LAHORE UNIVERSITY OF MANAGEMENT SCIENCES

CS 310 - Algorithms Midterm Exam [Time: 75 mins]
Wednesday, October 23, 2019. [Total Marks = 40]

In multiple choice questions, select ALL choices that apply. Full marks will only be assigned if *all* correct choices are selected.

Q1(A) Which of the following statements are **TRUE**? **Encircle** the **correct** choice(s). [1 mark]

A	$O(f(n)) = \{T(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cT(n)$ for all $n \geq n_0\}$
B	$O(f(n)) = \{T(n)$: there exist positive constants c and n_0 such that $0 \leq cT(n) \leq f(n)$ for all $n \geq n_0\}$
C	$O(f(n)) = \{T(n)$: there exist positive constants c and n_0 such that $0 \leq T(n) \leq cf(n)$ for all $n \geq n_0\}$
D	$O(f(n)) = \{T(n)$: there exist positive constants c and n_0 such that $0 \leq cf(n) \leq T(n)$ for all $n \geq n_0\}$

(B) Which of the following statements are **TRUE**? **Encircle** the **correct** choice(s). [4 marks]

A	2^{2n} is $O(2^n)$
B	$\log_2(n\sqrt{n})$ is $O(\log_2(n))$
C	$\sqrt{n}$ is $O(\log_2(n))$
D	If $f(n) = O(n^3)$ and $g(n) = O(n^3)$, then $f(n)$ is not in $O(g(n))$.

(C) Which of the following statements are **TRUE**? **Encircle** the **correct** choice(s). [4 marks]

A	Every DAG has exactly one source and one sink.
B	Given a connected, unweighted, undirected graph G. Let T be the BFS tree of G from root vertex R. If (u, v) is an edge in G that is **not** in T, then u and v are at the same number of hops from R.
C	In a connected, weighted, undirected graph G, the edge with maximum weight can never be in any minimum spanning tree of G.
D	Given a connected, weighted, directed graph G (V,E). All the edge weights in G are positive integers. Let P be the shortest path from vertex 'u' to 'v' in G. Suppose we create a new graph G′ by squaring the edge weights in G. P is not necessarily the shortest path from 'u' to 'v' in G′.
E	It's not necessary that *every* directed graph forms a DAG of its strongly connected components.
F	Cross edges do not exist in a DFS tree produced by running depth-first search on an undirected, weighted graph.

Q2(A) Consider the following code. [6 marks]

```
long RecFunc(int n) {
  int p;
  if (n==1) return n;
  else {
    RecFunc(n/2);
    p = n/16;
    return dosomething(p); }
}
```

```
long dosomething(int max) {
  int i; long sum=0;
  for (i=0; i<max; i++) {
      sum += i; }
  return sum;
}
```

Draw the recurrence tree of **RecFunc** and show the time spent at each level of the tree.

See the answer in **solution-Q2A-midterm-Fall-2019.jpg**
For values of n<16, dosomething() incurs a constant cost since body of for-loop is not executed. However, marks were not deducted if you did not show that in your answer.

Total time complexity of Recfunc()

Q2(B) You are given an unweighted, **acyclic directed** graph G and you run Depth-First-Search (DFS) on G starting from a source vertex and record the pre and post numbers. **Suppose there is a directed edge (u,v) from u to v in graph G.** Which of the following statements are **TRUE**. **Encircle** the **correct** choice(s). [4 marks]

A	The pre number of u > the pre number of v.
B	The pre number of u < the pre number of v.
C	The post number of u > the post number of v.
D	The post number of u < the post number of v.
E	pre(v) < pre(u) < post(u) < post(v) i.e. $[_{pre(v)}\ [_{pre(u)}\ \ post(u)]\ post(v)]$
F	There is no fixed relationship between the **post** numbers of u and v. It depends on graph G.

