Final Exam Study Guide 😊

1. Understand the different sorting algorithms we have seen in class and their corresponding average case and worst-case complexities: Insertion Sort, Selection sort, Bubble Sort, QuickSort, MergeSort, and HeapSort.
2. Understand the insertion and deletion operations on a Max-heap and min-heap.
3. Know the array-representation of the heap ADT.
4. Know the graph properties (connected, weighted, complete, multigraph…) and their representations using adjacency list and adjacency matrix.
5. Understand the graph traversals algorithms: BFS, DFS.
6. Be able to find the minimum-spanning-tree (MSP) using BFS-traversal, DFS-traversal and Prim’s algorithm for weighted graphs.
7. Understand recursive functions and be able to write one depending on the problem.
Final Exam: Review Problems

**PART I: Problems:**

**Problem 1:**

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. You are only required to show the final heap, although if you draw intermediate heaps.

**Problem 2:**

Use the following graph for this problem. Where needed and not determined by the algorithm, assume that any algorithm begins at node A.
a) Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.

b) Assume that the graph is unweighted. Draw both the adjacency matrix and adjacency list representations of this graph. Be sure to specify which is which.

c) Draw the MST (minimum Spanning Tree) on the graph above resulting from Prim’s algorithm starting from node A.

d) Draw the MST (minimum Spanning Tree) resulting from DFS traversal starting from node A following an alphabetical order in case of more than 1 choice of the node. (ignore the weights of the graph)
Problem 3:

a. Draw the min-heap that results from inserting: 60, 18, 9, 25, 12, 6, 14, 8, 17 in that order into an initially empty heap.

b. Draw the result of doing one delete on the heap you created in part a.

Problem 4: Consider the following graph. If there is ever a decision between multiple neighbor nodes, assume we always choose the letter closest to the beginning of the alphabet first.
a. The graph below is: (circle the correct answers):
- complete
- directed
- weighted
- connected
- multigraph
- unweighted
- subgraph
- disconnected

b. What is the order of visited nodes using BFS algorithm (starting from node A). 
   ABDCEGHF

c. What is the order of visited nodes using DFS algorithm (starting from node A). 
   ABCEHFGD

Problem 5:
Given the graph below If there is ever a decision between multiple neighbor nodes, assume we always choose the node with the greatest value to traverse first.
Check this resource: [https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html](https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html)

a. What is the order of visited nodes using Prim’s algorithm (starting from node 4).  
   Answer: 453102

b. What is the order of visited nodes using Prim’s algorithm (starting from node 1).  
   Answer: 145302

c. What is the order of visited nodes using Prim’s algorithm (starting from node 2).  
   Answer: 235410

d. What is the order of visited nodes using Prim’s algorithm (starting from node 0).  
   Answer: 014532

e. What is the order of visited nodes using Prim’s algorithm (starting from node 5).  
   Answer: 543102

**Problem 6:** Write a recursive method called `power()` that, given an integer n, returns $2^n$ (2 to the nth power). `power()` should not invoke any auxiliary methods.

```java
public static int power(int base, int powerRaised) {
    if (powerRaised != 0)
        return (base * power(base, powerRaised - 1));
    else
        return 1;
}
```
Problem 7: Write a recursive method called `sum()` that takes a positive integer from the user and calculates the sum up to the given number. For example:

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=5</td>
<td>15</td>
</tr>
</tbody>
</table>

```java
public static int addNumbers(int num) {
    if (num != 0)
        return num + addNumbers(num - 1);
    else
        return num;
}
```

Problem 8: Write a recursive method called `toBinary()` that converts a decimal number to its binary String representation.

```java
public String toBinary(int n) {
    if (n <= 1) {
        return String.valueOf(n);
    }
    return toBinary(n/2) + String.valueOf(n % 2);
}
```

Problem 9: The following array represents a heap stored in an array in the manner discussed in lecture.

a. Show the tree represented by this array:

```
0 95
1 77
2 88
3 11
4 45
5 85
6
```

