READ THIS NOW!

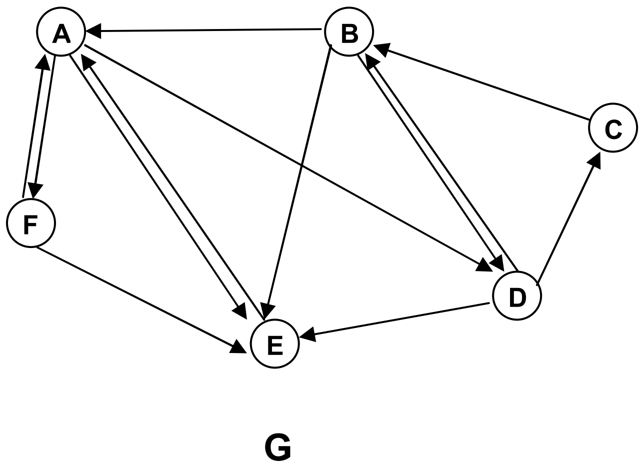
- Print your name in the space provided below.
- There are 6 short-answer questions, priced as marked. The maximum score is 100.
- Aside from the allowed one-page fact sheet, this is a closed-book, closed-notes examination.
- No laptops, calculators, cell phones or other electronic devices may be used during this examination.
- You may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.

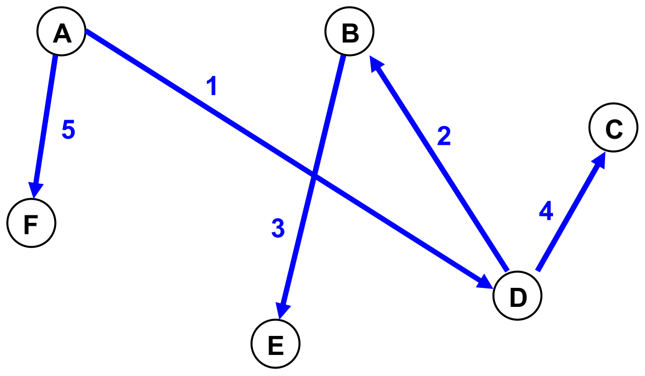
Name (Last, First) ______________________________
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

____
signed

1. [16 points] Draw the spanning tree for the graph **G** given below, obtained by applying a depth-first traversal starting at node **A**. Visit neighbors in alphabetical order, and number the edges in the order they are added to the tree.





2. [16 points] Recall that the access time for a hard drive is the sum of three separate measures of performance. An existing hard drive has an average rotational latency of 12 ms. Suppose the design is modified so that the rotational speed is increased by 25%.

No other characteristics of the original drive design are changed. Describe precisely what effect the change to the rotational speed will have on each of the three measures of drive performance, and why.

If the rotational speed is increased by 25% then the average latency will be decreased by 20% because it depends solely on the rotational speed:

$$L_{orig} = \frac{60000}{R}$$

$$L_{revised} = \frac{60000}{1.25R} = \frac{1}{1.25} L_{orig} = 0.8 L_{orig}$$

The average transfer time will also be decreased since it depends on the rotational speed and the number of sectors to be read or written.

The average seek time is unrelated to the rotational speed and will therefore not change.

3. Suppose that a database application repeatedly accesses records from a file on disk, containing millions of records. The application includes a buffer pool capable of storing up to 10 records at once.

   Suppose that the application accesses records according to the following repeated pattern:

   - access record 0
   - access a sequence of 10 random other, different records
   - repeat…

   a) [8 points] Which of the three replacement policies discussed in class would be the worst choice? Why?

      The FIFO policy will result in 0 buffer pool hits, since record 0 would be guaranteed to be flushed from the pool before the next access to it.

      LRU wouldn't do any better than FIFO in that regard.

      Potentially LFU could do better, but only if the buffer pool retained hit counts for records that have been flushed from the pool recently. In that case, record 0 could build up a large enough hit count to keep it in the pool.

   b) [10 points] Assuming the buffer pool uses the best of the three replacement strategies discussed in class, what percentage of record requests would you expect to result in a buffer pool hit? Why?

