Technical University of Denmark

Written examination, May 17, 2017.

Course name: Algorithms and Data Structures

Course number: 02326

Aids: Written aids. It is **not** permitted to bring a calculator.

Duration: 4 hours.

Weights: Exercise 1 - 20 %, Exercise 2 - 25 %, Exercise 3 - 20 %, Exercise 4 - 13 %, Exercise 5 - 22 %. The weights are approximate. The grade is given based on an overall assessment.

All exercise should be answered by filling out the areas below the description. As your solution to the exam, just hand in the page and the following pages. If you 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 (5 %) For each statement below, mark whether or not it is correct:

	Yes	No
$\frac{1}{4}n^3 + n^2 \log n + 17 \cdot n^2 = \Theta(n^3)$	☐	☐
$10^{17} + (n^3)^2 + \log^2 n = O(n^5)$	☐	☐
$42 \log n + \sqrt{n} + n^{1/3} = \Omega(\sqrt{n})$	☐	☐
$2^{\log n} + \frac{n^{1,5}}{30^{30}} + \log^{10} n = \Theta(n)$	☐	☐
$(\log n)(\log^2 n + \log n) = O(\log^2 n)$	☐	☐

1.2 (5 %) Arrange the following functions in increasing order according to asymptotic growth. That is, if $g(n)$ immediately follows $f(n)$ in your list, it must hold that $f(n) = O(g(n))$.

$$8\sqrt{n} \qquad \log(2^n) \qquad 7n^3 \qquad 1^n \qquad 3\log^2 n \qquad n^{4/2}$$

Solution:

1.3 (10 %) State the running time for each of the following algorithms. Write your answer in O-notation as a function of n.

```
ALG1(n)
c = 0
for i = 1 to ⌊n/3⌋ do
  c = c + 1
end for
for j = 1 to ⌊n/5⌋ do
  c = c + 1
end for
```

Solution:

```
ALG2(n)
if n ≤ 1 then
  return 1
else
  return 1+ALG2(n − 2)
end if
```

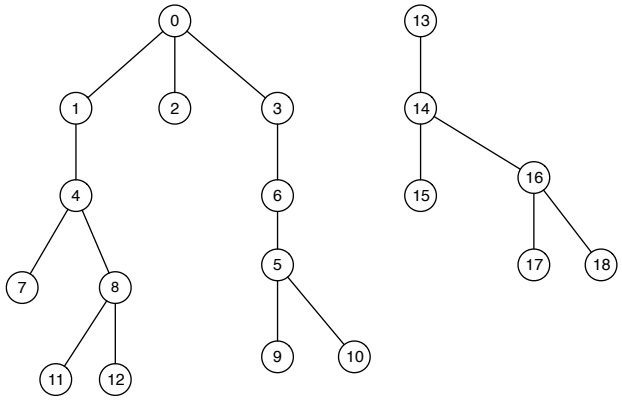
Solution:

```
ALG3(n)
for j = 1 to n do
  i = n
  while i ≥ 3 do
    i = ⌊i/2⌋
  end while
end for
```

Solution:

2 Data Structures

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



2.1 (5 %) State the results of the following FIND(·) operations.

FIND(2) : _____ FIND(8) : _____

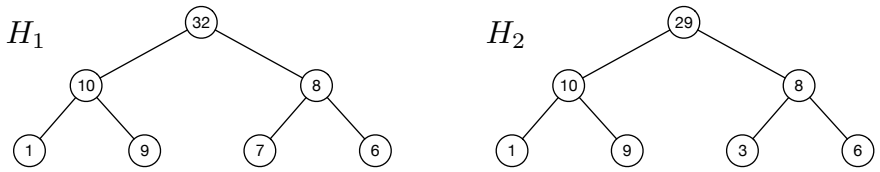
FIND(14) : _____ FIND(12) : _____

FIND(0) : _____ FIND(17) : _____

2.2 (5 %) Assume that we now use path compression on the forest above. Show the forest of trees after a FIND(8) operation.

Solution:

2.3 (5 %) Consider the following max heaps H_1 and H_2.



Draw H_1 after an INSERT operation with key 12.

Solution:

2.4 (5 %) Draw H_2 after an EXTRACT-MAX operation.

Solution:

2.5 (5 %) We want to support the DELETE-MAX() operation on each of the following data structures. DELETE-MAX() should remove the largest element in the data structure and then reestablish the data structure over the remaining elements. As an example, if the data structure is a min heap with elements $\{6, 32, 18, 7, 2\}$, DELETE-MAX() removes element 32 and re-establishes a min heap with the elements $\{6, 18, 7, 2\}$. For each of the data structures, state the time needed to perform a DELETE-MAX() operation in O-notation as a function of n, where n is the number of elements in the data structure.

