Algorithms and Data Structures (IBC027)

January 20, 2020

Whenever an algorithm is required (assignments 2-6), it can be described in pseudocode or plain English, and its **running time and correctness must always be justified. Your explanations may be informal but should always be clear and convincing**. *For problems where an algorithm must be defined (assignments 2-6), 20% of the points is for the correctness argument, and 20% for the analysis of the running time. The grade equals the sum of the scores for all 6 problems below (plus possibly a bonus score for the homework assignments). Success!*

1 Quiz (2 points)

Justify your answers.

Question 1 What is the worst case time complexity for search, insert and delete operations in a general binary search tree?

(A) $\mathcal{O}(n)$ for all

(B) $\mathcal{O}(\log n)$ for all

(C) $\mathcal{O}(\log n)$ for search and insert, and $\mathcal{O}(n)$ for delete

(D) $\mathcal{O}(\log n)$ for search, and $\mathcal{O}(n)$ for insert and delete

Question 2 The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the binary search tree (the height is the maximum number of edges from a leaf node to the root)?

(A) 2

(B) 3

(C) 4

(D) 6

Question 3 A data structure is required for storing a set of integers such that each of the following operations can be done in $O(\log n)$ time, where n is the number of elements in the set.

1. Deletion of the smallest element
2. Insertion of an element if it is not already present in the set

Which of the following data structures can be used for this purpose?

(A) A heap can be used but not a balanced binary search tree

(B) A balanced binary search tree can be used but not a heap

(C) Both balanced binary search tree and heap can be used

(D) Neither balanced search tree nor heap can be used

Question 4 Which of the following is true about Red Black Trees?

(A) The path from the root to the furthest leaf is no more than twice as long as the path from the root to the nearest leaf

(B) At least one children of every black node is red

(C) Root may be red

(D) A leaf node may be red

Question 5 A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(A) 46, 42, 34, 52, 23, 33

(B) 34, 42, 23, 52, 33, 46

(C) 46, 34, 42, 23, 52, 33

(D) 42, 46, 33, 23, 34, 52

Question 6 Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

(A) $(97 \times 97 \times 97)/10^6$

(B) $(99 \times 98 \times 97)/10^6$

(C) $(97 \times 96 \times 95)/10^6$

(D) $(97 \times 96 \times 95)/(3! \times 10^6)$

Question 7 Consider a binary max-heap implemented using an array. Which one of the following arrays represents a binary max-heap?

(A) 25,12,16,13,10,8,14

(B) 25,14,13,16,10,8,12

(C) 25,14,16,13,10,8,12

(D) 25,14,16,13,10,8,12

Question 8 Which of the following data structures is best suited for efficient implementation of priority queue? (A) Array (B) Linked List (C) Heap (D) Stack

(A) Array

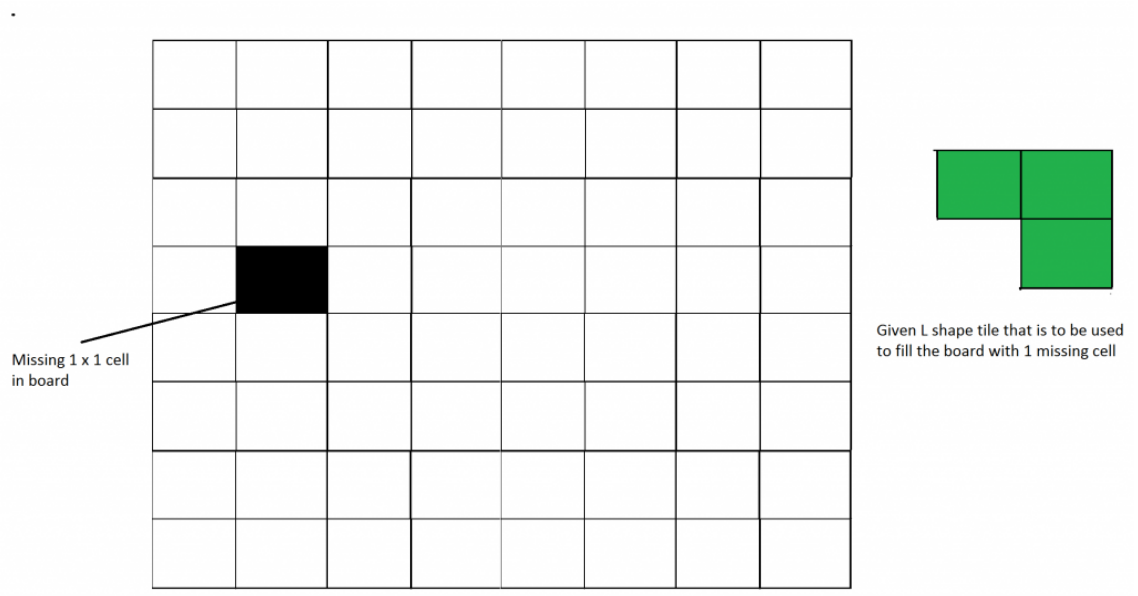
(B) Linked List

(C) Heap

(D) Stack

2 Divide and conquer (1.5 points)

Let us consider an extremely simplified version of Tetris, with a unique simplified L-shaped tile and a board that consists of $2^n \times 2^n$ cells, for some positive integer n. The game is to place as many tiles (the shape is given below, they can be rotated) as possible on the board. We suppose that there is exactly 1 designated cell on the board that is "missing" and that cannot be covered.


Missing 1 x 1 cell in board
Given L shape tile that is to be used to fill the board with 1 missing cell

Describe and analyze a divide-and-conquer algorithm to fully cover the board (except for the missing cell) with L-tiles. The input of the algorithm consists of the value for n and the coordinates of the missing cell, and the output is a list of tiles, where each tile is represented by the three cells on the board that it covers. For instance, for input $(1, (2, 2))$ the corresponding output is $(\{(1, 1), (1, 2), (2, 1)\})$.

3 Graphs (1.5 points)

Let $G = (V, E)$ be an undirected graph. For $u, v \in V$, let $d(u, v)$ be the length of the shortest path from u to v if it exists, and ∞ otherwise. Let $U \subseteq V$ be a nonempty set of vertices. Then $d(U, v)$ denotes how far set U is from vertex v:

$$d(U, v) \quad = \quad \min_{u \in U} \; d(u, v).$$

The *eccentricity* of U, written $\epsilon(U)$, denotes how far set U is from the vertex in V that is most distant from it in the graph:

$$\epsilon(U) \quad = \quad \max_{v \in V} \; d(U, v).$$

Intuitively, if G is a country and U is the part of it that you know, $\epsilon(U)$ gives the largest distance from the places you know to a place you haven't seen yet. Describe and analyze an efficient algorithm to compute the eccentricity of U.

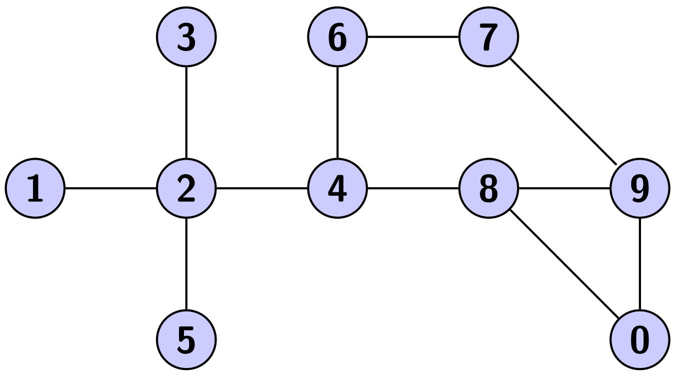


Figure 1: In this graph $d(1, 0) = 4$, $\epsilon(\{1\}) = 4$ and $\epsilon(\{1, 2, 4\}) = 2$.

