Algorithm and Operating Systems
End Sem Examination

Nov, 2024

PART A: Mention True/False followed by justification. [3 X 5 marks]

1. To find the all pairs shortest path in a complete graph (where every vertex has an edge to every other vertex), the Floyd Warshall algorithm has lower asymptotic running time than the Johnsons algorithm.

2. Given a connected undirected weighted graph G; let vertex u be an arbitrary vertex in G. The least cost edge incident on vertex u will definitely be part of some MST of G.

3. If the DFS of a directed graph contains a back edge then doing a DFS starting at a different start node will also contain at least one back edge.

4. For two vertices u and v in a directed graph, if the DFS finishing time is such that, f[u] > f[v] and u and v are in the same DFS tree in the DFS forest; then, u is an ancestor of v in the depth first tree.

5. Consider a weighted, directed acyclic graph G = (V, E, w) in which edges that leave the source vertex s may have negative weights and all other edge weights are nonnegative. Dijkstra's algorithm correctly computes the shortest-path weight δ(s, t) from s to every vertex t in this graph

PART B 10 X3

Q1: For any weighted graph G = (V, E,w) and integer k, define Gk to be the graph that results from removing every edge in G having weight k or larger. Given a connected undirected weighted graph G = (V, E, w), where every edge has a unique integer weight, describe an O(|E| log |E|)-time algorithm to determine the largest value of k such that Gk is not connected.

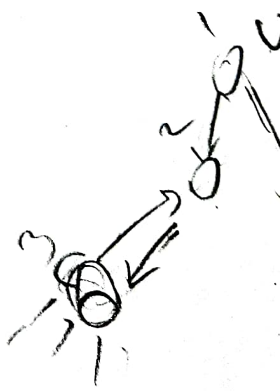
Q2: A word search puzzle consists of an n×n matrix of characters and a "word bank" containing m words each length k. Thus, the total size of the input is $\Theta(n^2 + mk)$. Each of the m words in the word bank is hidden in the matrix, either horizontally or vertically (not diagonally). Note that the words may be hidden in reverse also, as seen in the case of SORT and DICT in the example below. Describe an algorithm that finds all the words in the word bank in the and analyze its runtime in terms of n, m, and k.

Example:

n

H	T	A	Q	A	W	O	T
A	R	R	L	L	C	I	C
S	M	D	O	G	R	S	I
H	A	I	G	O	A	H	D
X	T	B	T	I	E	H	D
N	H	D	M	H	E	A	P
U	K	Y	O	O	C	C	W
X	Q	T	R	O	S	D	K

m =

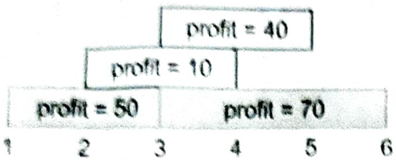


Word bank: ALGO, DICT, HASH, HEAP, MATH, SORT

Q3. We have n jobs where every job is scheduled to be done from startTime[i] to endtime[i] obtaining a profit of profit[i]. You are given the startTime, endTime and profit arrays. Return the maximum profit you can take such that there are no two jobs in the subset have overlapping time ranges.

If you choose a job that ends at time x you will be able to start another job that starts at time X as well.

Example:



Input: StartTime=[1,2,3,3], endTime=[3,4,5,6], profit=[50,10,40,70]
Output: 120
Explanation: The subset chosen is the first and fourth job.
Time range [1-3]+[3-6] we get profit of 120=50+70

PART C (Operating Systems) 5X3

Q1) Each of a set of n processes executes the following code using two semaphores a and b initialized to 1 and 0, respectively. Assume that count is a shared variable. The code of each process looks as below :

Code Section A:

```
{ wait(a)
Count=Count+1
if(Count == n) signal(b)
signal(a)
wait(b)
signal(b) }
```

Code Section B:

```
{}
```

The code achieves which among the below options (one or more options can be correct). Explain.

a. It ensures all processes execute section A mutually exclusively
b. It ensures that at most two processes are in section B at any time
c. It ensures no process executes Section B until everyone finishes Section A.
d. It ensures that at most n-1 processes are in Section A at any time

Q2) Fetch_And_Add (X, i) is an atomic Read-Modify-Write instruction that reads the value of memory location X, increments it by the value i, and returns the old value of X. It is used in the pseudocode shown below to implement a busy-wait lock. L is an unsigned integer shared variable initialized to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

```
AcquireLock(L){                                   Release Lock(L) {
    While (Fetch_And_Add(L, 1 ))                      L = 0 ;
    L = 1 ; }                                     }
```

This implementation

a. Fails as L can overflow
b. Fails as L can take a non zero value when the lock is actually available
c. Works correctly but may starve certain processes
d Works correctly without starvation

Q3) Consider a computer system with 57-bit virtual addressing using multi-level tree-structured page tables with L levels for virtual to physical address translation. The page size is 4 KB(1 KB=1024 B) and a page table entry at any of the levels occupies 8 bytes. The value of L is ?