![Tree Diagram]

```
95
77 88
11 45 85
```

```
0 95
1 77
2 88
3 11
4 45
5 85
6
```
b. Show the contents of the array after 105 is inserted into the original heap.

\[
\begin{array}{c|c}
0 & 105 \\
1 & 77 \\
2 & 95 \\
3 & 11 \\
4 & 45 \\
5 & 85 \\
6 & 88 \\
\end{array}
\]

c. Show the contents of the array after deletion is executed twice.

\[
\begin{array}{c|c}
0 & 85 \\
1 & 77 \\
2 & 45 \\
3 & 11 \\
\end{array}
\]

Problem 10:

Consider the following sorting algorithms: InsertionSort, SelectionSort, MergeSort, QuickSort, and HeapSort.

a. Which of the above sorting techniques run in worst-case time \(O(n \log n)\)? **MergeSort and HeapSort**

b. Which of the above sorting techniques run in time \(O(n)\) in the best case? **InsertionSort** (You can prepare a Heap in time \(O(n)\), but you can’t take items out of it fast enough)

c. Which of the above sorting techniques are based on the Divide-and-Conquer strategy? **MergeSort and Quicksort**

Problem 11:

a. Does the following array: 20 15 18 7 9 5 12 3 6 2 form a Max-Heap? Justify.

   \[
   \begin{align*}
   &A[6], \ldots, A[10] \text{ have no children.}
   \end{align*}
   \]

b. Which sort will operate in quadratic time relative to the number of elements in the array? **Bubble, insertion, selection**

c. In a max-heap with \(n\) elements, extracting the maximal value (and fixing the heap) requires how many key comparisons? **\(O(\log n)\)**

d. Which of the sorting algorithms yield approximately the same worst-case and average-case running time behavior in \(O(n\log n)\)? **MergeSort and HeapSort**

e. Given a set of keys in decreasing order, say \(n, n-1, n-2, \ldots 1\). We sort it using insertion sort. About how many comparisons are done? **\(n^2\)**

f. If you care more about the worst case performance than average case performance of a sorting algorithm which sorting algorithm(s) would you choose? **MergeSort or HeapSort**
If you care more about the worst case performance than average case performance of a sorting algorithm and you do not want to use any extra space which sorting algorithm(s) would you choose? **HeapSort**

If the average case performance matters more than worst case performance which sorting algorithm(s) would you choose? **MergeSort or HeapSort or Quicksort**

The worst case time complexity of a typical implementation of QuickSort is $O(n^2)$. The worst case occurs when the picked pivot is always an extreme (smallest or largest) element. What improvement could be made to make is $O(n\log n)$ complexity?

Yes, it is possible. The idea is to find the median element of an unsorted array which can be found in linear time. So we find the median first, then partition the array around the median element.

**Problem 12:** The following code represents one of the sorting algorithms we have seen in class.

```c
void sort(int arr[]){
    int i, k, n = arr.length;
    for (i = 0; i < n+2; i++){
        int k = n;
        for (int j = i+1; j < n; j++)
            if (arr[j] < arr[k])
                k = j;
        int temp = arr[k];
        arr[j] = arr[i];
        arr[i] = temp;
    }
}
```

a. What is the name of the algorithm? **SelectionSort**

b. There are 4 errors in the code, fix them.

```c
void sort(int arr[]){
    int i, k, n = arr.length;
    for (int i = 0; i < n-1; i++){
        int k = i;
        for (int j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                k = j;
        int temp = arr[k];
        arr[k] = arr[i];
        arr[i] = temp;
    }
}
```

**Problem 13:** The following code represents one of the sorting algorithms we have seen in class.

```c
void sort(int arr[]){
    int i, j, n = arr.length;
    for(i = 0; i < n-1; i++)
        for(j = 0; j < n-1; j++)
            if (arr[j] > arr[j+1])
                {
                int temp = arr[j];
                arr[j+1] = arr[j];
                arr[j] = temp;
                }
}
```

a. What is the name of the algorithm? **BubbleSort**
b. There are 2 errors in the code, fix them.

```java
void sort(int arr[])
{
    int i, j, n = arr.length;
    for(i = 0; i < n-1; i++)
        for(j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
}
```

Problem 14: Trace the following sorting algorithms as they process the arrays

a. Bubble Sort → 29, 17, 3, 94, 46, 8, -4, 12

```
index
original {29, 17, 3, 94, 46, 8, -4, 12}
sweep 1 {17, 3, 29, 46, 8, -4, 12, 94}
sweep 2 {3, 17, 29, 8, -4, 12, 46, 94}
sweep 3 {3, 17, 8, -4, 12, 29, 46, 94}
sweep 4 {3, 8, -4, 12, 17, 29, 46, 94}
sweep 5 {3, -4, 8, 12, 17, 29, 46, 94}
sweep 6 {-4, 3, 8, 12, 17, 29, 46, 94}
```

b. Selection Sort → 14, 25, 95, 0, 17, -2, 13, 56, 34

```
index
original {14, 25, 95, 0, 17, -2, 13, 56, 34}
pass 1 {-2, 25, 95, 0, 17, 14, 13, 56, 34}
pass 2 {-2, 0, 95, 25, 17, 14, 13, 56, 34}
pass 3 {-2, 0, 13, 25, 17, 14, 95, 56, 34}
pass 4 {-2, 0, 13, 14, 25, 17, 95, 56, 34}
pass 5 {-2, 0, 13, 14, 17, 25, 95, 56, 34}
pass 6 {-2, 0, 13, 14, 17, 25, 95, 56, 34}
pass 7 {-2, 0, 13, 14, 17, 25, 34, 56, 95}
pass 8 {-2, 0, 13, 14, 17, 25, 34, 56, 95}
```

c. Insertion Sort → 6, 7, 4, 11, 8, 1, 10, 3, 5, 2

```
index
original {6, 7, 4, 11, 8, 1, 10, 3, 5, 2}
pass 1 {6, 7, 4, 11, 8, 1, 10, 3, 5, 2}
pass 2 {4, 6, 7, 11, 8, 1, 10, 3, 5, 2}
pass 3 {4, 6, 7, 11, 8, 1, 10, 3, 5, 2}
pass 4 {4, 6, 7, 8, 11, 1, 10, 3, 5, 2}
pass 5 {1, 4, 6, 7, 8, 11, 10, 3, 5, 2}
pass 6 {1, 4, 6, 7, 8, 10, 11, 3, 5, 2}
pass 7 {1, 3, 4, 6, 7, 8, 10, 11, 5, 2}
pass 8 {1, 3, 4, 5, 6, 7, 8, 10, 11, 2}
pass 9 {1, 2, 3, 4, 5, 6, 7, 8, 10, 11}
```
Problem 15: Suppose that the in quicksort algorithm instead of choosing the pivot as the first element in the array, the middle value of the three values array[first], array[(first+last)/2], and array[last] is chosen as the new pivot. Modify the algorithm we have seen in class to include the new change.

```java
public static void swap(int[] arr, int a, int b) {
    int temp = arr[a];
    arr[a] = arr[b];
    arr[b] = temp;
}

void quicksort(int[] arr, int left, int right) {
    int pivot = medianof3(arr, left, right);
    int j = partition(arr, left, right, pivot);
    quicksort(arr, left, j - 1);
    quicksort(arr, j + 1, right);
}

public static int medianof3(int[] arr, int left, int right) {
    int mid = (left + right) / 2;
    if (arr[mid] < arr[left])
        swap(arr, mid, left);
    if (arr[right] < arr[left])
        swap(arr, right, left);
    if (arr[mid] > arr[right])
        swap(arr, mid, right);
    return arr[right - 1];
}
```

Problem 16: Show the values of the arguments of the Quicksort algorithm as it is applied on the following array: 4,14, 6, 9, 7, 22, 3, 8

- Quicksort called with low = 0 and high = 7
- Quicksort called with low = 2 and high = 7
- Quicksort called with low = 3 and high = 7
- Quicksort called with low = 3 and high = 4
- Quicksort called with low = 6 and high = 7
Problem 17: Use a max-Heap to sort the following array in descending order: 4, 10, 3, 5, 1 and show the content of the heap in every step.

