CSE 214
Practice Problems

Time Complexity Analysis:

(1) Find the closed form total running time of the following code fragment. What is the corresponding running time in big-O notation?

```
int sum = 0;
for(int i = 0; i < n/2; i++)
     for(int j = 1; j <= n; j *= 2)
          sum = sum + (i*j);
```

(2) Find the closed form total running time of the following code fragment. What is the corresponding running time in big-O notation?

```
int sum = 0;
for(int i = 1; i <= n; i *= 2)
    for(int j = i; j <= n; j++)
          sum = sum + (i*j);
```

(3) Consider the following pseudocode:

```
1   int D = 2;
2   for(int i = 1; i <= n; i++)
3        for(int j = i+1; j <= n; j++)
4             for(int k = 1; k <= n; k *= 2)
5                  D = D * k;
```

What is the total number of multiplications to be performed at line 5.

Given:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

(4) Find the closed form total running time of the following code fragment. What is the corresponding running time in big-O notation?

```
int D = 2;
for(int i = 1; i <= n; i++)
     for(int j = i; j <= n; j++)
          for(int k = j+1; k <= n; k++)
               D = D * k;
```

Given:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Linked-List:

(5) Write Output:

Input List: 1 2 3 4 5 6 7 8 9	Input List: 1 2 3 4 5 6 7 8 9

`void method1(Node node) {`
`  if(node == NULL)`
`  return;`
 	
`  method1(node.next);`
	`void method2(Node head) {`
	`  if(head== null)`
	`  return;`
	`  System.out.println(head.data);`

`System.out.print(node.data);`
`}`
 	
`list.method1(list.head);`
	`if(head.next != null )`
	`method2(head.next.next);`
	`System.out.println(head.data);`
	`}`
	
	`list.method2(list.head);`

Output:	Output:

(6) Given a sorted linked list, write a method removeDuplicate() to delete all duplicates such that each element appears only once. For example,
Given 1->1->2, return 1->2.
Given 1->1->2->3->3, return 1->2->3.

(7) Given an unsorted linked list, write a method sortList(Node head) to sort the linked list.

(8) Complete the following method of a linked list for which you know the reference to the head node only. This method finds the middle node and set this middle node as the head of the list (remove middle and insert it at the head).

```
public void setMiddletoHead(){
      if(head == null)
            return;
      Node slow_ref = head; // traverse by one
      Node fast_ref = head; // traverse by skipping one
      Node prev = null; // keep track of previous of middle

      while(fast_ref != null && fast_ref.next != null){
            _____________________________________________;
            _____________________________________________;
            _____________________________________________;
      }
      ________________________________________________;
      ________________________________________________;
      head = slow_ref;
}
```

(9) Write a method reverse() to reverse the order of a singly linked list that contains references to both head and tail.

(10) Write a method called AddAtTail(int element) that adds a new element at the tail of a linked list having integer values. The list might be empty.

(11) Write a method to check equality between two linked lists. In Java, objects usually implement a “equals” method in the object’s class to do this.

(12) Given two numbers represented by two linked lists, write a function that returns sum list. The sum list is linked list representation of addition of two input numbers. It is not allowed to modify the lists. Also, not allowed to use explicit extra space (Hint: Use Recursion).

Example

Input:

First List: 5->6->3 // represents number 563

Second List: 8->4->2 // represents number 842

Output

Resultant list: 1->4->0->5 // represents number 1405

Stack & Queue:

(13) Implement a stack using a single queue. Specifically, write pseudocode for **push** and **pop** operations on a stack using enqueue and dequeue operations of queue. Consider the queue class is given to you. What is the time complexity of these push and pop operations?

(14) Implement a stack using two queues. Specifically, write pseudocode for **push** and **pop** operations on a stack using enqueue and dequeue operations of queue. Consider the queue class is given to you. What is the time complexity of these push and pop operations?

(15) Implement a queue using two stacks. Specifically, write pseudocode for **enqueue** and **dequeue** operations on a queue using push and pop operations of stack. Consider the stack class is given to you. What is the time complexity of these enqueue and dequeue operations?

(16) Write a method isBalanced(String str) that returns true if a string of parentheses is balanced and false if it isn't.