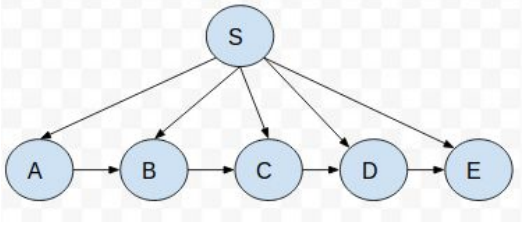
Q4. You are given an **unweighted, directed** graph G=(V,E), where vertices indicate courses and edges indicate a prerequisite relationship between courses. E.g. if there is an edge directed from x to y then x is a prerequisite for y. All prerequisites must be completed before a course can be taken. **The course-prerequisite graph does not contain any cycles. Assume that there is only one source vertex 'S' in G.** Also, assume that all courses are offered in each semester and there is no limit on the number of courses that can be taken per semester. Can any of the following methods be used to determine the **minimum number of semesters needed** to complete the entire coursework? **Encircle the correct choice(s).** [8 marks]

A	Run Breadth-First-Search (BFS) on G starting from the source vertex 'S' and place the nodes in G in levels based on hops from the source. The number of levels in the BFS tree is equal to the minimum number of semesters.
B	Find the shortest path between every pair of vertices in G. From all these shortest paths pick the one whose length has the largest value, let's call that path P and length of that path as Lp. The minimum number of semesters must be equal to Lp+1.
C	Find the longest simple paths in G starting from 'S' to every other vertex. From among these longest paths, pick the one with the most number of hops. Let's call that path P and length of P as Lp. The minimum number of semesters must be at least as large as Lp.
D	None of the above methods have an association with the minimum number of semesters.

If in the answer above, you have determined that one or more of the above methods **cannot** be used then provide counter-example for those methods.

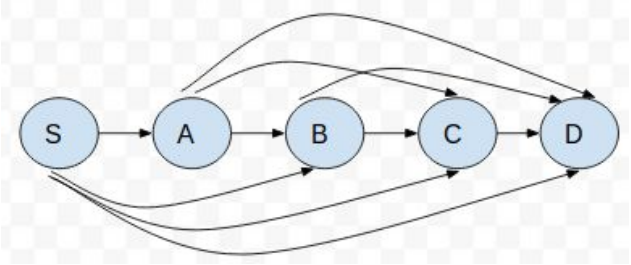
Counter example for A

BFS from 'S' has 2 levels. Min semesters = 6



Counter example for B

In a DAG there can't be a path between every pair of vertices, so only considering paths where they exist. Length of the longest among these shortest paths is 1. Min semesters = 5



Statement C is TRUE: Longest path has an association with the minimum number of semesters. The minimum number of semesters **cannot be less** than the longest path and hence must be ***at least*** as large as Lp.

Q5. Consider the following pseudo code. 'H' is a priority queue implemented via a heap and it supports the following operations.

Insert: Add a new element to the set.
Decrease-key: Accommodate the decrease in key value of a particular element.
Delete-min: Return the element with the smallest key, and remove it from the set.
Make-queue: Build a priority queue out of the given elements, with the given key values.

Input	Graph G = (V, E), directed or undirected; positive integer edge weights ≥1 $\{w_e : e \in E\}$; vertex s ∈ V

	for all u ∈ V :
	  dist(u) = ∞
	dist(s) = 1
	H = makequeue (V) (using dist-values as keys)
	while H is not empty:
	  u = deletemin(H)
	  for all edges (u, v) ∈ E:
	  if dist(v) > dist(u) + $\log_2(w_e(u, v))$:
	  dist(v) = dist(u) + $\log_2(w_e(u, v))$
	  decreasekey(H, v)
	for all u ∈ V except s:
	  dist(u) = log_2^{-1} (dist(u))
	dist[s]=0

Answer the following questions. [8 marks]

What is the above pseudocode computing? Justify your answer.
Using the property log a + log b + log c + log d + … = log (a * b * c * d * …)
It is computing paths from 's' to every other vertex that have the smallest **product** of edge weights and multiplying the product by 2 (because of dist(s) initialization to 1).

How many `deletemin` operations are executed in the pseudocode?
|V|

How many `decreasekey` operations are executed in the pseudocode?
O(|E|)

If auxiliary data structures discussed in class are not used with the priority queue, then what will be the time complexity of the `decreasekey` operation?

O(|V|) to find vertex v in the heap and O(log(|V|) to heapify after decreasing dist(v).

Common Formulae:

$$a + ar + ar^2 + ar^3 + \cdots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = a\left(\frac{1-r^n}{1-r}\right)$$

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}.$$

$$\log_a b = \frac{\log_c b}{\log_c a}$$

$$\log_a(xy) = \log_a(x) + \log_a(y)$$

$$\log_a n^b = b \cdot \log_a n$$

$$a^{\log(b)} = b^{\log(a)}$$