![Heap Diagram]

The sequence of deleted nodes: 10 -> 5 -> 4 -> 3 -> 1

Problem 18: Consider the following max-heap:

```
10
/  \
5  3
/ /  \
2 4
```

a. Show the heap after a deletion operation

Step 1: Replace the last element with root, and delete it.

```
4
/  \
5  3
/ \
2
```

Step 2: Rebuild heap root.
b. Show the heap after inserting 15

Step 1: Insert the new element at the end.

```
   10
 / \
5  3
/ \ /\ /\
2  4 15
```

Step 2: Rebuild heap the new element following bottom-up approach.

- > 15 is less than its parent 3, swap them.

```
   10
 / \
5  15
/ \ /\ /\
2  4 3
```

- > 15 is again less than its parent 10, swap them.

```
  15
 / \
5  10
/ \ /\ /\
2  4 3
```

Therefore, the final heap after insertion is: 15 is less than its parent 3, swap them.

```
  15
 / \
```

13
PART II: Multiple Choice Questions:

1. The time complexity of a quick sort algorithm which makes use of median, found by an O(n) algorithm, as pivot element is
   a) O(n²)
   b) O(nlogn)
   c) O(nloglogn)
   d) O(n)

2. The time complexity of heap sort in worst case is
   a) O(logn)
   b) O(n)
   c) O(nlogn)
   d) O(n²)

3. Which of the following algorithm pays the least attention to the ordering of the elements in the input list?
   a) Insertion sort
   b) Selection sort
   c) Quick sort
   d) None

4. What is the time complexity of bubble sort in best case?
   a) O(n)
   b) O(nlogn)
   c) O(n²)
   d) O(n(logn)²)

5. Which of the following algorithm design technique is used in the quick sort algorithm?
   a) Dynamic programming
   b) Backtracking
   c) Divide-and-conquer
   d) Greedy method

6. For merging two sorted lists of size m and n into sorted list of size m+n, we require comparisons of
   a) O(m)
   b) O(n)
   c) O(m+n)
   d) O(logm + logn)
7. In a binary **max heap** containing n numbers, the smallest element can be found in time
   
   a) $O(n)$  
   b) $O (\log n)$  
   c) $O (\log \log n)$  
   d) $O (1)$

8. Which of the following algorithm(s) can be used to sort n integers in range [1.....n3] in $O(n)$ time?
   
   a) Heap sort  
   b) Quick sort  
   c) Merge sort  
   d) Radix sort

9. If the given input array is sorted or nearly sorted, which of the following algorithm gives the best performance?
   
   a) Insertion sort  
   b) **Selection sort**  
   c) Quick sort  
   d) Merge sort

10. Select the appropriate code that performs selection sort.
    
    a)

    ```
    int min;
    for(int j=0; j<arr.length-1; j++)
    {
        min = j;
        for(int k=j+1; k<arr.length-1; k++)
        {
            if(arr[k] < arr[min])
                min = k;
        }
        int temp = arr[min];
        arr[min] = arr[j];
        arr[j] = temp;
    }
    
    b)

    int min;
    for(int j=0; j<arr.length-1; j++)
    {
        min = j;
        for(int k=j+1; k<arr.length; k++)
        {
            if(arr[k] < arr[min])
                min = k;
        }
        int temp = arr[min];
        arr[min] = arr[j];
        arr[j] = temp;
    }
    ```
c) 

```
int min;
for (int j=0; j<arr.length-1; j++)
{
    min = j;
    for (int k=j+1; k<=arr.length-1; k++)
    {
        if(arr[k] > arr[min])
            min = k;
    }
    int temp = arr[min];
    arr[min] = arr[j];
    arr[j] = temp;
}
```

11. The given array is `arr = {1,2,3,4,5}`. The number of iterations in selection sort and bubble sort respectively are
   a) 5 and 4
   b) 1 and 4
   c) 0 and 4
   d) 4 and 1

12. What is the best case complexity of selection sort?
   a) O(nlogn)
   b) O(logn)
   c) O(n)
   d) O(n^2)

13. Consider the following code snippet:
```
public static void my_recursive_function()
{
    my_recursive_function();
}
```
```
public static void main(String[] args)
{
    my_recursive_function();
    return 0;
}
```

What will happen when the above snippet is executed?
   a) The code will be executed successfully and no output will be generated
   b) The code will be executed successfully and random output will be generated
   c) The code will show a compile time error
   d) The code will run for some time and stop when the stack overflows

15. What is the output of the following code?
```
public static void my_recursive_function(int n)
{
```
if(n == 0)
    return;
    my_recursive_function(n-1);
    System.out.println(n);
}
public static void main(String[] args)
{
    my_recursive_function(10);
    return 0;
}

a) Prints the numbers from 10 to 1
b) Prints the numbers from 10 to 0
c) **Prints the numbers from 1 to 10**
d) Prints the numbers from 0 to 10

16. For the given graph(G), which of the following statements is true?

![Graph](image.png)

a) G is a complete graph
b) G is not a connected graph
c) **G is a connected graph**
d) G is a tree

16. For a given graph G having v vertices and e edges which is connected and has no cycles, which of the following statements is true?
a) v=e
b) **v = e+1**
c) v + 1 = e
d) None of the mentioned

17. What is the specialty about the inorder traversal of a binary search tree?
a) It traverses in a non increasing order
b) It traverses in an increasing order
c) It traverses in a random fashion
d) None of the mentioned
18. What would be the DFS traversal of the given Graph?

![Graph Image]

a) ABCDE  
b) AEDCB  
c) EDCBA  
d) ADECB

19. For a sparse graph (a graph that does not contain a lot of edges) an adjacency list is more space efficient against an adjacency matrix.  
a) True  
b) False

20. For the given conditions, which of the following is in the correct order of increasing space requirement?  
i) Undirected, no weight  
ii) Directed, no weight  
iii) Directed, weighted  
iv) Undirected, weighted  
a) ii iii i iv  
b) i iii ii iv  
c) iv iii i ii  
d) i ii iii iv

21. Which of the following is the valid min heap?  
a)
22. Which one of the following array elements represents a binary min heap?
   a) 12 10 8 25 14 17
   b) 8 10 12 25 14 17
   c) 25 17 14 12 10 8
   d) 14 17 25 10 12 8

23. What will be the position of 5, when a max heap is constructed on the input elements 5, 70, 45, 7, 12, 15, 13, 65, 30, 25?
   a) 5 will be at root
   b) 5 will be at last level
   c) 5 will be at second level
   d) 5 can be anywhere in heap

24. If we implement heap as min-heap, deleting root node (value 1) from the heap. What would be the value of root node after second iteration if leaf node (value 100) is chosen to replace the root at start.
   a) 2
   b) 100
   c) 17
   d) 3

25. Which of the following is false in the case of a spanning tree of a graph G?
   a) It is tree that spans G
   b) It is a subgraph of the G
   c) It includes every vertex of the G
   d) It can be either cyclic or acyclic

26. Every graph has only one minimum spanning tree.
   a) True
   b) False

27. Which of the following is false?
   a) The spanning trees do not have any cycles
   b) MST have n – 1 edges if the graph has n edges
   c) Edge e belonging to a cut of the graph if has the weight smaller than any other edge in the same cut, then
the edge e is present in all the MSTs of the graph

d) Removing one edge from the spanning tree will not make the graph disconnected

28. Which of the following is not the algorithm to find the minimum spanning tree of the given graph?
   a) DFS-traversal algorithm
   b) Prim’s algorithm
   c) Kruskal’s algorithm
   d) Dijkstra algorithm

29. Consider the graph shown below. Which of the following are the edges in the MST of the given graph?

   a) (a-c)(c-d)(d-b)(d-b)
   b) (c-a)(a-d)(d-b)(d-e)
   c) (a-d)(d-c)(d-b)(d-e)
   d) (c-a)(a-d)(d-c)(d-b)(d-e)

30. The height of a heap with n elements:
   a) n+1
   b) log(n)
   c) n log(n)
   d) None of the above