(17) Given the infix expression: a * b + (c + d - e)/(f * g) – h ^ i * j

a. Convert to postfix.

b. Convert to prefix.

(18) Write a method utilizing your knowledge of the stack data structure that takes an integer value n and prints out the binary representation of the integer n. [hint: use stack]

(19) The Celebrity Problem

When a Celebrity goes to a party, everyone else knows the Celebrity but Celebrity knows none. Given this condition and a list of people attending the party, can you figure out which one is the celebrity.

You're given a knows() method, which returns true if *A knows B* and false otherwise. For example: knows(person A, person B) returns *true* if person A knows person B. Note that if knows(A,B) is true, it's not necessary that knows(B,A) is true.

Implement the method (java code or pseudocode): int findCelebrity(int n) where n is the number of people in the party. This method returns the 'id' [0<= id <= n-1] of the person who is the celebrity.

Hint: You can use Stacks.

(20) Give an algorithm for reversing a queue Q. Only following standard operations are allowed on queue.

1. enqueue(x) : Add an item x to rear of queue.
2. dequeue() : Remove an item from front of queue.
3. isempty() : Checks if a queue is empty or not.

(21) Give an algorithm for reversing a stack S using recursion. Only following standard operations are allowed on stack.

1. push(x) : Add an item x to top of stack.
2. pop() : Remove an item from top of stack.
3. isempty() : Checks if a stack is empty or not.

(22) Give an algorithm for sorting an integer stack S using recursion. Only following standard operations are allowed on stack.

1. push(x) : Add an item x to top of stack.
2. pop() : Remove an item from top of stack.
3. isempty() : Checks if a stack is empty or not.
4. peek() : returns the top of the stack value without removing it

(23) Assuming that a queue is implemented using a singly linked list of IntNode nodes where front references the first node of the list only (there is no rear reference), what is the order of complexity of the enqueue operation if there are n nodes in the list?

(a) O(1) (b) O(n) (c) O(log n) (d) O(n^2) (e) none of these answers

Recursion and DP:

(24) Write a code for computing n choose k (combinations) Com(int n, int k) with recursion using top-down dynamic programming.

(a) Draw a graph illustrating the recursive call tree structure Com(5, 2) (find 5 choose 2) for a purely recursive approach

(b) How many recalculations you could avoid for Com(5, 2) by using top-down dynamic programming that you have just written?

(25) Write a code for finding nth Fibonacci number Fib(int n) with recursion using top-down dynamic programming.

(a) Draw a graph illustrating the recursive call tree structure Fib(7) for a purely recursive approach

(b) How many recalculations you could avoid for Fib(7) by using top-down dynamic programming that you have just written?

(26) Given a cost matrix cost[][] and a position (m, n) in cost[][], write a function that returns cost of minimum cost path to reach (m, n) from (0, 0). Each cell of the matrix represents a cost to traverse through that cell. Total cost of a path to reach (m, n) is sum of all the costs on that

path (including both source and destination). You can only traverse down, right and diagonally lower cells from a given cell, i.e., from a given cell (i, j), cells (i+1, j), (i, j+1) and (i+1, j+1) can be traversed. You may assume that all costs are positive integers.
(27) Write a program that recursively finds the number of occurrences of a given integer in an integer array.
(28) Write a recursive method that checks whether a given String is a palindrome.
(29) Consider a sequence of integers, where each number is the sum of the previous three numbers, except for the first three numbers. The first 10 integers of this sequence are 0, 0, 1, 1, 2, 4, 7, 13, 24, 44, … . Write a **recursive** function int tribonacci(int n), that takes n as a parameter and returns the nth number of this sequence.

Intro to Tree, Binary Tree, Binary Search Tree:
(30) (a)What is a complete binary tree? Give example. (b) What is the minimum and maximum number of nodes in a complete binary tree with a height of 4? (c) What is a full binary tree? Give example. (d) What is a binary search tree? Give example.
(31) Draw the binary search tree created by inserting these integers in the following order.
[25, 11, 38, 6, 13, 27, 40]
Write the in-order traversal of the nodes of this tree. Do you observe any interesting pattern in your answer?
(32) (a) Write an iterative method to find the maximum value in a binary search tree. This method will return the highest integer value in the tree. When the function is called, the root of the tree is passed as the argument to the function and it returns the integer value of the largest element in the binary search tree. You may assume that the nodes have all necessary access and modifier methods.
(b) Write a recursive method to find the maximum value in a binary search tree. You may make the same assumptions as before about the access and modifier methods. This function too when called initially, is passed the root as the parameter. Which method do you prefer (iterative/recursive)?
(33) Given the following integer values

50, 20, 80, 10, 30, 70, 90, 25, 40, 35, 45, 32, 37

(a) construct a binary search tree by inserting the elements in their given order.
(b) Draw the binary search tree after the following three lines of code

```
Btree.delete(30);
Btree.insert(33);
Btree.delete(25);
```

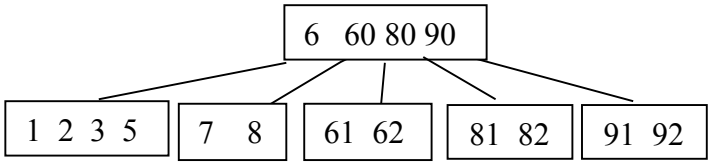
(c) What is the height and the postorder traversal sequence of the final tree after the operations performed in (b).
(d) Is the final tree an AVL tree?

(34) Consider the following expression:
a * b + (c + d - e)/(f * g) – h ^ i * j
Draw the corresponding expression tree for the above expression.

Heaps and Priority Queues:
(35) Consider a max heap, represented by the array: [40, 30, 20, 10, 15, 16, 17, 8, 4].

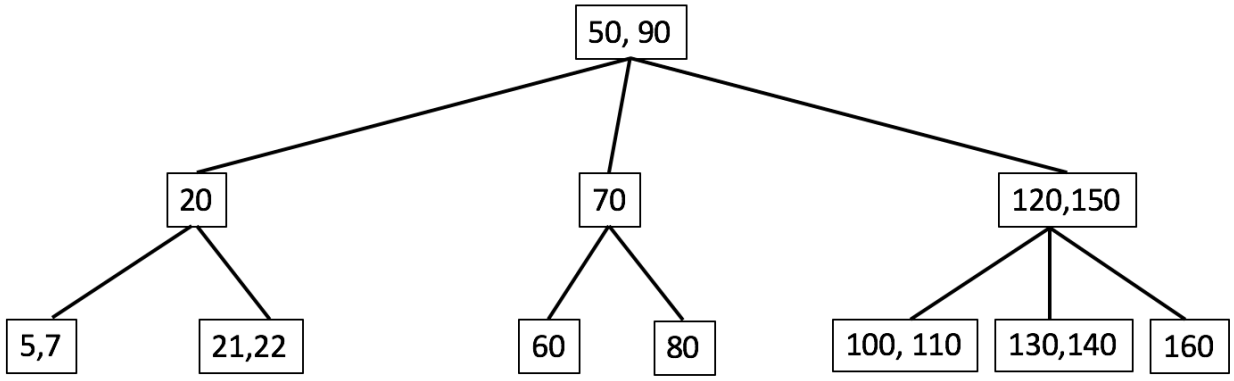
(a) Now consider that a value **35** is inserted into this heap. After insertion, draw the new heap.
(b) Now perform delete() twice on the heap obtained in (a) and draw the final heap.
(36) Construct a min-heap for the array [40, 30, 20, 10, 15, 16, 17, 8, 4].
(a) Now insert a value **3** into this heap. After insertion, draw the new heap.
(b) Now perform delete() twice on the heap obtained in (a) and draw the final heap.

Balanced Trees:
(37) Show the B-tree after the integer 4 is inserted into the following B-tree, where MINIMUM=2 ans MAXIMUM=4.

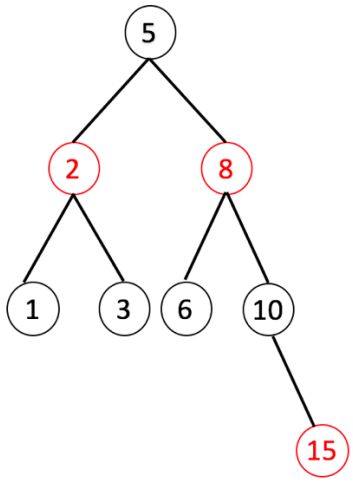


(38) Draw the B-tree that results when inserting R, Y, F, X, A, M, C, D, E, T, H, V, L, W, G (in that order) with minimum factor t = 3 and maximum is 2t. Show all the steps.
(39) Given the following 2-3 tree, draw the trees after each of the following operations,

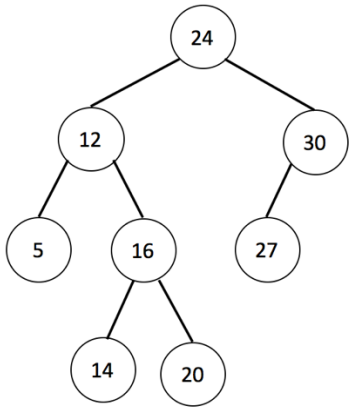
a. Insert(145)
b. Insert(105)
c. Insert(106)



(40) What is the maximum degree possible for a 2-3-4 tree? Draw the 2-3-4 tree that results after each of the integer keys 3, 1, 7, 2, 5, 4, 9, 6, 8, 12, 10, 11 are inserted, in that order, into an initially empty 2-3-4 tree. Clearly show the tree that results after any split that must be performed.
(41) Given the following red-black tree what will be the resulting tree after you insert 12. Clearly show the tree that results after the insertion (indicating the color of each node).



(42) Given the following AVL tree, what is the resulting AVL tree after inserting 15 and then 21, in that order. Also mention the types of imbalance occurred and make clear any rotation that must be performed.



Searching:

(43) An array is sorted in an increasing order and contains 64 data values.

(a) If sequential search is used, what is the maximum number of comparisons that are needed to search for a target in this array?

(b) If binary search is used, what is the maximum number of comparisons that are needed to search for a target in this array?

(c) If the target is in position 0 of the array, which search technique would find the data faster? Why?

(44) Consider the following double-hashing function for a hash table of size 100:

$H_1(k) = k \bmod 100$

$H_2(k) = 2 + (k \bmod 52)$

(a) For k = 75, how many elements of the hash table are examined in the worst case before an empty slot is found for k?

(b) Which of the following is the best replacement for $H_2(k)$ in the given problem for k =75.

(a) $H_2(k) = 1 + (k \bmod 52)$ (b) $H_2(k) = 5 + (k \bmod 52)$

(c) $H_2(k) = 4 + (k \bmod 52)$ (d) $H_2(k) = 27 + (k \bmod 52)$

(e) none of these answers

(45) Use the following hash table to answer questions (a) – (c). The hash table stores integer keys using a hash function h(k) = k mod 17. All keys were inserted without collisions.

INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KEY		69		88	4				59	94	27			47	31		16
HAS_BEEN_USED	F	T	F	T	T	F	F	F	T	T	T	F	F	T	T	F	T

(a) At what position will the key 60 be stored in the hash table using h(k) above if linear probing is used to resolve collisions?

(a) 2 (b) 5 (c) 11 (d) 15 (e) none of these answers

(b) At what position will the key 60 be stored in the original hash table if double hashing is used to resolve collisions, assuming $h_1(k) = h(k)$ and $h_2(k) = 2 + (k \bmod 11)$?

(a) 0 (b) 6 (c) 11 (d) 12 (e) none of these answers

(b) What is the load factor of the original hash table?

(a) 9/17 (b) 17/9 (c) 9 (d) 17 (e) none of these answers

(46) Given the following sequence of elements and a partially filled hash table with size 13 insert the elements into the hash table, in that order.

51, 27, 39, 44, 35, 46, 32

a. Use the **double hashing rule by Donald Knuth** for mapping the elements into the hash table.
Use the following hash functions for double hashing.
Hash1(k) = k % tableSize, [use this function for initial mapping, where % is modulus operator which finds the remainder after division]
Hash2(k) = 1 + (k % (tableSize - 2)), [use this function to find the next empty position when there is a collision]

If i is the index you have just examined (with a collision), then the next index to examine is (i + Hash2(k)) % tableSize

Why there are so many collisions even with double hashing in this case?

26		28					33	34				38
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]

b. Use **linear probing with divide hash function** for mapping the elements into the hash table. If you reach the end of the table wrap around.

26		28					33	34				38
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]

Sorting:

(47) Fill in the following table for the times to sort an array of n items. Use only big-O notation, and do not have any extraneous constants in your expressions.

Algorithm	Worst case	Average case
Selection sort		
Bubble sort		
Insertion sort		
Merge sort		
Quick sort without "median of three" pivot selection		
Quick sort with "median of three" pivot selection		
Heap sort		

(48) Trace how **selection** sort run on the following array of integers in an **increasing** order, showing the results after each run of the outer loop. Do not write a program.

10 21 8 18 14 5 70 1

(49) What is the main idea of Radix sort algorithm? Which sorting algorithm radix sort can use as a subroutine? Perform Radix sort on the following sequence of numbers. Use leading zeros for padding so that all the numbers have same number of digits. Show the sequence after sorting is done on each position.

{ 170, 45, 75, 90, 802, 24, 2, 66}

(50) Suppose we are sorting an array of eight integers using heapsort, and we have just finished one of the reheapifications downward. The array now looks like this:

6 4 5 1 2 7 8

How many reheapifications downward have been performed so far?

(a) 1
(b) 2
(c) 3 or 4
(d) 5 or 6

(51) Here is an array of ten integers:

5 3 8 9 1 7 0 2 6 4

Suppose we partition this array using quicksort's partition function and using 5 for the pivot. Draw the resulting array after the partition finishes.

(52) Complete the following code fragment which is used in merge sort to merge two sorted subarrays.

```
public static void merge(int[] A, int first, int n1, int n2){
      int[] temp = new int[___________];
      int copied = 0, copied1 = 0, copied2 = 0;
      while((copied1 < n1) && (copied2 < n2)){
            if(_____________________________________)
                  temp[copied++] = A[first + (copied1++)];
```

```
            else
                  temp[copied++] = _____________________________;
      }
      while(_____________)
            temp[copied++] = ____________________;
      for(int i = 0; i < copied; i++)
            A[first + i] = temp[i];
}
```

(53) The basic idea of bubble sorting is the following:

Starting from the beginning, traverse through the list, comparing
every two consecutive objects and checking if they are in wrong order. If there is a pair in wrong order, swap them and continue. Repeat from the beginning again until there is no swapping made while traversing the entire list of objects.

This implementation is more efficient than the obvious double for loop since it terminates if the list is sorted, and only looks at the unsorted elements through each for loop.

a) Simulate just one traversal of swapping in bubble sort with the following array {| 6 , 2 , 4 , 1 , 5 , 9 }
b) What is the best case and worst case for bubble sort in terms of order of complexity, Please give both cases based on the example in (a)?

(54) What is the time complexity of the following **reheapify** code?

```
private static void reheapify(int[] A, int heapsize) {
   int position = 0;
   int childPos;
   while (position*2 + 1 < heapsize) {
        childPos = position*2 + 1;
        if (childPos < heapsize-1 && A[childPos+1]
                        > A[childPos])
             childPos++;
        if (A[position] < A[childPos]){
             swap(A, position, childPos);
             position = childPos;
        }
        else return;
   }
}
```

(55) What is the time complexity of counting sort? What additional information do you need about the input to sort it using counting sort?

(56) Following is the counting sort algorithm:

```
countingSort(A, B, n, k)
   for i = 0 to k
        C[i] = 0;
```

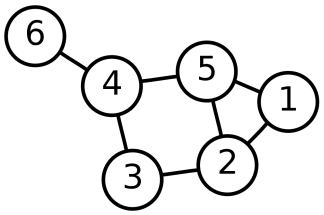
```
for j = 0 to n-1
     C[A[j]] += 1;
for i = 1 to k
     C[i] = C[i] + C[i-1];
for (j = n-1; j >= 0; j--)
     B[C[A[j]]-1] = A[j]
     C[A[j]] -= 1;
```

Consider the following array of unsorted integers.
1 2 0 3 1 1 0 2 4 2 5 5 3 6 2 1

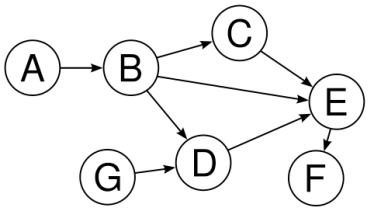
(a) Show the content of C after the third for loop.
(b) Show the content of B and C after the fourth for loop.

Graphs:

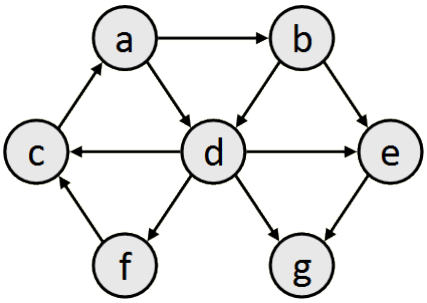
(57) For the following undirected graph. Show the adjacency matrix and adjacency list representation.



(58) For the following directed graph. Show the adjacency matrix and adjacency list representation.



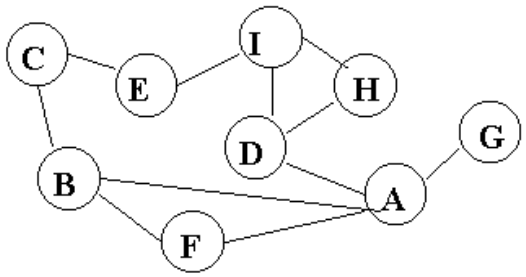
(59) Use the following directed graph for question 59(a) and 59(b).



Derive traversals starting at node a. At each node, neighbors should be visited in **alphabetical order**.

a. Write the depth first traversal of the graph above.
b. Write the breadth first traversal of the graph above.

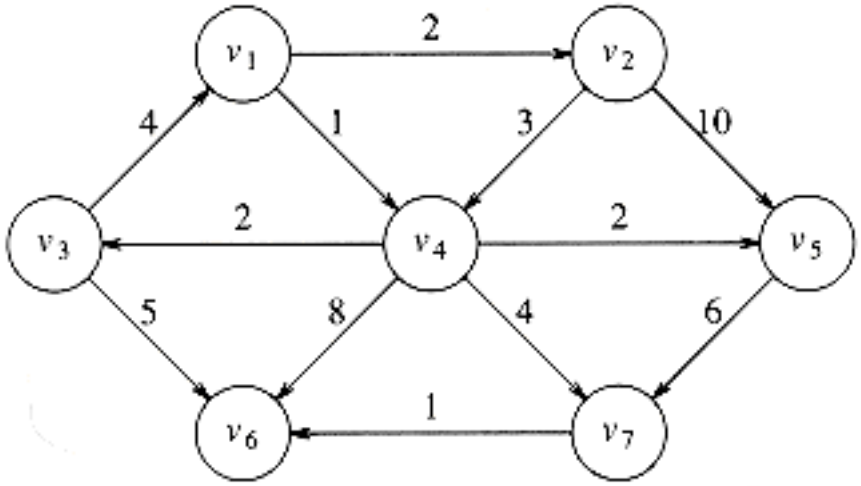
(60) Use the following undirected graph for question 60(a) and 60(b).



Derive traversals starting at node A. At each node, neighbors should be visited in **alphabetical order**.

a. Write the depth first traversal of the graph above.
b. Write the breadth first traversal of the graph above.

(61) Consider the following directed, weighted graph:



Step through Dijkstra's algorithm to calculate the single-source shortest paths from v_3 to every other vertex. Show your steps in the table below. Cross out old values and write in new ones, from left to right within each cell, as the algorithm proceeds. Finally, indicate the lowest-cost path from vertex v_3 to vertex v_5. Also, draw the final tree containing all the shortest paths from v_3 to all other nodes.

Answer:

Vertex	V_1	V_2	V_3	V_4	V_5	V_6	V_7
Cost							

Lowest-cost path from v_3 to v_5: _____

(62) The following matrix represents distances in miles between towns where direct routes exists.

	A	B	C	D	E	F
A	–	22	–	12	10	–
B	22	–	–	–	–	13
C	–	–	–	6	5	11
D	12	–	6	–	–	–
E	10	–	5	–	–	26
F	–	13	11	–	26	–

(a) Draw the graph.
(b) Use Dijkstra's algorithm to find the shortest route from A to F. Give the route and its length.