3013 - Algorithms **Name:** _____
Spring 2021
Final Exam Study Guide
April 28, 2021

---

1. What is an algorithm? Define. Explain. Use examples.

---

2. (a) List the categories of algorithms.
   (b) Give 2 example algorithms that fit into each category.

---

3. List the typical complexities from fastest to slowest.

---

4. (a) Draw the binary min heap that results from inserting: 77, 22, 9, 68, 16, 34, 13, 8 in that order into an initially empty binary min heap. You do not need to show the array representation of the heap.
   (b) Draw the binary min heap that results from doing 2 deletemins on the heap you created in part a).

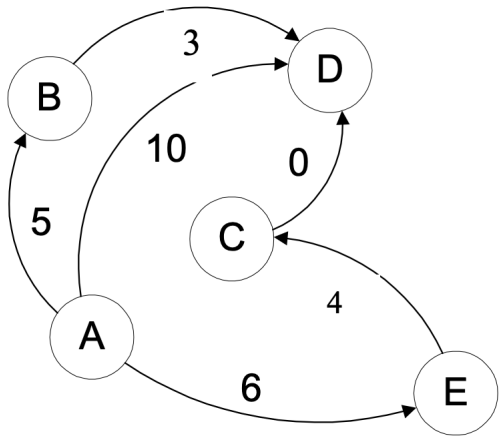
---

5. What is the main advantage that open addressing hashing techniques have over separate chaining?

6. You could use an AVL tree to do a sort. Describe how you would do this. What is the worst-case running time for your sort?

7. Kruskal's minimum spanning tree algorithm uses a heap to quickly find the lowest cost edge not already chosen. What would be the running time of the algorithm if instead of using a heap, the algorithm used a simple unsorted array to store the edges?

8.



(a) Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.

(b) Step through Dijkstra's Algorithm to calculate the single source shortest path from A to every other vertex. You only need to show your final table, but you should show your steps in the table below for partial credit. Show your steps by crossing through values that are replaced by a new value. Note that the next question asks you to recall what order vertices were declared known.

Vertex	Known	Distance	Path
A			
B			
C			
D			
E			

(c) In what order would Dijkstra's algorithm mark each node as known?

(d) What is the shortest (weighted) path from A to D?

(e) What is the length (weighted cost) of the shortest path you listed in part (e)?

---

9. Draw the contents of the hash table in the boxes below given the following conditions:

- The size of the hash table is 12.
- Open addressing and quadratic probing is used to resolve collisions.
- The hash function used is $H(k) = k \; mod \; 12$

What values will be in the hash table after the following sequence of insertions? Draw the values in the boxes below, and show your work for partial credit. 33, 10, 9, 13, 12, 45

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

10. Given an initially empty hash table with capacity 11 and hash function $H(x) = x \; \% \; 11$. Keys 0, 1, 8, 9 are already in the table. Insert keys 52, 44, 56, 53, 61, 64 (in that order).

Linear-Probing

0	0
1	1
2	
3	
4	
5	
6	
7	
8	8
9	9
10	

Quadratic-Probing

0	0
1	1
2	
3	
4	
5	
6	
7	
8	8
9	9
10	

Double-Hashing
$H'(x) = 7–(x\%7)$

0	0
1	1
2	
3	
4	
5	
6	
7	
8	8
9	9
10	

---

11. Consider these algorithms: Bubble sort, Heap sort, Insertion sort, Merge sort, Quick sort, and Selection sort. Read the situation below, and decide which algorithm would have the best performance given the situation. Briefly justify each answer.

    Helpful Link: https://afteracademy.com/blog/comparison-of-sorting-algorithms

    (a) The input array happens to be already sorted.

    (b) The input array is in random order, and average case time is most important.

    (c) Suppose that exchanges of the items in the array are very, very expensive. In other words, which method does the fewest "swaps" or "moves" of the elements of the array?

    (d) The worst case running time is the most important such as for a real-time system.

    (e) The input array consists of bits (0s and 1s) and the sort must be both stable and in-place.

12. Spanning Trees.

    Helpful Link:
    Look at the analysis section here: https://www.baeldung.com/cs/kruskals-vs-prims-algorithm

    (a) What is a spanning tree?

    (b) What are they used for?

    (c) What algorithms exist to find spanning trees within a graph? Do they they have the same complexities? Does each algorithm work better for certain types of graphs?

---

13. Given the following list of values:

    $[1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9]$

    What algorithm paradigm would you use to find the number of occurrences of a given number in the fastest way possible. Use any method you want. You may even combine paradigms if you want.

    Example:
    Target: 7

Output: 4

```
int arr[] = {1,1,2,2,2,3,3,4,4,4,4,4,5,5,5,6,6,7,7,7,7,8,8,8,8,8,8,8,9,9,9}
int x = find_occurences(arr,7)
cout << x << endl;
// writes out 4
```

Note:

Make sure you explain in detail what you are doing. Remember you are describing your algorithm for solving the problem. Anyone reading your answer should be able to re-create your steps and solve the problem themselves.

---

14. The majority of sorting algorithms perform in a couple of similar categories which really do not effect us when sorting small chunks of data. But it is good to know about the different approaches to sorting. So for this question I want you to explain to me:

    (a) (10 points) What categories of sorting algorithms are there?

    (b) (5 points) That majority of sorting algorithms can be categorized in with run times of O(_____) or O(_____)

    (c) (15 points) What is the best performance you can expect from a sorting algorithm? Give examples and be thorough.