Singly-linked list: _____

Doubly-linked list: _____

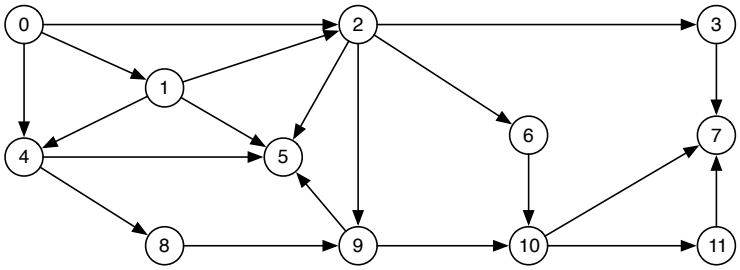
Max heap: _____

Min heap: _____

Binary search tree: _____

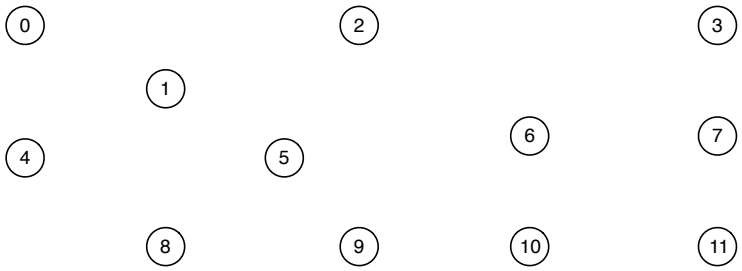
3 Graphs

Consider the following graph G.



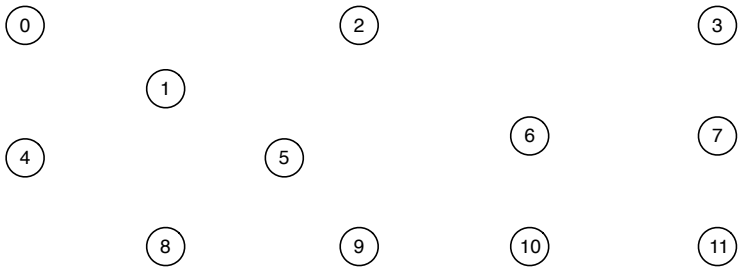
3.1 (5 %) Show the DFS tree for G when starting in node 0 and state the discovery and finish times for each node. Assume that the adjacency lists are sorted in increasing order.

Solution:

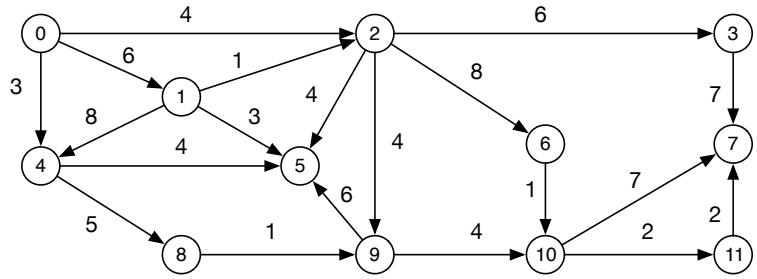


3.2 (5 %) Show the BFS tree for G when starting in node 0 and state the distance for each node. Assume that the adjacency lists are sorted in increasing order.

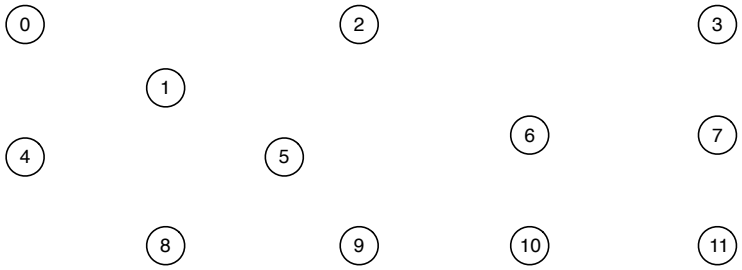
Solution:



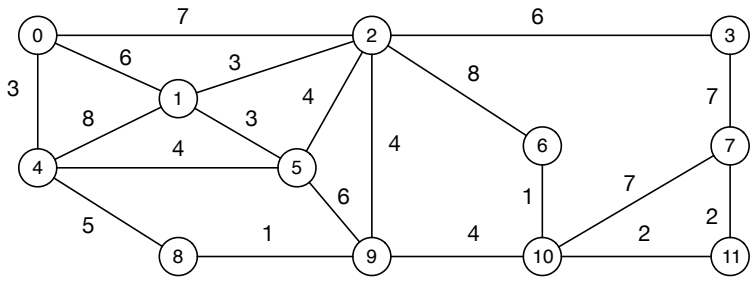
3.3 (5 %) Consider the graph below. Show a shortest path tree for the graph starting at node 0. State the length of the shortest path from node 0 to each node.



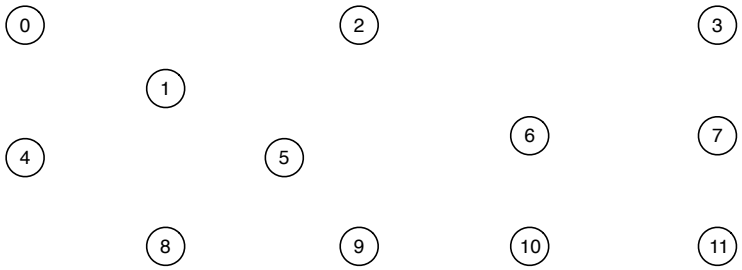
Solution:



3.4 (5 %) Consider the graph below. Show a minimum spanning tree and state the total weight of the tree.



Solution:



Total weight: _____

4 Trees

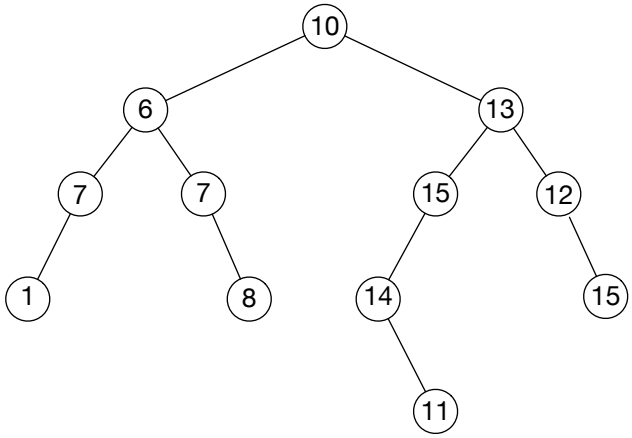
This exercise is about rooted binary trees. Any node x has fields $x.parent$, $x.left$ and $x.right$ denoting the parent, left child, and right child of x, respectively. The root r has $r.parent = null$. Furthermore, any node has a *weight* given by the field $x.weight$.

4.1 (1 %) Let x be a node in the tree. The node x is *skewed* according to the following conditions.

- If x is a leaf then x is skewed.
- If x is an internal node with two children then x is skewed if and only if $x.left.weight < x.weight < x.right.weight$.
- If x is an internal node with a left child but no right child then x is skewed if and only if $x.left.weight < x.weight$.
- If x is an internal node with a right child but no left child then x is skewed if and only if $x.weight < x.right.weight$.

Consider the following tree. Mark all skewed nodes in the tree with a cross.

Solution:



4.2 (5 %) Give an algorithm, SKEWED(X), that given a node x returns true if and only if the node is skewed. Write your algorithm in pseudocode and analyze the running time of your algorithm as a function of n, where n is the number of nodes in the tree.

Solution:

4.3 (7 %) Give a recursive algorithm, SKEWCOUNT(x), that, given the root node, computes the number of skewed nodes in the tree. Write your algorithm in pseudocode and analyze the running time of your algorithm as a function of n, where n is the number of nodes in the tree.

Solution:

