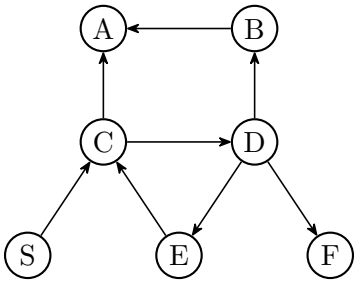
COMP S265F Design and Analysis of Algorithms
Take-home Assignment
25 May, 2021 (Tue)
00:00 - 23:59

This paper contains **SEVEN** questions. Please answer **ALL** of them.

You are required to handwrite your answers on blank papers, take photos on them using your smartphone, and convert them to a PDF file for submission to the submission page in the OLE. Note that computer-typed answers will not be accepted.

Please submit your program `q7.py` and your PDF answer to both the OLE and lklee@study.ouhk.edu.hk with email title **"[COMPS265F] Take-home assignment answers"**.

Question 1 (10 marks). Perform a Depth-First Search (DFS) on the following directed graph, using vertex S as the source.

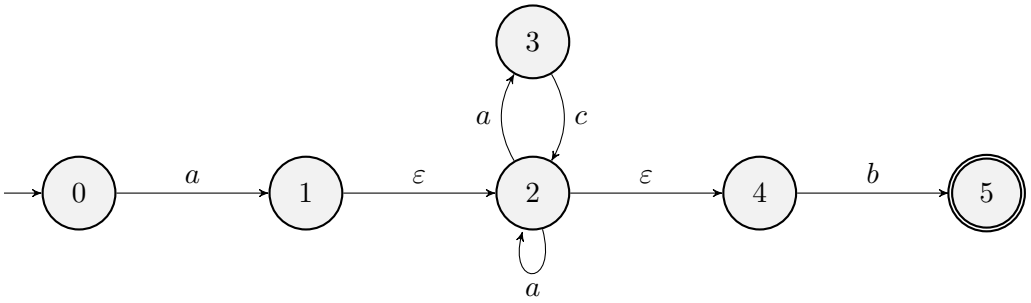


(a) List the vertices in the discovered order of DFS, and show for eact vertex v, its discovery time $d[v]$, finish time $f[v]$ and depth-first tree parent $\pi[v]$ in the DFS. Then, draw the depth-first tree obtained. [6]

(b) List all the edges and show their classifications (tree edge, back edge, forward edge, cross edge). [2]

(c) Argue whether the above directed graph has a topological sort or not. If yes, show the order of vertices in the topological sort. If no, justify your answer. [2]

Question 2 (15 marks). Let $\Sigma = \{a, b, c\}$ be the input alphabet of the following NFA with ε moves:



(a) Write down the transition table of the above NFA including the lambda closure of the states. [5]

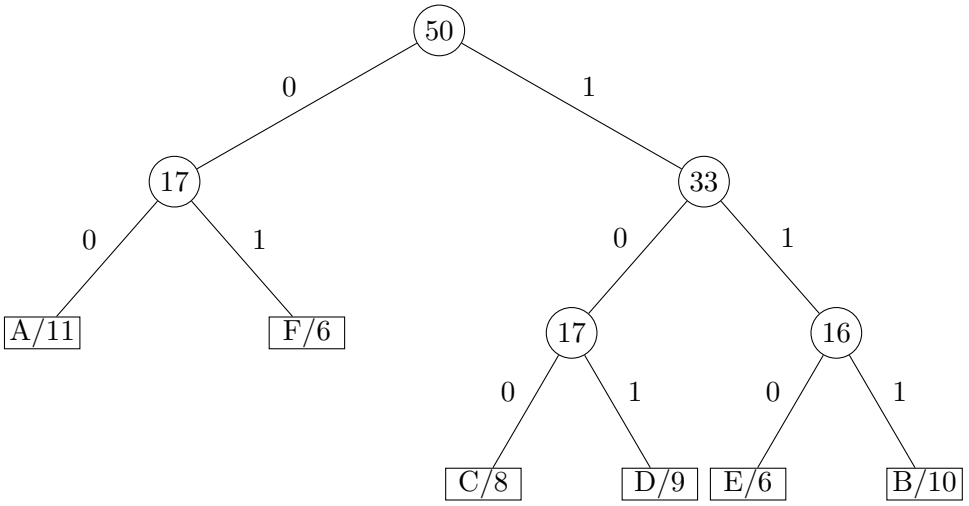
(b) Transform the above NFA to a DFA. Write down the transition table of the DFA, and reduce the number of states in the DFA (if any). Hence, draw the diagram of the DFA. [10]

Question 3 (5 marks). Use the pumping lemma to prove that $L = \{a^n b^n c^n | \ n \in \mathbb{N}\}$ is not regular.

