COMP20007 Design of Algorithms 2020
Additional Problems for Weeks 1 – 5

Notes

This text lists some additional exercises/problems for COMP20007 weeks 1–5. The exercises are taken or adapted from:

L Anany Levitin. *Introduction to the Design and Analysis of Algorithms* Pearson, 2012.

DPV Dasgupta, Sanjoy, Christos H. Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms* McGraw-Hill, 2008.

It should be noted the other text book of the course:

Steven S Skiena. *The Algorithm Design Manual* Springer, 2008

contains various useful exercises, including many job interview questions. The book is available for download from the library. Moreover, its list of exercises and solutions is available at `http://www.algorist.com/algowiki/index.php/The_Algorithms_Design_Manual_%28Second_Edition%29`.

In the rest of this document, the following notations are adopted:

LS-*n*.pdf – Lecture slide set ***n*.pdf**.

Lc.s-*n* – exercise ***n*** from secstion **c.s** of Levitin's book.

DPV-*s.n* – exercise ***s.n*** in the DPV book.

Other – other exercise.

The additional **(A)** at the end of exercise number, if any, indicates that it has been adapted.

1 LS-1.pdf and LS-2.pdf: Algorithms Design and Fundamental Data Structures

Described Text

Levitin's Chapters 1.

Additional Exercises

L1.1-5: Design an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5 and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively?

L1.4-2: If you have to solve the searching problem for a list of n numbers, how can you take advantage of the fact that the list is known to be sorted? Give separate answers for

(a) lists represented as arrays.

(b) lists represented as linked lists.

Other-1 Our convetional arithmetic expression employs the *infix notation*, where an *operator* such as +, - is written between its *operands* such as $10, 15$. For example, $3 * (5 + 8)$ is an infix expression.

Another way is to use *postfix notation* (also known as *reverse Polish notation*), where an operator is written immediately after its operands. For example, the above expression is rewritten as $3\ 5\ 8 * +$. Note that there is no need to use brackets in postfix expressions.

What data structure can be used to evaluate (ie. compute the value of) postfix expressions? Give a pseudocode for the algorithm. You can assume that there is a function `get_next_token()` that returns the next token (ie. next opertor or operand) from the input stream.

L1.4-10: *Anagram checking* Design an algorithm for checking whether two given words are anagrams, i.e., whether one word can be obtained by permuting the letters of the other. For example, the words `tea` and `eat` are anagrams.

2 LS-3.pdf and LS-4.pdf: Growth Rate, Algorithm Efficiency & Analysis

Described Text

Levitin's Chapter 2.

Additional Exercises

L1.1-5: Design an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5 and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively?

L2.1-6: Suggest how any sorting algorithm can be augmented in a way to make the best-case count of its key comparisons equal to just $n - 1$ (n is a list's size, of course). Do you think it would be a worthwhile addition to any sorting algorithm?

L2.2-5(A): . List the following functions according to their order of growth from the lowest to the highest:

$$(n-2)!,\ n^{0.001},\ 5\lg(n+100)^{10},\ 2^{2n},\ (\log n)^{\log n},\ 0.001n^4 + 3n^3 + 1,\ n/\log n,\ \ln^2 n,\ \sqrt[3]{n},\ 3^n.$$

While listing, separate neighboured functions by an empty space if they have the same order of growth, and by a comma otherwise.

DPV 0.2: Show that, if c is a positive real number, then $g(n) = 1 + c + c^2 + \cdots + c^n$ is:

(a) $\Theta(1)$ if $c < 1$.

(b) $\Theta(n)$ if $c = 1$.

(c) $\Theta(c^n)$ if $c > 1$.

The moral: in big-Θ terms, the sum of a geometric series is simply the first term if the series is strictly decreasing, the last term if the series is strictly increasing, or the number of terms if the series is unchanging.

L2.2-10: The range of a finite nonempty set of n real numbers S is defined as the difference between the largest and smallest elements of S. For each representation of S given below, describe in English an algorithm to compute the range. Indicate the time efficiency classes of these algorithms using the most appropriate notation (O, Θ, or Ω).

a. An unsorted array

b. A sorted array

c. A sorted singly linked list

d. A binary search tree

DPV-2.4(A): Suppose you are choosing between the following three algorithms:

Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.

Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $\Theta(n^2)$ time.

What are the running times of each of these algorithms (in big-Θ notation), and which would you choose?

