473 Levitin problems and hints   HW 06

Problem 1:   (6) 3.5.3 [5.2.3] Independence of properties from specific DFS traversals. Explain your answers.

3. Let G be a graph with n vertices and m edges.

   a. True or false: All its DFS forests (for traversals starting at different vertices) will have the same number of trees?

   b. True or false: All its DFS forests will have the same number of tree edges and the same number of back edges?
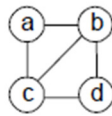
Author's Hints:

3. a. What is the number of such trees equal to?

   b. Answer this question for connected graphs first.

Problem 2: (10) 3.5.8a [5.2.8a]   Bipartite graph checking using DFS

8. A graph is said to be **bipartite** if all its vertices can be partitioned into two disjoint subsets X and Y so that every edge connects a vertex in X with a vertex in Y. (We can also say that a graph is bipartite if its vertices can be colored in two colors so that every edge has its vertices colored in different colors; such graphs are also called **2-colorable**). For example, graph (i) is bipartite while graph (ii) is not.
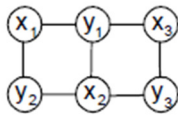
(i)   (ii)

a. Design a DFS-based algorithm for checking whether a graph is bipartite.

b. Design a BFS-based algorithm for checking whether a graph is bipartite.

Author's Hints:

8. Use a DFS forest and a BFS forest for parts (a) and (b), respectively.

Problem 3: (5) 4.1.1 [5.1.1]   Ferrying Soldiers

1. *Ferrying soldiers*   A detachment of n soldiers must cross a wide and deep river with no bridge in sight.   They notice two 12-year-old boys playing in a rowboat by the shore.   The boat is so tiny, however, that it can only hold two boys or one soldier.   How can the soldiers get across the river and leave the boys in joint possession of the boat? How many times need the boat pass from shore to shore?

Author's Hints:

1. Solve the problem for $n = 1$.

Problem 4: (5) 4.1.4 [5.1.3] generate power set

4. Design a decrease-by-one algorithm for generating the power set of a set of n elements. (The power set of a set S is the set of all the subsets of S, including the empty set and S itself.)

Author's Hints:

4. Use the fact that all the subsets of an n-element set $S = \{a_1, ..., a_n\}$ can be divided into two groups: those that contain a_n and those that do not.

Problem 5: (5) (not in book) [5.1.9]   binary insertion sort efficiency

9. ▷ Binary insertion sort uses binary search to find an appropriate position
   to insert $A[i]$ among the previously sorted $A[0] \leq \ldots \leq A[i-1]$. Determine
   the worst-case efficiency class of this algorithm.

Author's Hints:

9. The order of growth of the worst-case number of key comparisons made
   by binary insertion sort can be obtained from formulas in Section 4.3
   and Appendix A. For this algorithm, however, a key comparison is not
   the operation that determines the algorithm's efficiency class.   Which
   operation does?

Problems 6: (9) 4.2.6 [5.3.6] dag source

> In the 3rd edition, it says "Prove that a
> **nonempty** dag must have at least one
> source." That additional word is necessary!

6. a. Prove that a dag must have at least one source.

   b. How would you find a source (or determine that such a vertex does
   not exist) in a digraph represented by its adjacency matrix? What is the
   time efficiency of this operation?

   c. How would you find a source (or determine that such a vertex does
   not exist) in a digraph represented by its adjacency lists? What is the
   time efficiency of this operation?

Author's Hints:

6. a. Use a proof by contradiction.

   b. If you have difficulty answering the question, consider an example
   of a digraph with a vertex with no incoming edges and write down its
   adjacency matrix.

   c. The answer follows from the definitions of the source and adjacency
   lists.

