CS201 Data Structures and Algorithms
End-semester Examination
Answers and Evaluation Plan

Maximum marks: 50

Examination @ 09:00 - 12:00, 23 December 2018

The evaluation of any answer not given in this document will be done on a case by case basis.

1. Let $f(n)$ and $g(n)$ be asymptotically non-negative functions (i.e., there exists a constant c such that $f(n') \geq 0$ and $g(n') \geq 0$ for all $n' \geq c$). Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$. **[4 marks]**

   Sketch of Answer It is sufficient to obtain three positive constants c_1, c_2, and n_0 such that $0 \leq c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n))$ for all $n \geq n_0$. The following constants work. $c_1 = 0.5, c_2 = 1$ and $n_0 = c$, where c is the constant mentioned in the question.

   Evaluation The pair (c_1, n_0) gets 1.5 marks, the pair (c_2, n_0) gets 1.5 marks and if it is clear that the definition of Θ notation is clearly understood, you earn 1 mark. Some of you have assumed that $f(n) \geq g(n)$ for sufficiently large n. Note that this is an inappropriate assumption, as you can come up with two functions which does not satisfy this condition.

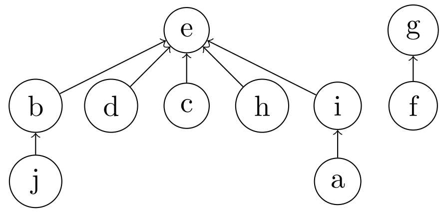
2. Let $A = \{a, b, c, d, e, f, g, h, i, j\}$. Consider Algorithm 1 which builds a disjoint-set forest out of the elements in A. **[4*2=8 marks]**

Algorithm 1

for $x \in A$ **do**
    MAKE-SET(x)
UNION(j, b)
UNION(c, e)
UNION(a, i)
UNION(i, c)
UNION(e, d)
UNION(f, g)
UNION(h, e)
UNION(a, j)

Let rep(x) denote the representative (root) of the tree containing x.

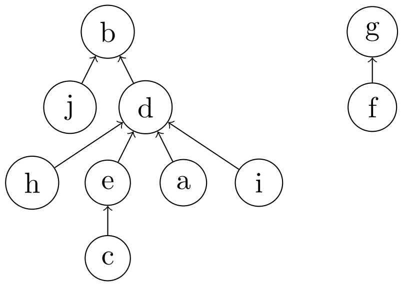
(a) Draw the forest obtained by applying the algorithm with *union by rank* heuristic and without *path compression* heuristic. Assume that, if rank of rep(x) equals rank of rep(y), then UNION(x, y) will make rep(x) a child of rep(y).



Answer

Evaluation Every correct child-parent relation gets 4/10 marks.

(b) Draw the forest obtained by applying the algorithm with *path compression* heuristic and without *union by rank* heuristic. Assume that, UNION(x, y) will make the rep(x) a child of rep(y).



Answer

Evaluation Every correct child-parent relation gets 4/10 marks.

3. Devise an algorithm to construct a binary search tree (BST) where the input of the algorithm is a pre-order traversal of the BST to be constructed. Argue the correctness of the algorithm. Derive its worst-case complexity. **[7 marks]**

Sketch of Answer Clearly, the first element in the pre-order traversal is in the root node. Collect all elements less than the root node value in a list L and all elements greater than the root node value in a list R. Recursively do the same on L and the root of the tree formed is the left child of the root node. Similarly, recursively do the same on R and the root of the tree formed is the right child of the root node. Take care of the boundary conditions. Correctness follows from the properties of pre-order traversals. The worst-case complexity of this algorithm is $O(n^2)$.

Evaluation A correct algorithm will give you 4 marks. A correct complexity analysis (irrespective of the correctness of the algorithm) gives you 2 marks. The correctness arguments give you 1 mark.

4. Consider a digraph (directed graph) represented by the adjacency list given below. **[4*2=8 marks]**

a: d, e, g, h

b: a, f

c: f

d: b, g

e: c, h

f: c, e

g: e

h: c, g

(a) Do a Depth First Search (DFS) of the digraph and mark the discovery time and finish time of every vertex. Exploration of vertices/edges should be done in the order given by the adjacency list.

Answer

a: 1|16
b: 3|14
c: 5|6
d: 2|15
e: 7|12
f: 4|13
g: 9|10
h: 8|11

Evaluation Every correct number gets you 0.25 marks.

(b) Find all strongly connected components of the digraph.

Answer $\{a, b, d\}, \{c, e, f, g, h\}$

Evaluation A correct SCC gets you 2 marks. 0.5 mark will be reduced (if you have scored some marks for this question) if you claim other SCC than these.

5. Prove or disprove the following statements: **[4*2=8 marks]**

(a) There exists a connected and undirected graph which has distinct edge weights such that the edge with the maximum weight is part of every minimum spanning tree of the graph.

Sketch of Answer The statement is true. Any tree with distinct edge weights is an example.

Evaluation Guessing that the statement is true gives you 1 mark. A proof gives you 3 marks.

(b) For every connected undirected graph G with distinct edge weights, the edge with smallest weight is part of every minimum spanning tree of G.

Sketch of Answer The statement is true. The proof is similar to that of Theorem 23.1 in CLRS, which has been discussed in the class. The idea is that if the least-weighted edge is not part of a MST, then you can replace another edge in the tree with the least-weighted edge to get a better MST, which is a contradiction. A common pitfall is to invoke Kruskal's algorithm to prove this. Note that Kruskal's algorithm gives you just one minimum spanning tree of a graph.

Evaluation Guessing that the statement is true gives you 1 mark. A proof gives you 3 marks.

6. Using Extended Euclidean algorithm find the integer y such that $(23 \cdot y)\%63 = 1$. **[4 marks]**

Sketch of Answer When you call Extended-Euclid algorithm with parameter 63 and 23, you get GCD as 1 and the value $x = -4$ and $y = 11$ such that 63*x + 23*y = 1 mod 63. Therefore the answer is 11.

Evaluation 11 gets you 1 mark and the steps gives you 3 marks.

7. Consider Algorithm 2, which is a modified version of Bellman-Ford algorithm. The inputs to the algorithm are G: a directed graph with at least 4 vertices and without any negative-weighted cycle; w: the weight function; and s: the source vertex. Note that, unlike Bellman-Ford, the second outer for-loop runs only $|G.V| - 2$ times. Does this algorithm correctly find shortest path distances from s to every vertex in G? Give proof for your answer. **[4 marks]**

Algorithm 2

```
procedure Modified-Bellman-Ford(G,w,s)
    for v ∈ G.V do
        v.d = ∞
    s.d=0
    for i = 1 to |G.V| − 2 do
        for (u, v) ∈ G.E do
            if v.d > u.d + w(u, v) then
                v.d = u.d + w(u, v)
```

Sketch of Answer An algorithm is correct when it works in all inputs. So, to prove that this algorithm doesn't work, it is enough to provide an example in which it fails. Consider a weighted digraph which is a directed path on 4 vertices $s \longrightarrow a \longrightarrow b \longrightarrow c$. (Assume some weights on the edges). Assume that bc, ab, sa is the order in which the edges are relaxed by the algorithm. Verify that the algorithm doesn't find the shortest distance from s to c correctly.

Evaluation The answer 'No' gives you 1 mark and a proof gives you 3 marks.

8. Devise an algorithm to detect directed cycles in a digraph. Give arguments for the correctness of the algorithm. Derive its worst-case time complexity. **[7 marks]**

Sketch of Answer Run DFS and at any point if a back edge is explored then it indicates a directed cycle. Clearly, the worst-case complexity of this algorithm is $O(|V| + |E|)$.

Evaluation A correct algorithm fetches 4 marks. Correct complexity analysis gives 2 marks and the correctness arguments get you 1 mark.

—o—