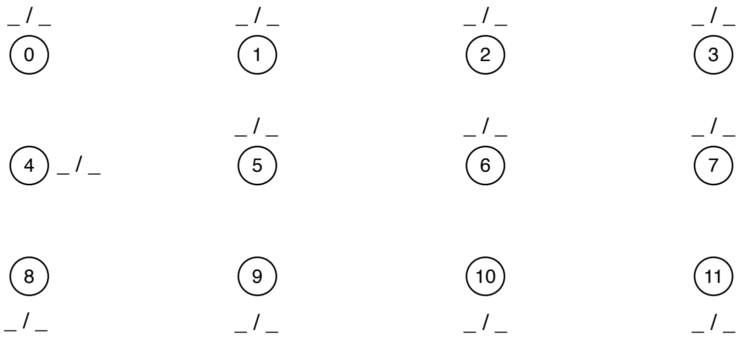
3 Graphs

Consider the following graph *G*.



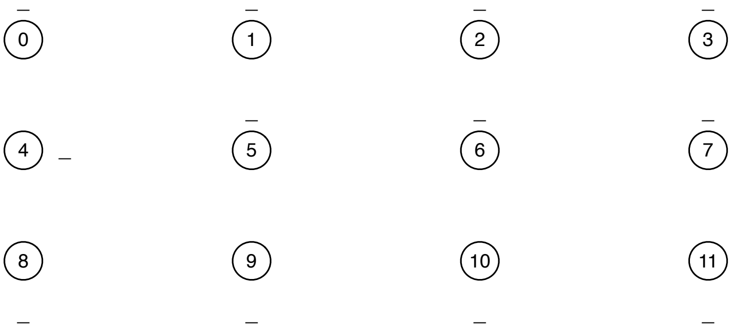
3.1 (4 %) Show the DFS tree for *G* when starting in vertex 0 and write the discovery and finish times for each vertex. Assume that the adjacency lists are sorted in increasing order. Write the discovery and finish times for each vertex in the area marked by "_ / _" next to each vertex.

Solution:

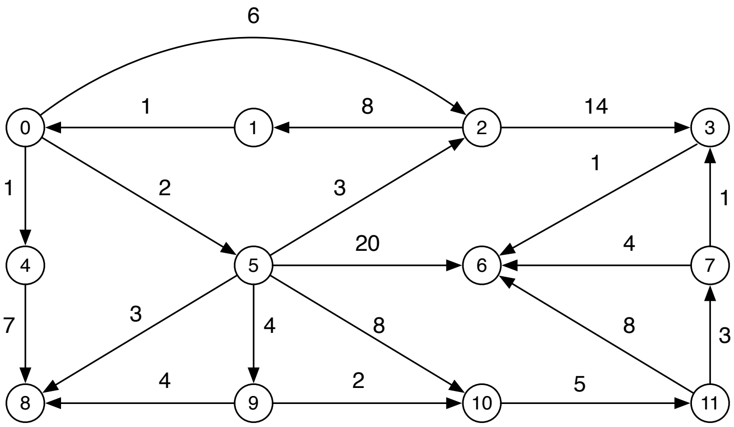


3.2 (4 %) Show the BFS tree for *G* when starting in vertex 0. Assume that the adjacency lists are sorted in increasing order. Write the BFS layer for each vertex in the area marked by "_" next to each vertex.

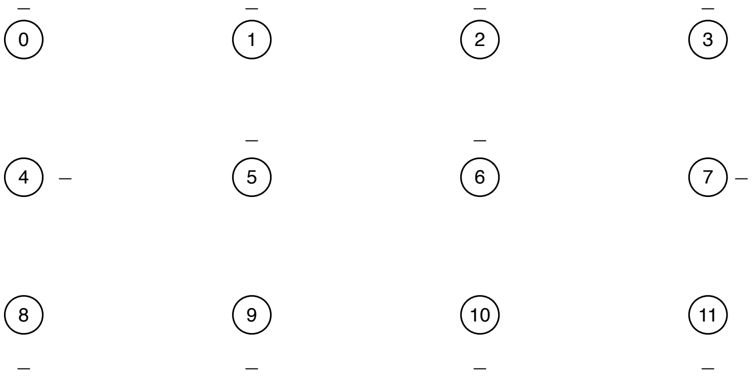
Solution:



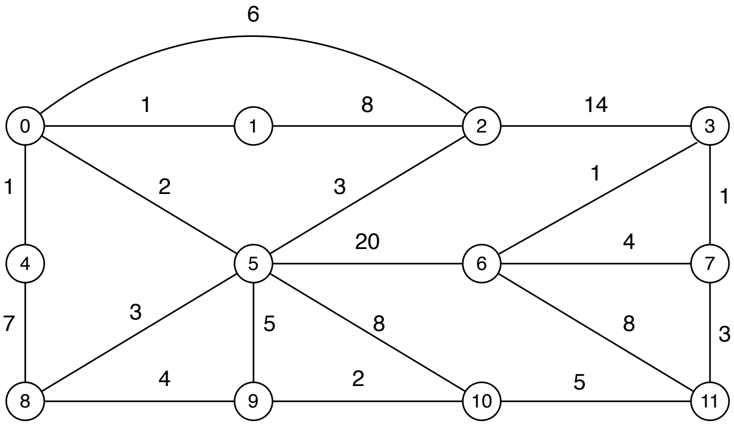
3.3 (4 %) Consider the following graph. Show a shortest path tree for the graph starting at vertex 0. Write the length of the shortest path for each vertex in the area marked by "_" next to each vertex.



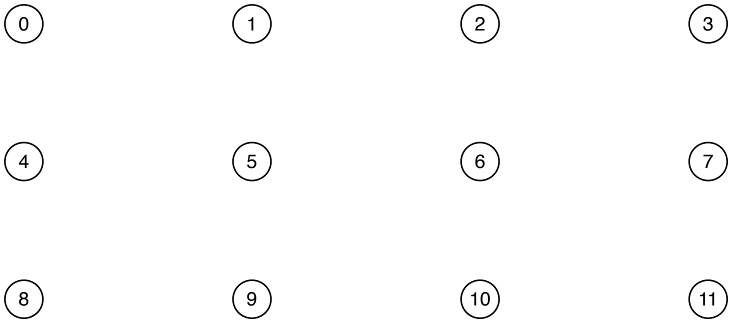
Solution:



3.4 (4 %) Consider the following graph. Show a minimum spanning tree for the graph. State the total weight of the tree.



Solution:



Total weight: ________________

4 Trees

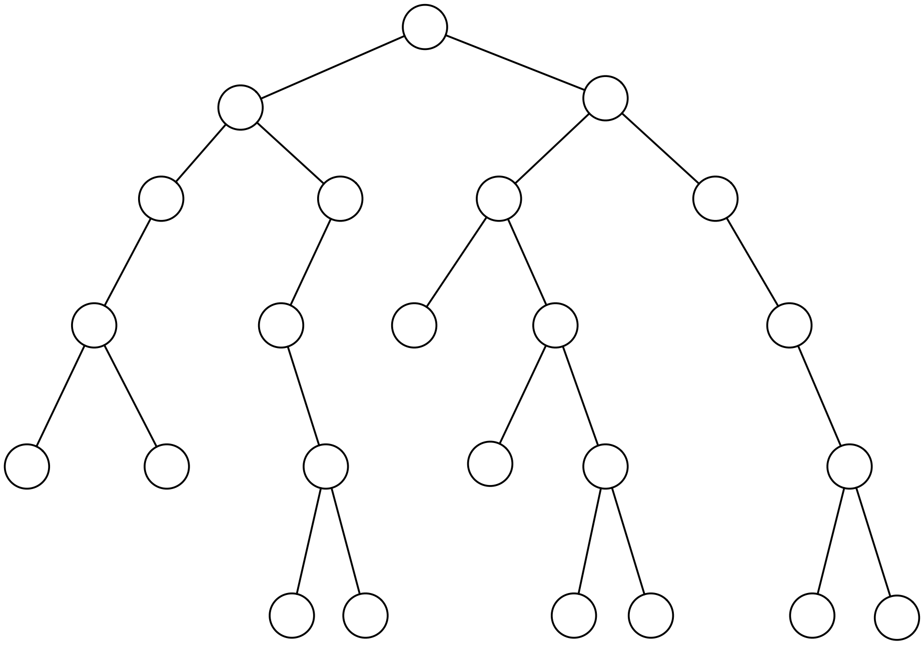
This exercise is about rooted binary trees. Each node x has fields $x.parent$, $x.left$ and $x.right$, denoting the parent, left child, and right child of x. For the root $root$, $root.parent = null$. Throughout the exercise, we let n denote the size of the input tree.