5 Super Mario Run

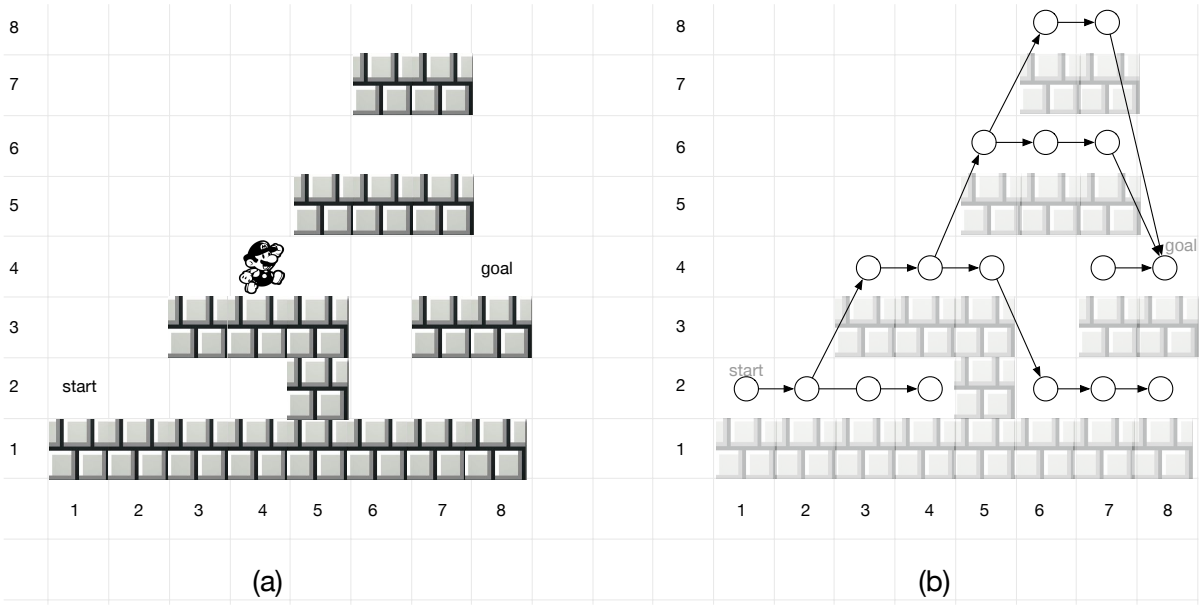


Figure 1: 8×8 Mario world and corresponding Mario graph.

A *Mario world* M consists of a $k \times k$ grid. Each field in the grid is *empty* or *brick*. Two empty fields are marked as *start* and *goal* (see Fig. 5(a)). The goal of the game is to move the player, called Mario, from the start field to the goal field. When Mario is in field (x, y) he has the following options:

Forward Mario moves to the field $(x+1, y)$. This move is possible if $(x+1, y)$ is empty and $(x+1, y-1)$ is brick.

Jump Mario moves to the field $(x+1, y+2)$. This move is possible if $(x+1, y+2)$ is empty, $(x, y+1)$ is empty, and $(x+1, y+1)$ is brick.

Fall Mario moves to the field $(x+1, y')$ where $y' < y$ is the maximal value y' such that $(x+1, y'-1)$ is brick and $(x+1, y), (x+1, y-1), \ldots, (x+1, y')$ are empty.

For instance, Mario in field $(4, 4)$ can move forward to $(5, 4)$ or jump to field $(5, 6)$. A move is *valid* if the move can be done according to the above rules. All fields that can be reached via valid moves from the start field are the *valid fields*. For example, $(4, 4)$ is a valid field, since it can be reached from the start field in $(1, 2)$ doing the sequence of valid moves: forward, forward, jump.

A Mario world M defines a directed graph G with n vertices and m edges (see Fig. 5(b)). All valid fields correspond to a vertex and the valid moves define the edges (there is an edge from field (x, y) to field (x', y') if Mario can move from (x, y) to (x', y') with a valid move). The edges corresponding to forward, jump and fall are denoted by *forward edges*, *jump edges* and *fall edges*.

5.1 (3 %) Let M be a Mario world defined on a $k \times k$ grid and let G be the corresponding Mario graph.

Give the maximum number of nodes n that can appear in G as a function of k.

Solution: ☐ $O(\log k)$ ☐ $O(\sqrt{k})$ ☐ $O(k)$ ☐ $O(k \log k)$ ☐ $O(k^2)$ ☐ $O(2^k)$

Give the maximum number of edges m that can appear in G as a function of k.

Solution: ☐ $O(\log k)$ ☐ $O(\sqrt{k})$ ☐ $O(k)$ ☐ $O(k \log k)$ ☐ $O(k^2)$ ☐ $O(2^k)$

5.2 (6 %) We are now interested in checking if it is possible to go from the start field to the goal field. A Mario graph is *valid* if there exists a path from start to goal. Give an algorithm that given a Mario graph G, decides whether or not there is a path from the start field to the goal field. Analyze the running time of your algorithm as a function of n and m.

Solution:

5.3 (5 %) It's tough work for Mario to jump and fall. Thus, we associate a *price* with each edge e, $price(e)$, such that a forward edge costs 1, a jump edge costs $2\frac{1}{2}$ and a fall edge costs 4. We define the price of a path to be the sum of the prices on the edges of the path. Give an algorithm which, given a valid Mario graph, finds a cheapest path from the start field to the goal field. Analyze the running time of your algorithm as a function of n and m.

Solution:

5.4 (5 %) Now, we're also interested in collecting a *mushroom* during the game. A *mushroom graph* is a valid Mario graph where a single node c is marked as a *mushroom node* and there exists a path from start to goal going through c. Give an algorithm which, given a mushroom graph, finds a cheapest path from the start field to the goal field going through c. Analyze the running time of your algorithm as a function of n and m.

Solution:

5.5 (3 %) A *k-mushroom graph* is a valid Mario graph where k nodes $c_0, \ldots, c_{k-1}$ are marked as mushroom nodes and for each mushroom c_i, $0 \leq i \leq k-1$, there exists a path from start to goal going through c_i. A *mushroom path* in a *k-mushroom graph* is a path from start to goal going through *at least* one of the k mushrooms. Give an algorithm which, given a k-mushroom graph, finds a cheapest mushroom path from the start field to the goal field. Analyze the running time of your algorithm as a function of n, m and k.

Solution: