
***READ THIS NOW!***

- Print your name in the space provided below.
- There are 8 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)** **Solution**

printed

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

*signed*

1. Consider the notion of a *replacement policy*.

   a) [5 points] For which data structure is a replacement policy necessary?

   **For a buffer pool.**

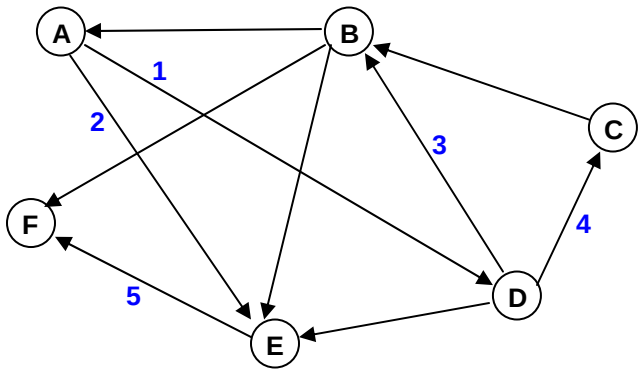
   b) [10 points] When is a replacement policy used? What are the goals of a replacement policy?

   **A replacement policy is necessary when all the slots in a buffer pool are full and another element must be added to the pool.**

   **The replacement policy is simply a rule by which we decide which element to replace. The most important goal is to avoid replacing an element that is likely to be needed again in the near future. A secondary goal is that the choice should be made cheaply.**

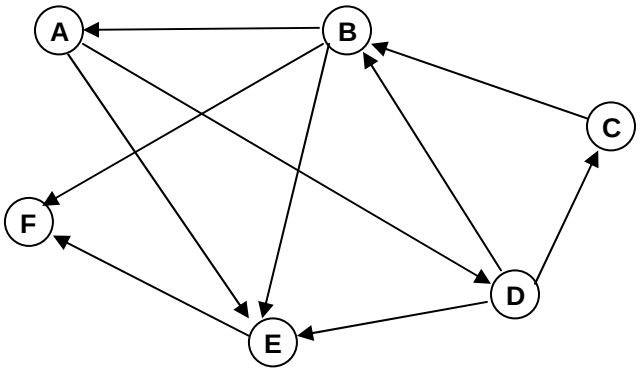
---

2. [15 points] Draw the spanning tree for the graph **G** given below, obtained by applying a breadth-first traversal starting at node **A**. Visit neighbors in alphabetical order, and number the edges in the order they are added to the tree.




**G**

3. [10 points] Draw the adjacency list (list of neighbors) representation for the graph below:




G

| Vertex | Neighbors |
|---|---|
| A | D, E |
| B | A, E, F |
| C | B |
| D | B, C, E |
| E | F |
| F | |

4. [10 points] Suppose you have large a collection of strings to store. Describe the important differences between the way they would be stored in a BST and the way they would be stored in an alphabet trie.

**In the BST, each complete string is stored in a node by itself; in an alphabet trie, each string is stored character-by-character, so that strings that share a common prefix are "twigs" of the same branch in the tree.**

**This is significant for a number of reasons.**

**First, in the BST, if there are N strings then the average search cost will be (at best) log N string comparisons; in an alphabet trie, the cost of find a particular string is determined by the length of the string, not by the number of strings in the tree. Moreover, in the BST comparisons involve entire strings whereas in the alphabet trie they involve single characters.**

**As a general rule, searching will be cheaper in the alphabet trie.**

**Second, the alphabet trie will have far more pointers than the BST (most NULL) and typically requires significantly more storage, even though common prefixes of strings are not stored multiple times as they are in a BST.**

5. Suppose that a hard drive design currently produces an average rotational latency of 8ms and an average seek time of 4ms.

   a) [8 points] What aspect(s) of the hard drive's implementation could be changed in order to reduce the average rotational latency?

   **Only the rotational speed of the platters is significant for the rotational latency.**

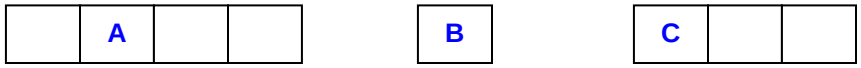
   b) [8 points] What aspect(s) of the hard drive's implementation could be changed in order to reduce the average seek time?

   **The seek time depends on the head start time and the track-to-track seek time and the number of tracks that must be crossed on an average seek. The average value of the latter is related to the number of tracks per surface.**

---

6. [10 points] Describe a specific scenario in which it would be logically necessary to use tombstones in a hash table implementation.

   **Suppose you insert three values, A, B and C, that collide in the same home slot and the table uses a probing strategy to find an alternate slot:**




   **Now suppose that B is deleted from the table. If the slot holding B is simply marked as "empty" then subsequent searches for C will fail, since they will end when the first empty slot is found.**

   **If, however the slot that formerly held B is marked as a tombstone, then searches for C will proceed past that slot and C will be found.**

   **The only alternatives (to avoid this) would be to either always search every slot in the table before concluding that a value is not present, or to "contract" probe sequences on a deletion (e.g., move C into B's slot).**

   **The former is too expensive since now failed searches would cost N comparisons. The latter is also too expensive, but is also more difficult to get right than this example shows. (For instance, how do you KNOW that C wound up where it did by probing there?)**

7. [10 points] Suppose that a hash table has 13 slots and uses pure quadratic probing to resolve collisions. If a key value hashes to the home slot 7, what slots would potentially be examined in the resulting probe?

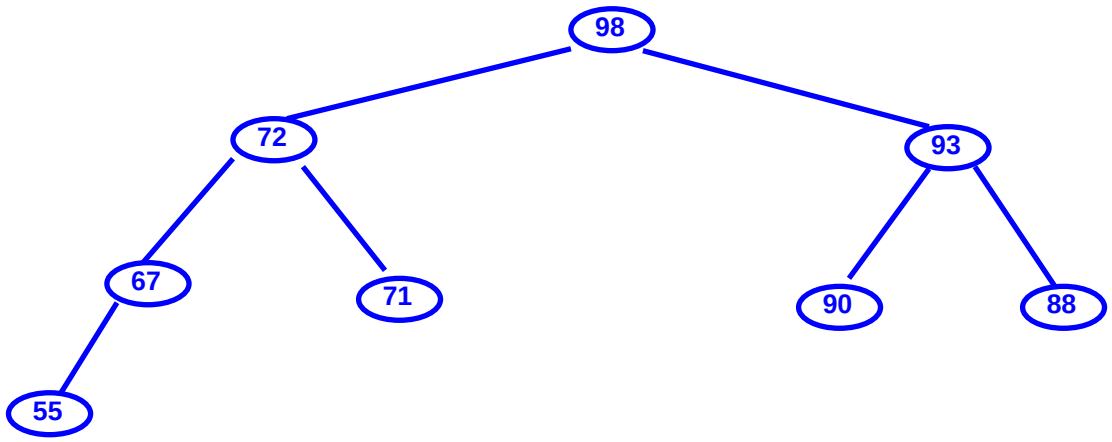
**You'd start with slot 7, of course. Then you'd look at slots (7 + i^2)mod 13 until you began repeating the slot numbers. Work it out:**

**7, 8, 11, 3, 10, 6, 4**

---

8. a) [8 points] Draw a tree representation of the following max-heap:

| 98 | 72 | 93 | 67 | 71 | 90 | 88 | 55 |
|---|---|---|---|---|---|---|---|




b) [6 points] A heap is a *complete* binary tree. Is it possible to store a binary tree in an array, using the same basic approach as for a heap, if the binary tree is not complete? If no, why not? If yes, is there anything "extra" you would have to do?

**The positions of the values are determined by formulas for where the children go: if a value is at index k, then its left child is at 2k+1 and its right child is at 2k+2.**

**That would work for any binary tree.**

**The problem is that if the tree isn't complete then there may be "gaps" due to missing nodes in a level. For example, if we removed the value 90 from the tree above, it's no longer a complete binary tree, and there would be an empty cell in the array where the given array shows the value 90.**

**So, we'd have to include some way to mark a cell as being "empty"; that's not hard, but it's an extra data value per cell.**