15. Again, here is another "topic" we didn't discuss in class. Why? We tend to focus on problems that we can solve using algorithms that we can implement. These are the majority of problems: solvable in a reasonable amount of time with a reasonable algorithm. Reasonable = "P" or "polynomial" time. However, there is a category of problems that are not solvable in a reasonable amount of time (reasonable is relative of course). We categorize those problems differently. They are:

    - P
    - NP
    - NP Hard
    - NP Complete

    Each of these problem categories get progressively harder. Where NP Complete problems are basically problems that could take **years** to solve and P problems take **seconds** or **minutes** to solve.

    - Discuss the possibility of converting NP to P, is it possible?

    - What are the ramifications of converting a problem that is categorized as NP into a solution in P (aka making a Non-Polynomial solution solvable in Polynomial time) (this question should give you a hint that its currently not possible).
    - Also, discuss approximation algorithms and how they fit into the big picture for solving very hard problems.

---

16. Do you notice any relationship between problem solving paradigms and P vs NP?

Note:

- Be careful with the next 3 questions.
- Try to tie in each answer with a paradigm, mention complexity, and also discuss if it's an optimal solution.

---

17. Assume you are part of a relief effort in a third world nation. You need to drop medical supplies to as many locals as possible from your AC-130 aircraft supplied to you by some participating nation. Your goal is to leave your home base, visit as many villages as possible by air, drop your supplies to each, and return to home base.

    - Describe an algorithm to help you achieve this goal.
    - Is there an existing algorithm to help you solve this problem?
    - Will the algorithm provide an optimal solution?

---

18. You are given the task of writing a class to implement "undo" / "redo" for a popular word processor.

    - What data structure do you choose?

- Describe your algorithm for both undo and redo. Do they use the same data structure?
- Do you use static memory allocation or dynamic memory allocation?

---

19. You are in charge of campus tours for the school year 2020-2021. Forget about covid19 and concentrate on your tours. There are a minimum set of places that each family needs to visit in varying locations on campus. They are all over campus and can be reached from multiple paths, and in any order. Ideally you would start in one place, visit every location, and end up where you started.

    - How do you solve this problem?
    - Do you use a data structure an algorithm or both?
    - Can your solution satisfy all involved? Meaning will it be efficient and not waste time?

---

20. You have been tasked to come up with a schedule of courses for Spring 2021 semester. You are given all the necessary course requirements (classes that need to be scheduled) and the list of available rooms on campus.

    - What algorithms will you use to perform the scheduling?
    - What problem solving paradigm is your algorithm in?
    - Can you com up with a perfect solution?

---

21. The government announced that we have reached herd immunity! There are many many party's planned all over town. You would like to visit each party, but you don't want to waste time talking to people below your station when you do (play along, its a word problem for gosh sake's). You have your own "people" who have scouted all of these party's and have collected information on everyone. Pretend you're in a movie, and some girl just jammed keys on a keyboard (clickety clickety clack clack clickety) and "hacked" everyones cellphone. Using this info, she has assigned a value to each person that gives them a score of "interestingness" (new word). Let us assume you have already determined the most efficient route to visit the max number of party's, but now you need to maximize your quality time at each party by only visiting with the most "interesting" person at each party. If the person with the best "interestingness" score leaves your current party while you are still there, you simply move on to the next most interesting person still remaining.

    - What algorithm(s) can you employ to make this happen?
    - What problem solving paradigm is your algorithm in?
    - Is there a way to guarantee you will maximize your interaction with interesting people?

Binary Search

- When searching for an item in an array of values, what is a necessary pre-condition before an actual "binary search" can be performed? (look below to see answer)
- If given an unsorted array, you can search it in linear time O(N). But if given a sorted array you can search it in _____ time.
- One question is: If given an unsorted array, when would the cost of sorting those N values become worth it so you can perform a binary search? What scenarios may or may not justify that cost?

Tracing Binary Search Code:

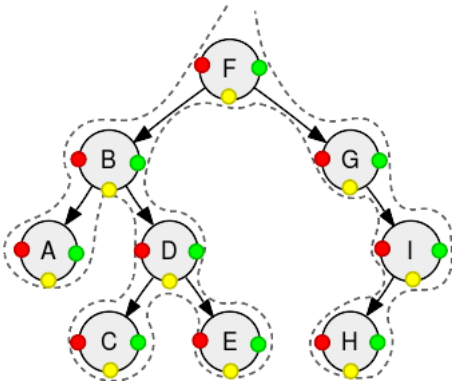
```
int left;   // = ??
int right;  // = ??
int middle; // = ??

// index          0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
int[] numbers = {-2,  4,  6,  7, 11, 15, 29, 33, 38, 45, 56, 59, 70, 80, 81, 99};

// search for the value -10
int index = binarySearch(numbers, -10);

// Show the values of "Left", "Right", and "Middle" as your search progresses for ea
```

Tree Traversals



- Perform a Pre-Order, Post-Order, and In-Order traversal on the binary tree above.
- Notice that the tree has Red, Yellow, and Green dots. If you follow the dotted path, when you encounter each dot it implies what order traversal you are performing: Red = Pre, Yellow = In, and Green = Post
- Don't forget that the recursive function to traverse a tree helps you remember as well (see below).

```
void TreeTraverse(Node* root){
   if(!root){
       return;
   }
   // Place cout here for PreOrder
   TreeTraverse(root->Left);
   // Place cout here for InOrder
   TreeTraverse(root->Right);
   // Place cout here for PostOrder
}
```

Graph Traversals

- Know how to perform a DFS and a BFS on a given graph.
- Each of those traversals utilize a specific data structure to ensure the proper order is obtained. know which traversal is paired with which data structure.
- Know the complexity of a graph traversal.
- What are the maximum number of edges that a graph can contain (fully connected graph, with no repeating edges, and no self edges).