L2.4-10(A): Consider the following algorithm to check whether a graph defined by its adjacency matrix is complete.

```
ALGORITHM  GraphComplete(A[0..n − 1, 0..n − 1])
    //Input: Adjacency matrix A[0..n − 1, 0..n − 1]) of an undirected graph G
    //Output: 1 (true) if G is complete and 0 (false) otherwise
    if n = 1 return 1   //one-vertex graph is complete by definition
    else
        if not GraphComplete(A[0..n − 2, 0..n − 2]) return 0
        else for j ← 0 to n − 2 do
                 if A[n − 1, j] = 0 return 0
             return 1
```

a) Convince yourselves that the algorithm does actually do its job.

b) What is the algorithm's efficiency class in the worst case?

c) What is the best case for this algorithm? What is the algorithm's efficiency class in the best case?

d) Write the equivalent/similar iterative algorithm. What is the algorithm's efficiency class in the best case?

3 LS-5.pdf: Brute Force Methods

Described Text

Levitin's Chapter 3. In particular, sections 3.1 (Selection Sort), 3.2 (String Matching), 3.3 (Closest-Pair) and 3.4 (Exhaustive Search).

Additional Exercises from Levitin's Book:

L3.1-4: (a) Design a brute-force algorithm for computing the value of a polynomial:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

at a given point x_0 and determine its worst-case efficiency class.

(b) If the algorithm you designed is in $\Theta(n^2)$, design a linear algorithm for this problem.

(c) Is it possible to design an algorithm with a better-than-linear efficiency for this problem?

L3.2-5: How many comparisons (both successful and unsuccessful) will be made by the brute-force algorithm in searching for each of the following patterns in the binary text of one thousand zeros?

a. 00001 b. 10000 c. 01010

L3.4-8: Explain how exhaustive search can be applied to the sorting problem and determine the efficiency class of such an algorithm.

4 LS-6.pdf & LS-7.pdf: Graphs, DFS & BFS

Described Text

Levitin's sections 1.4 (Graphs), 3.5 (DFS & BFS) and 4.2 (Topological Sorting).

Additional Exercises from Levitin's Book:

Other-1: Suppose that graphs are used in modelling:

(a) The road network between major towns/cities in Australia.

(b) The Internet.

(c) Possible transmission of coronavirus amongst people residing in a country.

(d) The relationship ”A knows B's name” between the members of a government's cabinet.

In each case indicate if the properties of the corresponding graph such as being dense or sparse, connected or unconnected, cyclic or acyclic, directed or undirected. You can add some (reasonable) assumptions if needed.

Other-2: Given an adjacency-list representation of a directed graph, design algorithms to compute a) the out-degree of every vertex, and b) the in-degrees of every vertex. What is the efficiency of each algorithm?

L3.5-6: a) Explain how one can check a graph's acyclicity by using breadth-first search.

b) Does either of the two traversals—DFS or BFS—always find a cycle faster than the other? If you answer yes, indicate which of them is better and explain why it is the case; if you answer no, give two examples supporting your answer.

L3.5-11(A): *Three Jugs* Siméon Denis Poisson (1781–1840), a famous French mathematician and physicist, is said to have become interested in mathematics after encountering some version of the following old puzzle. Given an 8-pint jug full of water and two empty jugs of 5- and 3-pint capacity, get exactly 4 pints of water in one of the jugs by completely filling up and/or emptying jugs into others.

a) Model this as a graph problem: give a definition of the graph involved and state the question about this graph that needs to be answered.

b) Can we solve the puzzle using DFS or BFS?

c) Solve the puzzle by applying the appropriate algorithm.

L4.2-7(A): One way to perform topological sorting on a digraph is to repeatedly find a vertex of in-degree 0 (a `source`), output it, and remove it and all of its outgoing edges from the graph. Can you implement this source-removal algorithm for a digraph represented by its adjacency lists so that its running time is in $O(|V| + |E|)$. What happens to this algorithm if G has cycles?

5 LS-8.pdf: Greedy Algorithms: Prim and Dijkstra

Described Text:

Levitin's Chapter 9, sections 9.1 (Spanning Trees and MST, Prim's Algorithm), 9.3 (Dijkstra's Algorithm); Also see 1.4 and 6.4 for Priority queues

Additional Exercises

L9.1-14: Prove that any weighted connected graph with distinct weights has exactly one minimum spanning tree.

DPV-5.9: The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

(a) If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

(b) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.

(c) Let e be any edge of minimum weight in G. Then e must be part of some MST.

(d) If the lightest edge in a graph is unique, then it must be part of every MST.

(f) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

Other: A city council want to build a bycicle track that connects all of the city's n major business buildings, while minimizing the construction cost. As a graduate, your task is to help the council in approaching the problem and designing the track.

DPV-4.8: Professor F. Lake suggests the following algorithm for finding the shortest path from node s to node t in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra's algorithm starting at node s, and return the shortest path found to node t.

Is this a valid method? Either prove that it works correctly, or give a counterexample.

DPV-4.13: You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, l_e. You want to get from city s to city t. There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $l_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in linear time whether there is a feasible route from s to t.

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t. Give an $O((|V| + |E|) \log |V|)$ algorithm to determine this.

Date: April 9, 2020