Problem 7: (9) 4.2.9 [5.3.9] Strongly connected components of a digraph

9. A digraph is called **strongly connected** if for any pair of two distinct ver-
tices u and v, there exists a directed path from u to v and a directed path

from v to u. In general, a digraph's vertices can be partitioned into dis-
joint maximal subsets of vertices that are mutually accessible via directed
paths of the digraph; these subsets are called **strongly connected com-
ponents**.   There are two DFS-based algorithms for identifying strongly
connected components.   Here is the simpler (but somewhat less efficient)
one of the two:

Step 1. Do a DFS traversal of the digraph given and number its vertices
in the order that they become dead ends.

Step 2. Reverse the directions of all the edges of the digraph.

Step 3. Do a DFS traversal of the new digraph by starting (and, if neces-
sary, restarting) the traversal at the highest numbered vertex among still
unvisited vertices.

The strongly connected components are exactly the subsets of vertices in
each DFS tree obtained during the last traversal.

a. Apply this algorithm to the following digraph to determine its strongly
connected components.
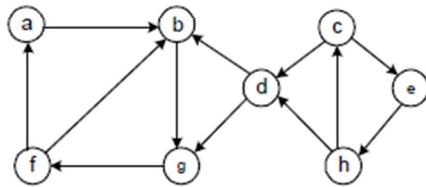
b.   What is the time efficiency class of this algorithm?   Give separate
answers for the adjacency matrix representation and adjacency list repre-
sentation of an input graph.

c. How many strongly connected components does a dag have?

Author's Hints:

9. a. Trace the algorithm on the input given by following the steps of the
algorithm as indicated.

b. Determine the efficiency for each of the three principal steps of the
algorithm and then determine the overall efficiency.   Of course, the an-
swers depend on whether a digraph is represented by its adjacency matrix
or by its adjacency lists.

Problem 8: (15) Not in book

(Miller-Rabin test) Let $N = 1729$ for all parts of this problem.
(a) How many values of a in the range $1..1728$ pass the Fermat test [i.e. $a^{1728} \equiv 1 \pmod{1729}$]?
(b) For how many of the "Fermat positive" values of a from part (a) does the Miller-Rabin test
provide a witness that N is composite?
(c) If we pick a at random from among $1, 2,...N$, what is the probability that running the Miller-Rabin
test on a will show that N is composite? Rabin showed that for any N, the probability is at least
75%, what is it for this case?

[**Hint**: writing a few lines of code my help you here. If you do that, include them in your submission].

Problem 9: (9) 4.3.2 [5.4.2] Examples of permutation generation algorithms

2. Generate all permutations of $\{1, 2, 3, 4\}$ by

    a. the bottom-up minimal-change algorithm.

    b. the Johnson-Trotter algorithm.

    c. the lexicographic–order algorithm.

You do not have to write any code, but you can do it that way if you wish.

Author's Hints:

2. We traced the algorithms on smaller instances in the section.

Problem 10: (10) 4.3.10 [5.4.10]   Generation of all k-combinations from an n-element set

10. ▶ Design a decrease-and-conquer algorithm for generating all combina-
    tions of k items chosen from n, i.e., all k-element subsets of a given n-
    element set. Is your algorithm a minimal-change algorithm?

Author's Hints:

10. There are several decrease-and–conquer algorithms for this problem. They
    are more subtle than one might expect. Generating combinations in a pre-
    defined order (increasing, decreasing, lexicographic) helps with both a de-
    sign and a correctness proof. The following simple property is very helpful.
    Assuming with no loss of generality that the underlying set is $\{1, 2, ..., n\}$,
    there are $\binom{n-i}{k-1}$ k-subsets whose smallest element is i, $i = 1, 2, ..., n-k+1$.

Problem 11: (10) 4.3.11 [5.4.11]

11. *Gray code and the Tower of Hanoi*

    (a) ▷ Show that the disk moves made in the classic recursive algorithm
        for the Tower-of-Hanoi puzzle can be used for generating the binary
        reflected Gray code.

    (b) ▶ Show how the binary reflected Gray code can be used for solving
        the Tower-of-Hanoi puzzle.

Author's Hints:

11. Represent the disk movements by flipping bits in a binary n-tuple.