4.1 (1 %) Recall that a node has a *sibling* if its parent has two children. A node x is a *twig* if it satisfies the following two properties:

- x has depth at least 2.
- x has exactly two children that are both leaves.
- x and the parent of x have no siblings.

Consider the tree below. Mark all twigs in the tree directly on the vertices in the tree.

Solution:



4.2 (5 %) Give an algorithm, SIBLING(x), that given a non-root vertex x returns true if and only if x has a sibling. Write your algorithm in pseudocode. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

4.3 (5 %) Give an algorithm, TWIG(x), that given a vertex x of depth at least 3 returns true if and only if x is a twig. In addition to the SIBLING algorithm from the previous exercise, you can assume that you have a constant time algorithm LEAF(X) that given a vertex x, returns true if and only if x is a leaf. Write your algorithm in pseudocode. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

4.4 (4 %) Give a recursive algorithm, TwigCount(x), that given the root, returns the number of twigs in the tree. Write your algorithm in pseudocode. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

5 Ruling the Galaxy

A *galaxy* is a set of P *planets* and a set of H *hyperspace links*. Each hyperspace link is a pair (x, y) of distinct planets x and y and we say that planet x is *hyperspace linked* to y. Furthermore, a *hyperspace sequence* from planet z_1 to planet z_j is a sequence of planets $z_1, \ldots, z_j$ such that (z_i, z_{i+1}) is a hyperspace link for $i = 1, \ldots, j-1$. We assume that any pair of planets has at least one hyperspace sequence between them. For example, the sets $\{a, b, c, d, e\}$ and $\{(a, b), (b, c), (c, d), (d, e), (c, e), (c, a)\}$ is a galaxy with $P = 5$ planets and $H = 6$ hyperspace links. The sequence a, b, c, e is a hyperspace sequence from a to e.

5.1 (2 %) Describe how to model a galaxy as a graph

Solution:

5.2 (1 %) Draw the graph corresponding to the galaxy in the example.

Solution:

5.3 (6 %) Given a non-negative integer t, a *t-hub* is a planet that is hyperspace linked to at least t other planets. Give an algorithm, that given a galaxy and a non-negative integer t, computes the number of t-hubs in the galaxy. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

5.4 (6 %) We are now interested in optimizing the cost of maintaining the empire while ruling the galaxy. We associate to each hyperspace link $\ell = (x, y)$ a *maintenance cost*, denoted $m(\ell)$, that indicates the cost of keeping the hyperspace link open for the fleet. An *hyperspace link set* is a subset of hyperspace links such that any two planets have a hyperspace sequence between them and an *optimal hyperspace link set* is a hyperspace link set such that the total cost of its hyperspace links is minimal.

Give an algorithm, that given an galaxy, computes an optimal hyperspace link set. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

5.5 (6 %) The empire has a new Death Star weapon system that is extremely powerful, but also slow to move from planet to planet. We want to compute how fast the Death Star can reach any planet in the galaxy from the empires home planet. We associate to each hyperspace link $\ell = (x, y)$, a *travel time*, denoted $t(x, y)$, that indicates the time needed for the Death Star to travel between x and y. The travel time for a hyperspace sequence is the sum of travel time of the hyperspace links in the hyperspace sequence. The *home planet*, h, is a special planet where the Death Star is docked between operations. The *response time to planet x* is the minimum travel time for the Death Star from h to x. The *galactic response time* is the maximum response time for any planet in the galaxy.

Give an algorithm, that given a galaxy and a home planet h, computes the galactic response time. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution:

5.6 (4 %) To improve the response times the empires now invests in k Death Stars, located at distinct home planets $h_1, \ldots, h_k$. In this setting, the response time to a planet x is now the shortest response time for any of the Death Stars, i.e., the minimum travel time for any of the k Death Stars to x. The galactic response time is the maximum response time for any planet in the galaxy. Give an algorithm, that given a galaxy and k home planets $h_1, \ldots, h_k$, computes the galactic response time. Analyse the running time of your algorithm in the relevant parameters of the problem.

Solution: