Final Review
Announcements

• PA4 is due tonight at 11pm
• PA3 re-grade requests due by Monday
• Study guide and practice final posted on course website (website was upgraded to be mobile friendly).
• Review session tomorrow from 11am to 1pm in YORK
• More additions *may* be made to the practice final before Monday
• Section A: If you have multiple finals (three or more) on Friday and need to take the exam on Wed, send me an email (with the necessary proof).
• Section B: If you have multiple finals (three or more) on Wed and need to take the exam on Friday, send me an email (with the necessary proof)
Final Exam...

• 4 Parts
  • Part 1: Basic knowledge of data structures and C++
    – 20% to 30% of final score
    – Multiple choice
  • Part 2: Application, Comparison and Implementation of the data structures
    – 20% to 30% of final score
    – Short answers
  • Part 3: Simulating algorithms and run time analysis
    – 20% to 30% of final score
    – Short answers
  • Part 4: C++ and programming assignments
Part 2: B-trees

(c) Which of the following are legal 2,3 trees (B tree of order 3)? For a tree that is not a valid 2,3 tree, state a reason why.

![B-tree diagrams](https://www.cs.usfca.edu/~galles/visualization/BTree.html)

Insert the value 42 into the following BTree.
(a) Draw the final forest of up-trees that results from the following sequence of operations using union-by-size. Break ties by keeping the first argument as the root. Union(0, 2), Union(3, 4), Union(9, 7), Union(9, 3), Union(6, 8), Union(6, 0), Union(12, 6), Union(1, 11), Union(9, 6)
(b) Draw the new forest of up-trees that results from doing a Find(4) with path compression on your forest of up-trees from (a).
Ternary Tries

(a) Consider the following ternary search tree. Nodes with double circles have their end bits set to true. Circle all of the words from the list on the right that are in the tree and write in any words that are missing. At the end you should have a complete list of all words found in the tree, and only those words.

(b) Does the height of this tree depend on the order in which the keys have been inserted?

(c) Briefly explain why you would prefer to use a ternary search tree rather than a binary search tree to implement `getAllValidWords()` in PA4.

```
<table>
<thead>
<tr>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>i</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>s</td>
</tr>
</tbody>
</table>

| t     |
```

`air, all, a, and, as`
1. **Graphs.** Given the following graph, run Dijkstra’s algorithm on it, with source node A. Alongside the graph is the data structure for each node that you will modify, see below.

<table>
<thead>
<tr>
<th>Node</th>
<th>dist</th>
<th>prev</th>
<th>done</th>
<th>Adjacencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>f</td>
<td>((B,5),(D,10),(E,6))</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>-1</td>
<td>f</td>
<td>((D,3))</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>-1</td>
<td>f</td>
<td>((D,0))</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>-1</td>
<td>f</td>
<td>()</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>-1</td>
<td>f</td>
<td>((C,3))</td>
</tr>
</tbody>
</table>

   (initial priority queue)

<table>
<thead>
<tr>
<th>Node</th>
<th>dist</th>
<th>prev</th>
<th>done</th>
<th>Adjacencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>f</td>
<td>((B,5),(D,10),(E,6))</td>
</tr>
</tbody>
</table>

   (priority queue after exploring A)
3. Run Time Analysis

(a) Consider the following algorithm operating of a graph $G(V, E)$ and a source vertex $s$

i. Initially, set the visited field of all vertices in $V$ as FALSE

ii. Start at source vertex, $s$. Set the visited field of $s$ as TRUE, set counter to 1

iii. Enqueue $s$ into a queue

iv. While the queue is not empty or counter less than or equal to $|V|$
   
   A. Dequeue the vertex $v$ from the head of the queue
   
   B. For each of $v'$s neighbor vertex, $w$ (with visited FALSE):
      
      • Mark the visited field of $w$ as TRUE
      • Increment counter by 1
      • Enqueue $w$ into the queue

i. Identify the above graph algorithm.

ii. Assume the graph is implemented as an adjacency list. Write the Big O running time of each step of the algorithm (against each line of above) and derive the tightest overall Big O bound.
Data structure Comparison

Is operation K faster in data structure X or data structure Y?
Data structure Comparison

Of the following pairs of data structures, which of the pair is the better choice for... (circle 1 in each set)

• Inserting a list of *sorted* elements:
  Randomized search tree / Binary search tree / They are about equal
• Ease of implementation (assume it’s not built in):
  Skip list / Red-Black tree (RBT) / about equal
• In-order traversal of elements:
  Hashtable / Binary search tree / They are about equal
• Smallest average-case (Big-O) time to find an element:
  Hashtable / AVL tree / They are about equal
• Fastest *actual* time to find an element from secondary storage (NOT big-O):
  RBT / AVL tree / B-trees/They are all about equal
• Requires less space: Multi-way trie / Ternary tree / They are about equal
• Congratulations on pulling through – you are a survivor!
• Hope you go on to do wonderful things with your knowledge of advanced data structures.

Good luck with PA4, the final and life in general 😊