Question 4 (15 marks). Given a set of characters A, B, C, D, E, F and their corresponding frequencies.

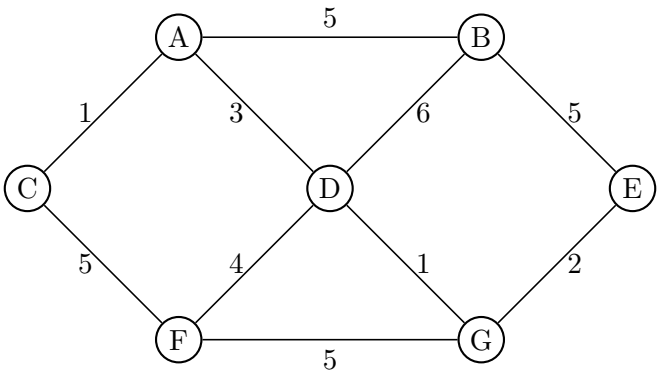
Character	A	B	C	D	E	F
Frequency	11	10	8	9	6	6

(a) Determine whether the following code tree has the minimum average character length. Justify your answer. [5]



(b) Construct the Huffman code for the above characters. Show the merging steps clearly and draw the code tree. Hence, compute the average character length of the Huffman code. [10]

Question 5 (20 marks). Consider the following weighted graph:



(a) What is the weight of its minimum spanning tree? [5]

(b) Suppose Kruskal's algorithm is run on this graph. In what order are the edges added to the MST? For each edge in this sequence, give a cut that justifies its addition. [10]

(c) Does the above graph have more than one minimum spanning tree? Justify your answer. [5]

Question 6 (15 marks). Consider the following algorithm that works on an list `L` of $n \geq 2$ numbers, sorted in ascending order:

```
def func(L):
    if len(L) == 1:
        return float('inf')
    if len(L) == 2:
        return L[1] - L[0]
    m = len(L)//2
    return min(func(L[:m]), func(L[m:]), L[m] - L[m-1])
```

(a) If `L = [1, 3, 7, 8, 10, 12]`, what is the output of `func(L)`? [2]

(b) What is the problem that `func(L)` solves? [3]

(c) Prove that `func(L)` can correctly solve the problem in (b). [5]

(d) Find the time complexity $T(n)$ of `func(L)`, where $n = |\texttt{L}|$.
(You may assume n is a power of 2 for simplification.) [5]

Question 7 (20 marks). Finding subsequence in a long sequence has applications in many areas, e.g., financial analysis. Given a sequence $T[0..n-1]$ of $n \geq 1$ strings and a candidate subsequence $S[0..m-1]$ of m strings (where $1 \leq m \leq n$), S is a subsequence of T if it is possible to delete some strings from T such that the remaining strings in T equal to the sequence S. Note that the strings in S should appear in T in order, but not necessarily consecutively.

You are given the following program q7.py:

```
from typing import List

def subseq(s: List[str], t: List[str]) -> bool:
    '''Return True if s is a subsequence of t;
    return False otherwise
    '''

if __name__ == '__main__':
    t = ["wake up", "sleep", "wake up", "eat", "sleep"]
    s1 = ["wake up", "eat"]
    s2 = ["eat", "eat"]
    print(subseq(s1, t))
    print(subseq(s2, t))
```

(a) Design and implement a linear time algorithm for the function `subseq(s,t)` **without changing other code in** q7.py such that it returns `True` if `s` is a subsequence of `t`, and `False` otherwise. [15]

(b) Show that your `subseq(s,t)` has a time complexity of $O(n)$. [5]

[End of Paper]