COMPUTER SCIENCE 112 - FALL 2010
FINAL EXAM

NAME: ________________________________

CIRCLE your recitation	Wed 10:35 Ehtesam	Wed 12:15 Mustafa	Tue 6:55 Tin	Thu 6:55 Yixin

---

- Be sure your exam has 7 questions. DO NOT TEAR OFF THE SCRATCH PAGES OR REMOVE THE STAPLE.
- Be sure to fill your name and circle your recitation time above, and your name in all subsequent pages where indicated.
- This is a CLOSED TEXT and CLOSED NOTES exam. You MAY NOT use calculators, cellphones, or any other electronic device during the exam.

Do not write below this line

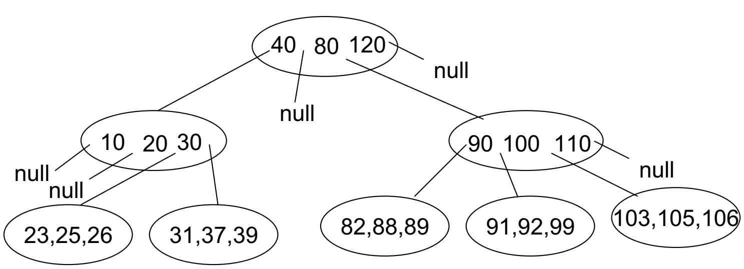
Problem	Value	Score
1 M-ary Tree	25	______
2 Huffman Coding	25	______
3 Graph Connected Components	25	______
4 Dijkstra’s Algorithm	25	______
5 Merging Heaps	25	______
6 Sorting	25	______
7 AVL Tree	25	______
TOTAL	175	______

1. **M-ary Tree (25 pts, 8+9+8)**

a) An m-ary tree is a tree in which every node has at most m children. Consider a special m-ary tree in which a node either has exactly m children or no children. What is the minimum number of nodes in a special m-ary tree of height h? (A single node tree has height 0). Show your work, and show how your answer works out correctly on an example special tree with $h = 3$ and $m = 3$.

b) Suppose each node of the m-ary tree stores $m - 1$ keys, $k_1, k_2, \ldots k_{m-1}$ in **ascending order**. Then, all keys k such that $k < k_1$ is stored in the first subtree, all keys k such that $k_1 < k < k_2$ is stored in the second subtree, and so on. All keys k such that $k > k_{m-1}$ is stored in the m-th subtree.

Here's an example 4-ary tree, with 3 keys in each node:



A search for a target key, k_t, proceeds as follows (recursive):
0. If root is null, return false. Otherwise, proceed to step 1.
1. Do a sequential search of k_t against the keys $k_i, 1 \leq i \leq m - 1$, at the root, either finding a match, or stopping because $k_t > k_{m-1}$, or stopping because $k_t < k_i$ for some i. Proceed to step 2.
2. If a match is found, return true. Otherwise, proceed to step 3.
3. If $k_t > k_{m-1}$, then recursively search the m-th subtree and return the result. Otherwise, proceed to step 4.
4. Recursively search the i-th subtree and return the result.

If the height of an m-ary tree is h, how many key-to-key comparisons (exact number, **not** big O) would it take in the worst case to successfully search for a target key? Give your answer in terms of h. Show your work, and show how it would work out correctly for the example 4-ary tree given above.

c) Assuming a full m-ary tree, i.e. every level has the maximum possible number of nodes, compute h in terms of n, the total number of nodes.

2. **Huffman coding (25 pts, 17+8)**

Given the following set of character-probability pairs:

```
(A,0.2), (B,0.1), (C,0.3), (D,0.15), (E,0.05), (F,0.2)
```

(The probability associated with each character is the probability of the character occurring in the text to be compressed.)

(a) Build a Huffman tree for this character set. Fill in the following table to show the leaves queue (S) and the trees queue (T) at the end of each step.

```
Step                         Leaves Queue (S)                        Trees Queue (T)
--------------------------------------------------------------------------------
  1                                                                      Empty
```

b) Assume that enqueue, dequeue, peek, creating a leaf node, creating a new tree out of two subtrees, and picking the minimum of two probabilities all take unit time. Ignore the time for all other operations. How many units of time did it take in all to build your tree? Show your work.

3. **Graph Connected Components (25 pts)**

You are given an undirected, unweighted graph that may be disconnected i.e. some vertices may not be reachable from other vertices. Every group of mutually reachable vertices forms an island, called a *connected component*. There is no path between vertices in different connected components. If a graph is not disconnected, then its vertices are in a single connected component, which is the entire graph itself.

Implement a method using depth-first search that will mark each vertex with the connected component that it belongs to. If there is a single connected component, then all vertices will be numbered 0 (zero). If there are two islands, then vertices in one island will all be numbered 0, and those in the other will all be numbered 1. And so forth.

