CS 203 COMPUTING AND ALGORITHMS III
HOMEWORK QUESTIONS

Chapter 1- Introduction to Algorithms and Complexity

1. Design an algorithm for computing sqrt(n) for any positive integer n. Besides assignment and comparison you can inly use four basic arithmetic operations.

2. Write an algorithm for each of the tasks below and find the worst case, best case and average case times for the following algorithms.

   a. Find if an array has repeated elements.

   b. Find the smallest element in an array.

3. Write an algorithm to find the second largest element in an array.

   Exactly how many comparisons are done in the worst case?

4. Write an algorithm to find the largest and smallest element in an array. In general this should take 2n comparisons if you wrote two independent algorithm. Try to write an algorithm that does this in 1.5n comparisons.

5. List the following functions from lowest to highest order.

   n 2^n $n\lg n$ $\ln n$ $n-n^3+7n^5$ $\lg n$ $n^{.5}$ e^n

   $n^2 + \lg n$ n^2 2^{n-1} $\lg\lg n$ n^3 $(\lg n)^2$ $n!$

   n^{1+e} where $0 < e < 1$.

6. Describe a standard algorithm for finding the binary representation of a positive decimal integer

   a) In English

   b) In Pseudocode

7. Find gcd (31415, 14142) by applying Euclid's algorithm.

   How much more work will be done if use the other algorithm that checks every number for being gcd starting from min(m,n).

8. Prove that equality gcd(m,n) = gcd (n, m mod n) for every pair of positive integers m and n.

10. There are 22 pairs of gloves in a drawer. 5 pairs are red, 4 pairs are yellow and 2 pairs are green. You select the gloves in dark and can check them only after selection is made. What is the smallest number of gloves you need to select in order to have one pair of matching gloves, in the best case? in the worst case?

*11. You are facing a wall that stretches infinitely in both directions. There is a door in the wall, but you neither know how far away it is or which direction it is. You can see the door only when you get there. Design an algorithm that reaches you to the door by walking at most O(n) steps where n is the number of steps between you and the door. Prove that your algorithm is O(n).

12. Compute the following sums. Each of the summation goes over i.

a) $$\sum_{i=3}^{i=n+1} 1$$

b) $$\sum_{i=0}^{i=n-1} i(i+1)$$

c) $$\sum_{i=n}^{i=n}\sum_{j=1}^{j=n} i*j$$

d) $$\sum_{i=1}^{i=n}\frac{1}{i*(i+1)}$$

13. Find the order of growth for the following:

a) $$\sum_{i=0}^{i=n-1}(i^2+1)^2$$

b) $$\sum_{i=0}^{i=n}(i+1)*2^{i-1}$$

14. Solve the following recurrence relations

a) x(n) = 3x(n-1) , n > 1 and x(1) = 4

b) x(n) = x(n-1) + n, n > 0, x(0) = 0

c) x(n) = x(n/2) + n , n > 1 x(1) = 1

[solve when n = 2^k]

d) x(n) = x(n/3) + 1, n > 1, x(1) = 1

[solve when n = 3^k]

15. Write a recursive algorithm for finding the nth fibonacci number and find the time complexity for finding the nth fibonacci.

16. Consider the following formula for finding the sum of first n cubes

S(n) = S(n-1) + n*n*n; , n >1

S(1) = 1

Write the recursive algorithm and find the time complexity.

Chapter 2 - Searching

1. Write out an algorithm to find x in an ordered list by comparing x to every fourth element until x is found or an entry larger than x is found, and in the later case, compares x with previous three. How many comparisons are done in the worst case by this algorithm?

2. How can you modify binary search to eliminate unnecessary work if you are certain that x is in the list? Find the worst case and average case time complexity of this modified algorithm. (you may assume that $n=2^k-1$ for some k)

3*. The first n cells of an array L contain integers sorted in increasing order. The remaining cells all contain some very large integer that we may think of as infinity. The array may be arbitrarily large (you may think of it as infinite) and you don't know n. Give an algorithm to find the position of a given integer x (x < maxint) in the array in logn time.

4. What is the largest number of key comparisons made by binary search for a key in the following array?

3 14 27 31 39 42 55 70 74 81 85 93 98

List all the elements in the array that will require largest number of key comparisons.

5. Sequential search can be used with about same efficiency whether the list is an array of linked list. Is it also tru e for binary search? If not, explain the time complexity of binary searching a linked list of sorted elements.

6. How can one use binary search for range searching, for finding all the elements in a sorted array whose values fall between two given values L and U (inclusively) L <= U. What is the worst case efficiency of the algorithm?

7. Write a code to find maximum in a 2-3 tree. State the time complexity of the algorithm.

8. Write a code to find successor if a node is a 2-3 tree.

9. What is the time complexity of your algorithm.

10. How would you check if the given tree is a 2-3 tree. What is the time complexity of your algorithm.

11. Why are balanced trees useful?

12. Use a good hashing functions to store all the words in the first four lines of the following poem. Is your search time 1? (do not use linear or quadratic probing technique)

“ Twinkle Twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky”

13. Insert the words in the poem above using both linear and quadratic probing. You may use string length as the data you will insert. Then search for “high” and “star”, “sky” and “you”. How many probes were needed in linear probing? How many were needed in quadratic probing?

Chapter 3- Sorting

Note : A sorting method is "stable" if equal keys remain in the same relative order in the sorted list as they were in the original list. That is, let us say that L[i] = L[j] and i < j in the original list and after the sorting,let us say L[i] was moved to L[k] and L[j] was moved to L[m], then sort is stable only if k < m. Which of the following sorting algorithms are stable?

1. Run the smallest based sort (selection sort) on the following array. Show the pointers smallestIndex and fix.

```
45  67  78  90  34  45  8
```

2. Prove if selection sort is stable or unstable.

3. Run the insertion sort (bubble sort) on the following array. Show the i and j .

E X A M P L E

4. Is it possible to adopt insertion sort to linked list (doubly linked). Will it have the same $O(n^2)$ time complexity?

5. Write a divide and conquer algorithm for finding the largest and smallest number in an array. Set up and solve the recurrence relation that represents the time complexity of your algorithm.

6. Write the merge code for mergesort so that it uses two arrays for storage.

7. Show the demonstration of mergesort on the following array.

   56 45 67 78 90 34 45 8

8. Is mergesort a stable algorithm? Prove or disprove.

9. Let A[1..n] be array of n distinct real numbers. A pair (A[i], A[j]) is said to be an inversion if these numbers are out of order. i,e i < j but A[i] > A[j]. Design an O(nlogn) algorithm for counting the number of inversions.

10. There are 2n glasses standing next to each other in a row. The first n glasses are filled with a soda drink, while the remaining n glasses are empty. Make the glasses alternate in a filled-empty-filled-empty pattern in the minimum number of glass moves.

11. Demonstrate the split algorithm on the following array. Show the pointers s, y low and high.

   56 45 67 78 90 34 45 8

12. Demonstrate the quick sort algorithm on the following array. Show all the split algorithms and the low and

high for each call.

56 45 67 78 90 34 45 8

13. What is the worst case space complexity of Quicksort. Explain your answer. Describe the worst case scenerio.

14. What is the best case space complexity of Quicksort. Explain your answer. Describe the best case scenerio.

15. Find the time complexity of quick sort when split takes O $(n^{.9})$ time and split in the middle.

16. Show that it is impossible to have a split algorithm for quicksort that takes $O(n^{.9})$ time and always splits in the center of the array.

17. Design an algorithm to rearrange an array of characters R, W and B (Red White and Blue, which are colors in the Dutch National Flag) so that all R's come first, and W's come next and finally all B's. Design a linear in-place algorithm for the problem.

18. Although we have shown that O(n) is the worst case space complexity of Quicksort, it is possible to reduce this to O(logn). Find the technique that will accomplish this.

19. (Nuts and Bolts) You are given a collection of n bolts of different widths and corresponding n nuts. You are allowed to try a nut and a bolt together from which you can determine if the nut is larger than the bolt, smaller than the bolt or fits perfectly. However, there is no way to compare two nuts together or two bots together. The problem is to match each bolt with its nut. Design an algorithm for this problem with average case efficiency of O(nlogn).

20. The problem is to sort 500 exam papers alphabetically by students last name. The sorting will be done in the

office with two desktops temporarily cleared of all other papers, books and coffee cups. It is 1AM and the person wants to go home as soon as possible. Which sorting algorithm would you recommend.

21. Show that execution of heapsort on the following array.

C O M P L E X

22. Is heapsort a stable algorithm? Why or why not?

23. Run the Shell sort algorithm on the following array:

78 90 34 56 89 103 61 9 2 34 24 19 45

Use h- sequence as { 4,2,1}

24. Demonstrate the most significant and least significant radix sort on the following elements. Describe the space and time complexity of each algorithm.

278 490 334 156 189 103 161 19 72 134 724 819 745

CHAPTER 4 - Graphs

1. Write an algorithm to find a <u>simple</u> path between x and y, given a path between x and y in a graph G. Develop the time complexity analysis for your algorithm.

2. a)Let A be an adjacency matrix of an undirected graph. Explain what property of the matrix indicates that

i) The graph is complete

ii) the graph has a self-loop, i.e an edge connecting a vertex to itself.

iii) The graph has an isolated vertex, i.e a vertex that has no edges incident to it.

3. Answer the above questions if the A was a adjacency list representation.

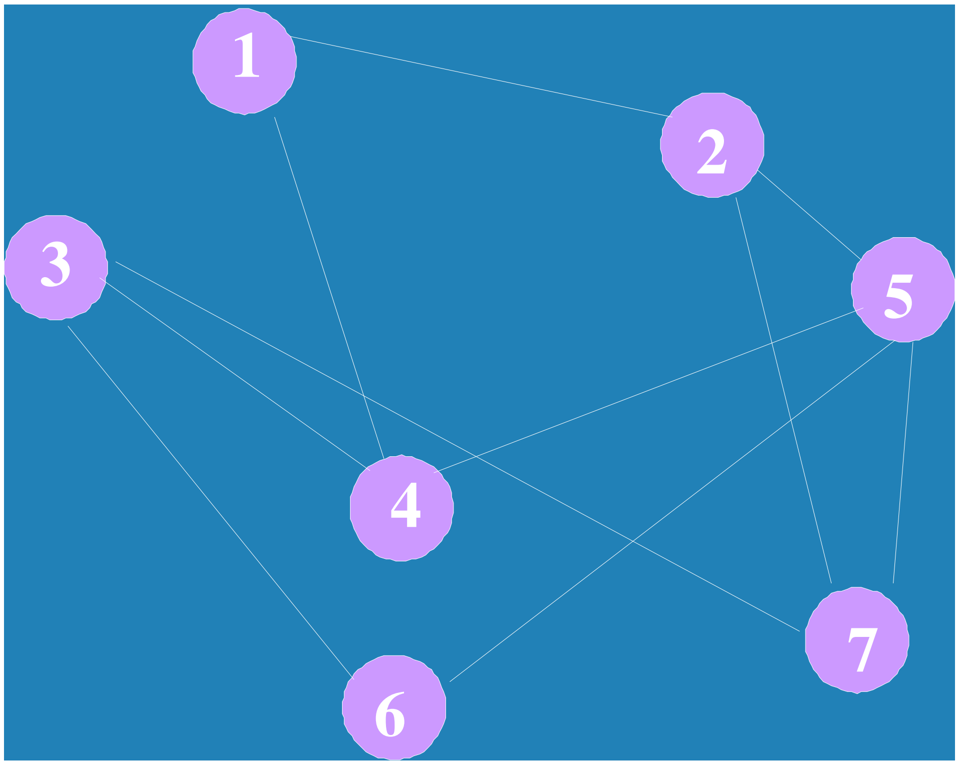
4. Write an algorithm to find a simple path between two given vertices in a rooted tree in its adjacency list representation.

5. Write an algorithm to check if a given graph is connected?

6. Write the detailed algorithm for BFS using adjacency list as the input data structure. Make sure the resulting is tree is created as a tree data structure.

7. Write the detailed algorithm for DFS using adjacency list as the input data structure. Make sure the resulting is tree is created as a tree data structure.

8. Run the BFS and DFS for the graph below:

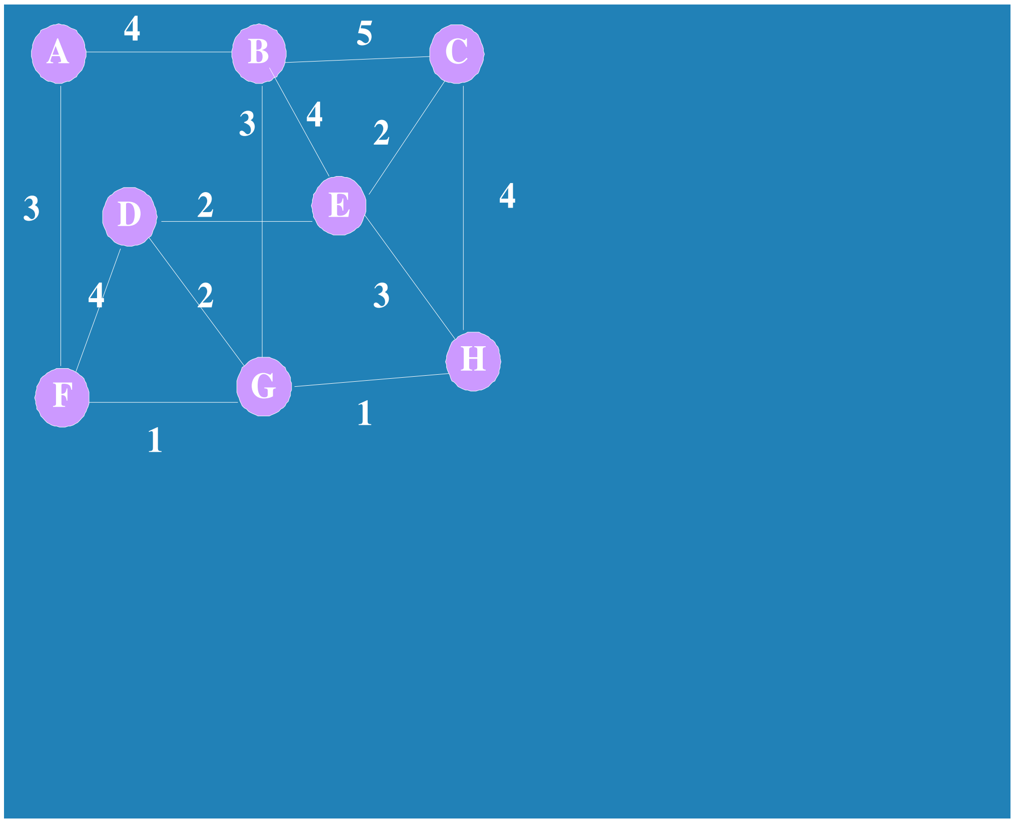


Show the relevant details

9. Write the detailed algorithm for Kruskal's MST using adjacency list as the input data structure. Ensure the time complexity of the algorithm is as described in the notes.

10. Write the detailed algorithm for Dijkstra-Prim's MST using adjacency list as the input data structure. Ensure the time complexity of the algorithm is as described in the notes.

11. Run the Kruskal's MST and Dijkstra-Prim's MST algorithm on the graph below: Show the details of each step.



12. Write an algorithm to find Euler circuit and Euler path whenever it exists. Develop the time complexity analysis for your algorithm.

13. Consider the clique problem - given a graph G and a positive integer k, determine whether the graph contains a clique of size k, i.e a complete subgraph of k vertices. Design an exhaustive-search algorithm for this problem.

14. What is the time efficiency of the DFS-based algorithm for topological sorting.

15. How can one modify the DFS-based algorithm to avoid reversing the vertex ordering generated by the DFS?

16. Apply the DFS based topological sorting algorithm to solve the topological sorting problem for the following digraph.

17. Apply Warshall's Dynamic Programming algorithm for finding the transitive closure of the digraph given by the following adjacency matrix.

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

18. Prove that the time efficiency of Warshall's algorithm is cubic.

19. Solve the all-pairs shortest-path problem for digraph with the weight matrix using Floyd's Dynamic Programming technique

$$\begin{bmatrix} 0 & 2 & \infty & 1 & 18 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

20. Give an example of a graph or digraph with negative weights for which Floyd's algorithm does not yield correct results.

Chapter 5 Coping with Hard Problems

1. Assuming that each tour can be generated in constant time, what will be the time complexity class of the exhaustive search algorithm for TSP.

2. . Hamiltonian circuit is defined as a tour in an undirected graph that visits every vertex exactly once. Outline an exhaustive-search algorithm for finding Hamiltonian circuit.

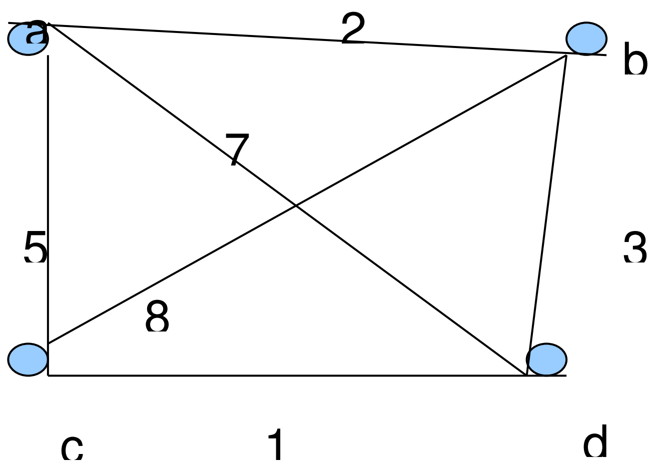
3. Give an example of the best case input for the branch and bound algorithm for the assignment problem. In the best case, how many nodes will be in the state space tree of the branch and bound algorithm of the assignment problem.

4. Solve the following instance of the knapsack problem by branch-and-bound algorithm.

item	Weight	value	
1	10	$100	
2	7	$63	W = 16
3	8	$56	
4	4	$12	

5. Suggest a more sophisticated bounding function for solving the knapsack problem.

6. Apply branch and bound algorithm to solve the TSP problem on the following graph.


a
2
b
7
5
3
8
c
1
d