      Since the accesses to records other than record 0 are described as "different" and "random", there seems to be no hope that there will be any significant number of hits for those records.

      The only hope for hits for record 0 would be if LFU was used (as described above), and it that case we'd expect to get 1 hit in every 11 accesses, or about 9% of the time.

      (Clearly a buffer pool will not be of much use in this situation.)

4. Consider storing a large collection of strings that are words in the English language, using an alphabet trie versus using a minimum-height BST.

   a) [8 points] On average, which approach would you expect to require less space in memory, assuming the trie is implemented as efficiently as possible? Justify your conclusion.

      The trie will use less space to store the strings themselves since shared prefixes will be stored only once (or only stored implicitly).

      What about pointers? The trie will have one pointer for every character in a word (aside from shared prefixes where there's one pointer per shared character). The BST will have two pointers for each word, since there are two pointers to each node and there will be one node for each word.

      Pointers generally cost as much space as 4 characters.

      So, it's hard to state a specific rule in which one approach will take less space than the other, since it would seem to come down to how many characters are in shared prefixes, but it seems likely that the trie will usually take more space than the BST.

   b) [8 points] On average, which approach would you expect to have a lower average search cost? Justify your conclusion.

      Ideally the height of the BST will be log N, where N is the number of strings. So, the average search in the BST would require about log N comparisons of strings. <u>However</u>, the average cost of a string comparison would depend on the average length of the strings.

      Note that in the BST we would perform redundant comparisons if strings with shared prefixes are present.

      For a trie, the cost of finding any string is simply the length of the string.

      So, the cost of searching would be lower in the trie.

5. Recall the B-tree data structure.

a) [6 points] What is the relationship between the number of values stored in an internal node of the B-tree and the number of children that node has?

A B-tree node containing k data values always has k + 1 children, unless it's a leaf.

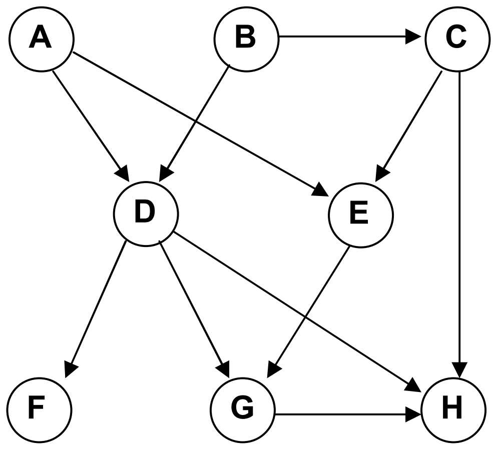
b) [12 points] B-trees and their relatives are generally used to index very large databases, <u>in which even the index cannot be entirely stored in memory at once</u>. Why is it better to use a B-tree, whose nodes store many values and have many children, rather than an on-disk AVL tree?

There are essentially two points to be made in the comparison:

A B-tree node will store many data values (or keys) while an AVL node will store only one. That means it will take more disk accesses for an average search if an AVL is used.

On the other hand, the B-tree node will be much larger and therefore require reading much more data from disk than the AVL node. BUT, we know that the cost of reading a full sector, or several sectors, from disk is hardly greater than the cost of reading just a few bytes, so reading a B-tree node will cost very little more than reading an AVL node.

6. [16 points] Find a topological ordering of the nodes in the directed acyclic graph given below. Recall that having an edge (x, y) implies that x must precede y in the ordering. For full credit, you must show some work that explains the logic of your analysis.



Fill in the node labels, according to your solution, below:

A B C E D F G H

done 1st … done last

Analysis: pick a node, say A, and perform a depth-first traversal, adding nodes all of whose pre-neighbors have already been added, untill we get stuck (mmediately)

A (both of A's neighbors have unmarked nodes at this point, so they can't be added yet)

Now pick an unmarked node and repeat:

B C E D F G H

But there are lots of don't cares, so there are many other solutions.