**CS 3333 Final Exam (Review)**

1. (Multiple choices) [10 points] Consider any given undirected unweighted graph G(V, E). The diameter of G is defined as the maximum shortest distance between any pair of vertices u and v. Which of the following will find the diameter of G?
    1) Call Dijkstra's algorithm starting with any given vertex u to find a shortest path to every other vertex. The longest path from u to some vertex is the diameter of G.
    2) Call Prim's algorithm starting with any given vertex u to find a minimum spanning tree. The longest path from u to some vertex is the diameter of G.
    3) Call Kruskal's algorithm to find a minimum spanning tree. Pick a root u and the longest path from u to some vertex is the diameter of G.
    4) None of the above.

2. (Multiple choices) [10 points] Given a set of points $(x_i, y_i), 1 \leq i \leq n$, we need to find two points with the shortest distance. Obviously, there is an $O(n^2)$ time algorithm for solving this problem. Which of the following is true?
    1) With divide and conquer, we can solve this problem in O(n log n) time.
    2) With a greedy algorithm, we can solve this problem in O(n log n) time.
    3) With dynamic programming, we can solve this problem in O(n log n) time.
    4) None of the above.

3. (Multiple choices) [10 points] Given a list of (possible negative) integers $a_1, a_2, \ldots, a_n$, find the maximum value of $S(i,j) = a_i + a_{i+1} + \cdots + a_j, 1 \leq i \leq j \leq n$. Note: For convenience, the maximum value is 0, if all integers are negative. Which of the following is the best possible for solving this problem?
    1) A 3-nested-loop algorithm with $O(n^3)$ time complexity
    2) A 2-nested-loop algorithm with $O(n^2)$ time complexity
    3) A divide and conquer algorithm with $O(n\ log\ n)$ time complexity
    4) A simple linear scan algorithm with $O(n)$ time complexity.

2. **(13 points) Basic Knowledge 1**
    1. What is the big-Oh run time of Merge Sort on n items?
    2. What is the big-Oh run time of Heap Sort on n items?
    3. What is the big-Omega lower bound for comparison based sorting of n items?
    4. What is the worst case big-Oh run time to insert 1 item into an AVL-tree that contains n items?
    5. What is the worst case big-Oh run time of binary search on an n item sorted list?
    6. What is the worst case big-Oh run time to insert 1 item into a min-heap?
    7. What is the worst case big-Oh run time to remove the minimum value from a min-heap?
    8. What is the run time of breadth-first search (in terms of |V| and |E|)?
    9. What is the run time of Dijkstra's algorithm (in terms of |V| and |E|) assuming a min-heap based implementation?
    10. What is the run time of Dijkstra's algorithm (in terms of |V| and |E|) assuming a Fibonacci-heap based implementation?
    11. What is the run time of the Bellman-Ford single-source shortest path algorithm (in terms of |V| and |E|)?
    12. What is the run time of the Floyd-Warshall "all-pairs" shortest path algorithm (The one that uses dynamic programming)?
    13. What is the run time of the Edmonds-Karp maximum flow algorithm?

3. **(12 points) Basic Knowledge 2.** Circle all of the statements below that are true:
    a. log(n) = O(n)
    b. n = O(log n)
    c. log^2 (n) = O(n)
    d. n = O( log^2 n)
    e. log(n) = Ω(n)
    f. n = Ω(log n)
    g. 5n^3 + 7n + 13 = O(n^5)
    h. 5n^3 + 7n + 13 = Ω(n^5)
    i. 5n^3 + 7n + 13 = Ω(n)
    j. log(n!) = Θ( nlog(n) )
    k. n^(1/2) = O( log n)
    l. 2^n = Ω( n! )

4. (20 points) Analyze the big-oh run time for each function with respect to parameter n:

```
void A(int n)
{
	if (n < 10)
		time++;
	else
	{
		for (int i = 0; i < n; i++)
			time++;
		A(n - 2);
	}
}


void B(int n)
{
	if (n < 10)
		time++;
	else
	{
		for (int i = 0; i < 8; i++)
			B(n / 2);
		for (int i = 0; i < n*n; i++)
			time++;
	}
}


void C(int n)
{
	if (n < 10)
		time++;
	else
	{
		C(n / 5);
		C(n / 10);
		for (int i = 0; i < n; i++)
			time++;
	}
}

void D(int n)
{
	time = 0;
	for (i = 0; i < n; i++)
		for (j = 0; j < i * i; j++)
			for (k = 0; k < j*j; k++)
				time++;
}

void E(int n)
{
	minHeap H;
	AVLtree T;
	for (int i = 0; i < n; i++)
	{
		H.insert(i);
		T.insert(i);
	}

	for (int i = 0; i < n; i++)
		H.extractMin();
}
```

5. **(10 points) AVL Trees**: Show the result of inserting 2,1,4,5,9,3,6,7 into an initially empty AVL tree.

6. **(10 points) Heaps**: Show the result of inserting 2,1,4,5,9,3,6,7 into an initially empty min-heap. Then, perform 3 extract-min operations and show the final resultant heap.