Write your implementation in the `components` method in the following graph class. Assume that when that method is called, the `adjLists` structure is already populated. You can write helper methods as necessary– you must fully implement all helpers you use.

```
class Neighbor {
   int vnum;
   Neighbor next;
}
public class Graph { // undirected
   Neighbor[] adjLists;
   ...
   // returns an array with connected component numbers for all vertices
   // i.e. returned[i] is the compnonent number for vertex i
   public int[] components() {
      /* COMPLETE THIS METHOD */
```

4. **Dijkstra's Algorithm (25 pts, 15+10)**

Here is a weighted directed graph, represented in adjacency linked lists form.

```
A -> B,5 -> C,3

B -> D,2

C -> D,2 -> E,4 -> B,1

D -> empty

E -> empty
```

a) Dijkstra's shortest paths algorithm is executed on this graph, starting at A. Trace the execution of the algorithm as follows: at the end of every step, show the distance array and the fringe. Initially, the distances are all infinity, and the fringe is empty. The fringe is stored in a heap, in which updates can be made (apart from delete-top and insert). To show the fringe/heap, draw the heap diagram as a binary tree, with vertices at the nodes. Every time there is a change to the heap, show the number of item-to-item comparisons needed to make that change.

b) The worst case running time of Dijsktra's algorithm is $O((n+e)\log n)$, on a graph with n vertices and e edges, provided the fringe is stored in a heap with $O(\log n)$ insert, delete-top, and update operations, and the graph is stored in adjacency linked lists. Assume that the graph is stored in an adjacency matrix instead, without changing any other data structure. State clearly which step of the algorithm would be impacted by this change, how it would be impacted, and show how you arrive at the new running time.

5. **Merging Heaps (25 pts, 6+7+4+8)**

Suppose that we have k heaps, with at most n elements in each. We wish to combine these into a single heap. Each of the following sub-parts describes one approach to merging the heaps, and you are asked to derive the big O running time. To get any credit, **you must provide an explanation** for your answer.

a) We create a new, empty heap H. For each of the k heaps h_i, we repeatedly remove the highest priority element and insert it into H, until h_i is empty. What is the worst-case big O running time?

b) We create a new, empty linked list L. For each of the k heaps h_i, we repeatedly remove the highest priority element and insert it onto the beginning of L, until h_i is empty. We then transfer the elements of L into an array, and **heapify** the array. What is the worst-case big O running time?

c) We create a new, empty linked list L. For each of the k heaps h_i, we iterate over its internal array and insert each element onto the beginning of L. We then transfer the elements of L into an array, and heapify the array. What is the worst-case running time?

d) We group the k heaps into $k/2$ pairs, and apply the algorithm from part (c) on each pair, leaving $k/2$ heaps. We then repeat this process until we are left with a single heap. What is the worst-case running time?

6. **Sorting (25 pts; 8+8+9)**

a) A stable sorting algorithm is one which preserves the order of duplicate key entries when sorted. In other words, if entries e_1 and e_2 have the same key k, and e_1 appears before e_2 in the input, then e_1 will appear before e_2 in the sorted output.

Is quicksort stable? Prove your answer, detailing every step that you take to arrive at the conclusion.

b) Suppose you use radix sort to sort countries according to ascending order of population. The populations are read in as text, and digits are extracted using the `Character.digit(char ch, int radix)` method. What is the worst case number of unit time operations that would be performed to sort this list? State the unit time operations, and show the derivation of your result. (You may want to use the fact that there are about 190 countries, and China has the largest population, of around 1.3 billion.)

c) For the sorting problem described in (b), describe the fastest (in real time, not big O) possible implemenation you can think of. (Side note: Vatican City has a population of 800). Detail the data structure and the process. You can use any of the sorting algorithms you have learned in class (except radix sort, which you have already analyzed in part b).

7. **AVL Tree (25 pts;8+8+9)**

You have been asked to write a program to count the number of times each word occurs in a text file, and print them grouped according to word length. Below is a sample text file on the left, and the expected output of your program on the right:

```
This is the first line,
and this line is the ultimate line,
```

```
is: 2
the: 2
and: 1
this: 2
line: 3
first: 1
ultimate: 1
```

Note that words are counted ignoring case, and are printed in order of word length. In any group of words of the same length, the printout can be in any order (e.g. *the* and *and* are both printed before words of greater length, but they need not be in any specific relative sequence.)

<u>You choose to store the words in an AVL tree.</u> Each node in the AVL tree has a word (with its count), and words (alphabetical) are the basis of the AVL ordering.

a) Show the <u>final</u> AVL tree after *all* the words from the sample input file have been inserted. (Remember that words will be inserted one at a time as they are read in, but you only need to show the final tree, not all the intermediate trees). Make sure you show balance factors at all the nodes. There is *only one correct AVL tree answer*, following the AVL insertion algorithm you have learned in class.

b) If there are k distinct words, and n words in all (in the sample input file, $k = 7$ and $n = 12$), how much time will it take in the worst case (big O) to store all the words with counts? Assume that all word comparisons take constant time, and ignore the time taken to read a word from input.

c) What would be the worst case time (big O) to print the output, if the maximum word length is m? Describe the process, and clearly state the basic operations (ignore the time to actually write to disk). Clearly describe any additional data structures you may use to get the job done. Show your work in summing up the basic operations toward the final big O answer. (Assume that determining the length of a word takes constant time.)

SCRATCH SPACE

SCRATCH SPACE

SCRATCH SPACE