CSCI 6470 Quiz #4 Questions Answers

October 9, 2023 (12:40am-1:05pm EST)

Student Name ______________________________ **Student ID**__________________________

Rules. Violation will result in **zero credit** for the exam and possibly the final grade.
1. Closed book/note/electronics/neighborhood.
2. Surrender your cell phone to the podium before using the restroom.

There are 4 questions and 60 points in total. Good luck!

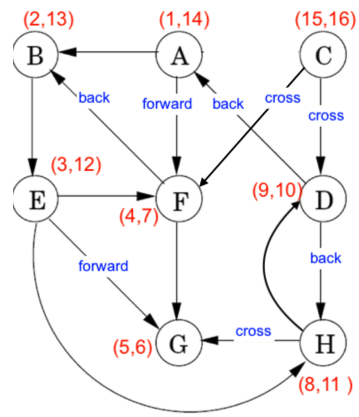
1. (**10 points**) This question concerns time complexity lower bound for algorithms and lower bound for problems. Choose correct statements from (A) – (E). (2 points each)
   (A) A lower bound of an algorithm is the amount of time required by the algorithms to solve worst case input instances. T
   (B) Some algorithms have lower bound $\Omega(n^2)$ and upper bound $O(n \log_2 n)$. F
   (C) For `Merge Sort` algorithm, because the worst case instances can be solved in $O(n \log_2 n)$ time, it allows us to conclude that the algorithm has lower bound $\Omega(n \log_2 n)$. F
   (D) If an algorithm solving some problem $\mathcal{P}$ has lower bound $\Omega(p(n))$, then the problem $\mathcal{P}$ has lower bound $\Omega(p(n))$. F, consider InsertionSort algorithm for problem **Sorting**
   (E) If an algorithm solving some problem $\mathcal{P}$ has upper bound $O(q(n))$, then the problem $\mathcal{P}$ has upper bound $O(q(n))$. T, since $\mathcal{P}$ can be solved by this algorithm in time $O(q(n))$

2. (**15 points**) We have learned the proof that problem **Sorting** has the lower bound $\Omega(n \log_2 n)$. Complete the following statements that have been used for the proof.
   (A) It is to prove that the worst case number of comparisons required by **Sorting** is $\Omega(n \log_2 n)$ on the "comparison-based model". In this comparison-based model, every comparison is done between elements only. $\boxed{2\,points}$
   (B) The number of different scenarios for an n-element list to any sorting algorithm is $n!$. $\boxed{2\,points}$
   (C) The statement in (B) implies that there are $n!$ number of leaves/paths in the binary decision tree that models the algorithm. $\boxed{total\ 3\,points}$
   (D) Two quantities x and y in the binary decision tree are related as $x \geq \log_2 y$, where x represents height/depth/longest path length and y represents number of leaves/paths. $\boxed{total\ 4\,points}$

(E) Quantity $\log_2(n!)$ is the number of comparisons on a longest, shortest [**choose one**] path of a deepest, shallowest [**choose one**] decision tree model. $\boxed{total\ 4\, points}$
longest path length is the worst case time, shallowest tree is the best algorithm

3. (**25 points**) Run the DFS search algorithm on the following directed graph, starting from vertex `A` and choose vertices in the alphabetical order when there are options. Answer the following questions.

(1) Based on the DFS, label a time stamp pair (`pre`, `post`) next to every vertex in the graph.
$\boxed{each\ pair\ \ 1\, points}$ if all correct, add 3 points



(2) List all edges according to their types on the DFS forest.
$\boxed{each\ edge, 1\, points}$

Tree edges: (A,B), (B,E), (E, F), (F, G), (E, H), (H, D)

Back edges: (F, B), (D, A), (D, H)
Forward edges: (E, G), (A, F)
Cross edges: (H,G), (C,D), (C, F)

4. (**10 points**) Consider the following typical pseudocode for the depth-first search (DFS). It can be slightly modified to solve a number of graph problems. Answer the following questions. **(Answers need to be succinct. Answers exceeding given spaces will not be graded.)**

```
function dfs-main(G);
  1. for every vertex x in G
  2.     visited(x) = false;
  3. for every vertex x in G
  4.     if visited(x) equals false
  5.         explore(G, x)
```

```
function explore(G, x);
  1. visited(x) = true;
  2. for every edge (x, y) in G
  3.     if visited(y) equals false
  4.         explore(G, y);
```

⇒ Grading this question should look at ideas only, not pseudocode or implementations.

(A) To determine if a given non-directed graph is connected, what is your idea of modification?
check if `explore` in line 5 of `dfs-main` is called only once. $\boxed{5\, points}$

(B) To determine if a given directed graph G contains a cycle, what is your idea of modification?
check if there is back edge formed. Not just a visited vertex is encountered again, since it could be a forward or cross edge as well. $\boxed{3\, points}$

(C) To determine if a given non-directed graph G contains a cycle, in addition to the suggestion in (B), what else? the edge probed is not a reversed tree edge. $\boxed{2\, points}$

[The following space will not be graded.]