7. **(10 points) Hash Tables**: Show the result of hashing the following keys to a size 10 hash table using the hash function h(x) = x mod 10 and quadratic probing to resolve collisions:

4371, 1323, 6173, 4199, 4344, 9679, 1989

**Coding: Binary Trees**

For the following questions, assume binary trees consist of nodes from the following class:

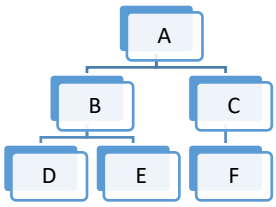
```
class node{
public:
        int data;
        node * left;
        node * right;
};
```

8. (15 points) Write a function such that given a root of tree and an integer sum, return true if there is a path from the root down to a leaf, such that adding up all the values along the path equals the given sum.

```
bool hasPathSum(node* p, int sum) {
```

9. (15 points) Write a method 'levelOrderTraversal(node * r)' which prints the items of a binary tree in a "level order". That is, the first item printed is the root, the next items printed are the children of the root, the next items printed are the grandchildren of the root, etc. You may freely use STL containers such as the vector, stack, or queue data types to solve the problem. Analyze the run time of your solution (more points for a faster solution).

The following tree, for example, would be printed in the following order: A B C D E F



**10. (15 points) Coding: Trie Coding**

Write a function 'int numEntries(Node * p)' that computes and returns the number of strings stored in the Trie rooted at node p. Use the provided Node class.

```
class Node
{
public:
        //Variable denoting if this node is the end
        //character of a string in the Trie
        bool marked;

        Node * children[256];

        Node()
        {
                marked = false;

                for (int i = 0; i < 256; i++)
                        children[i] = nullptr;
        }
};
```

11. **Skyrim Loot Problem**:

You are playing Skyrim and your character can carry a total weight of W in their inventory, where W is some positive integer. You enter the bandits' lair where you find n pieces of loot. The first piece of loot weighs w_1 pounds and has a value of v_1 gold pieces, where w_1 and v_1 are positive integers. The second item in the lair has weight w_2 and value v_2, etc. Your goal is to select a subset of the n items to take such that the total value is as large as possible, but the sum of the weight is less or equal to your weight capacity W. In other words:

Input:

- a positive integer W
- n pairs of positive integers denoting the weight and value of each object: (w_1,v_1), (w_2,v_2),…,(w_n,v_n)

Output: The maximum possible total value you can obtain by selecting a subset of the n items while keeping the total weight of the selected items to be at most W.

Design an algorithm to solve the Skyrim loot problem. Analyze the run time of your solution in terms of W and n.

12. **Zippers:** Given three strings, you are to determine whether the third string can be formed by combining the characters in the first two strings. The first two strings can be mixed arbitrarily, but the characters from each must stay in their original order in the third string.

For example, consider forming "tcarete" from "cat" and "tree":
String A**: c a t**
String B: **t r e e**
String C: **t c a r e t e**

As you can see, we can form string C by selecting the first character of "tree", followed by the first 2 characters of "cat", followed by the second and third characters of "tree", followed by the last character of "cat" and "tree" respectively.

As a second example, consider forming "catrtee" from "cat" and "tree":
String A: **c a t**
String B: **t r e e**
String C: **c a t r t e e**

The answer for this input is also 'yes'

Finally, notice that it is impossible to form "cttaree" from "cat" and "tree", meaning the answer for this input would be 'no'.

Zipper Problem
Input

- String A = *a1, a2, a3, ... ,* $a_n$
- String B = *b1, b2, b3, ...,* $b_m$
- String C = *c1, c2, c3, ...,* $c_{m+n}$

Output: Output *yes* if A and B can be combined (zippered) into string C.
Output *no* if A and B cannot be combined to form C.

Design an O(nm) time algorithm to solve this problem.

13. **Making Optimal Change**

Consider the problem of making change for *n* cents using as few coins as possible.

a. Describe a greedy algorithm to solve this problem using only quarters, dimes, nickels, and pennies. Prove that your algorithm will give the optimal (smallest) number of coins.

b. Give a set of coins for which the greedy algorithm will not yield the optimal solution. Your set should include a penny so that a solution always exists.

c. Give an *O(nk)*-time algorithm to compute the optimal way to make change for n cents from any set of *k*-different coin denominations, assuming one of the denominations is a penny.

**14. Santa needs some help**

For this problem imagine you are Santa and you want to try to perfectly pack your sleigh with presents. That is, you want to pack up the sleigh with some of the presents from the workshop so that the total weight of the presents on the sleigh is exactly equal to the capacity of the sleigh, no more, no less. Formally, the problem is as follows:

Perfect Christmas Packing Problem:

Input

- *n* Christmas presents, each with weight *w1, w2, ..., wn*
- A positive integer *k* denoting the maximum weight Santa's sleigh can hold.

Output: Output *yes* if there is a subset of the *n* presents whose total weight sums to exactly *k*. Output *no* if there is no such subset.

Design a O(nk) run-time algorithm to solve this problem.

**15. Rocky Mountain High!**

Suppose you are given an n x m matrix of positive integers denoting an elevation map of a park in the Rocky Mountains. Your goal is to design a path from the left side of the map (column 1) and the right side of the map (column m). Each step in the path must proceed by going either straight right to the next column, or diagonally up and right, or diagonally down and right (no going straight up or backwards). The cost of each step in the path is the absolute value of the difference in elevations between the two cells being traversed. The total cost of a path is the sum of each of step in the path. The goal is to compute a minimum possible cost path (hikers want a path that is as level as possible) connecting the left side of the map to the right side. Design an algorithm to compute the cost of the shortest possible path in O(nm) time.

**16. 4-Color is NP-complete**

The 3-Color problem is a classic NP-complete problem in which you are given a graph and must output whether or not the graph is "3 colorable". A graph is k colorable if it is possible to assign each vertex one of k distinct colors such that no two adjacent vertices are the same color. The 4-color problem asks whether a given graph is 4-colorable.

i) Show that the 4-color problem is also NP-complete.
ii) Show that the 100-color problem is also NP-complete.