4 Dynamic programming (2 points)

In the *longest increasing subsequence* problem, the input is a sequence of numbers $s = a_1, \dots, a_n$. A *subsequence* is any sequence obtained from s by omitting zero or more elements, and an *increasing* subsequence is one in which the numbers are getting strictly larger. The task is to find an increasing subsequence of greatest length.

For instance, the longest increasing subsequence of 5, 2, 8, 6, 3, 6, 9, 7 is 2, 3, 6, 9.

1. Describe and analyze an $\mathcal{O}(n^2)$-time dynamic programming algorithm for the longest increasing subsequence problem.

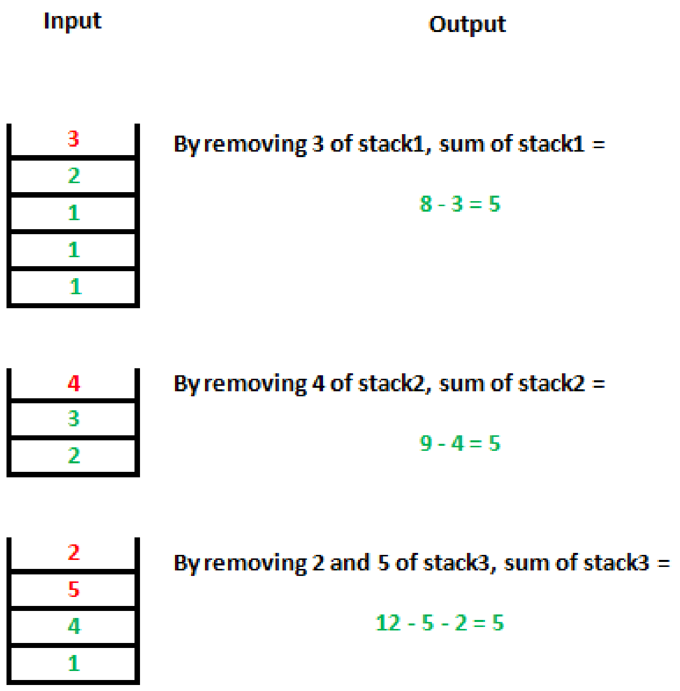
2. Describe and analyze (possibly using a different approach than dynamic programming) an $\mathcal{O}(n \log n)$-time algorithm for the longest increasing subsequence problem.

5 Network flow (1.5 points)

Suppose you are organizing a conference where researchers present articles they have written. Researchers who want to present an article send a paper to the conference organizers. The conference organizers have access to a set A of reviewers who are each willing to read up to m_A articles. Let B the set of papers to review: each gets reviewed by up to m_B reviewers. Moreover, each submission has a particular topic and each reviewer has a specialization for a set of topics, so papers on a given topic only get reviewed by those reviewers who are experts on that topic. The conference organizers need to decide which reviewers will review each article (or equivalently, which articles will be reviewed by which reviewers). Describe and analyze an efficient algorithm that solves this problem.

6 Greedy (1.5 points)

You are given three stacks that contain positive numbers. You are allowed to remove, repeatedly, the top element of one of the stacks. Your task is to do this in such a way that the sum of the remaining numbers for each of the stacks is equal and maximal. Note that if we remove all the numbers from all the stacks, the sum of the remaining numbers (there aren't any) in all of the stacks is equal to 0; often it will be possible to obtain an equal sum that is larger by removing fewer elements. The diagram below shows an example. Describe and analyze a greedy algorithm to compute the maximum equal sum.


Input
Output
3
2
1
1
1
By removing 3 of stack1, sum of stack1 =
8 - 3 = 5
4
3
2
By removing 4 of stack2, sum of stack2 =
9 - 4 = 5
2
5
4
1
By removing 2 and 5 of stack3, sum of stack3 =
12 - 5 - 2 = 5