# Assignment 2

**Q1 (5 Points)**

Give a brief definition for the following:

i. Tree graph
In graph theory, a tree is an undirected graph in which any two vertices are connected by exactly one simple path. In other words, any connected graph without simple cycles is a tree.

ii. Adjacency List
In graph theory and computer science, an adjacency list representation of a graph is a collection of unordered lists, one for each vertex in the graph. Each list describes the set of neighbors of its vertex.

iii. Spanning Tree
In the mathematical field of graph theory, a spanning tree T of an undirected graph G is a sub graph that is a tree which includes all of the vertices of G, with minimum possible number of edges.

iv. Breadth-first search (BFS)
Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root and explores the neighbor nodes first, before moving to the next level neighbors.

v. Admissible heuristic
A heuristic function is said to be admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path.

**Q2 (5 Points)**

Arrange the following functions in increasing order of asymptotic growth:

• 5n5

The value of theta is $n^5$

• 0.33n

The value of theta is n i.e. 0.33 *n

• 5n3

The value of theta is $n^3$

• $n^2 \sqrt{n}$

The value of theta = $n^{2.5}$

• 5n

The value of theta = n i.e. 5*n

• log n

The value of theta = log n

• √n

The value of theta = $n^{0.5}$

The increasing order:

1. Log n
2. √n
3. $n^2$ √n
4. 0.33n
5. 5n
6. $5n^3$
7. $5n^5$

**Q3 (5 Points)**

Master Theorem: For the following recurrence, give an expression for the runtime T(n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.

T (n) = 8T (n/2) + n

a = 8, b = 2, c = 1, f(n) = n
k = $\log_b a = \log_2 8 = 3$
Since 3 > c i.e. it satisfies Master's Theorem
T (n) = $\Theta(n^3)$

**Q4 (5 Points)**

Master Theorem: For the following recurrence, give an expression for the runtime T(n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.

T (n) = n2 T (n/2) + log n

Since the value of a is not a constant, Master theorem condition is not satisfied

**Q5 (5 Points)**

Master Theorem: For the following recurrence, give an expression for the runtime T(n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.
T (n) = 4T (n/2) + $n^2$
a = 4, b = 2, c = 2, k =0,f(n) = $n^2$
k = $\log_b a = \log_2 4 = 2$
Since 2 >= c i.e. it satisfies Master's Theorem case 2
T (n) = $\Theta(n^c \log^{k+1} n) = \Theta(n^2 \log n)$

**Q6 (5 Points)**

Sort the list of integers below using Merge sort. Show your work. Write a recurrence relation for Merge sort.

(22, 13, 26, 1, 12, 27, 33, 15)

n = 8

22 13 26 1 12 27 33 15

22 13 26 1     12 27 33 15

Dividing

22 13     26 1     12 27     33 15

22 13 26 1 12 27 33 15

13 22     1 26     12 27     15 33

1 13 22 26     12 15 27 33

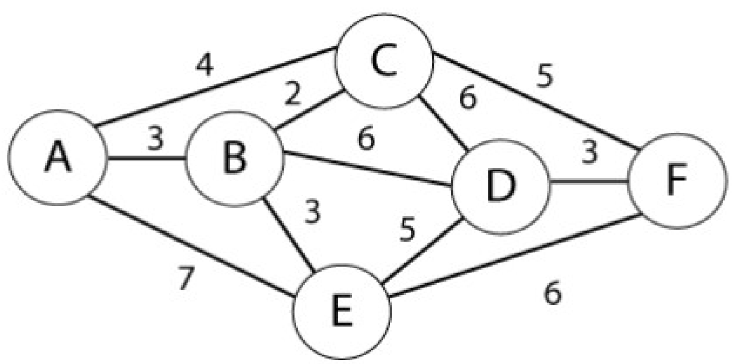
Merging

1 12 13 15 22 26 27 33 → Sorted list

Recursion Relation: T (n) = 2T (n/2) + n

**Q7 (5 Points)**

Use Kruskal's algorithm to find a minimum spanning tree for the connected weighted graph below:




1. Edges are sorted in ascending order:

| | | |
|---|---|---|
| E1: BC 2 | E5: AC 4 | E9: CD 6 |
| E2: AB 3 | E6: CF 5 | E10: AE 7 |
| E3: BE 3 | E7: DE 5 | |
| E4: DF 3 | E8: BD 6 | |

2. Then add the edges in spanning tree and if edges form cycle then it is discarded

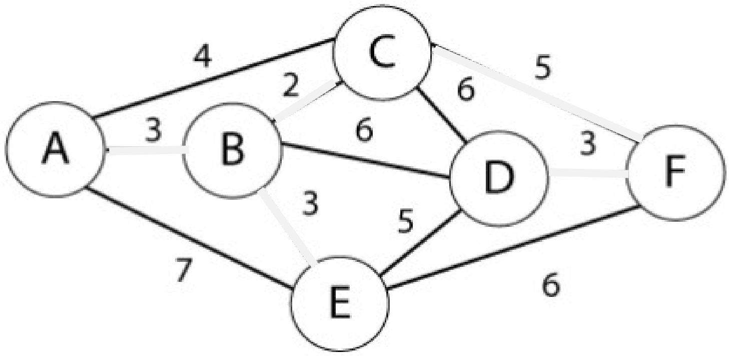
   According to the weights given above there are two trees which can be formed to get minimum cost

   Minimum spanning tree = G1 {B-C, A-B, B-E, D-F, C-F}

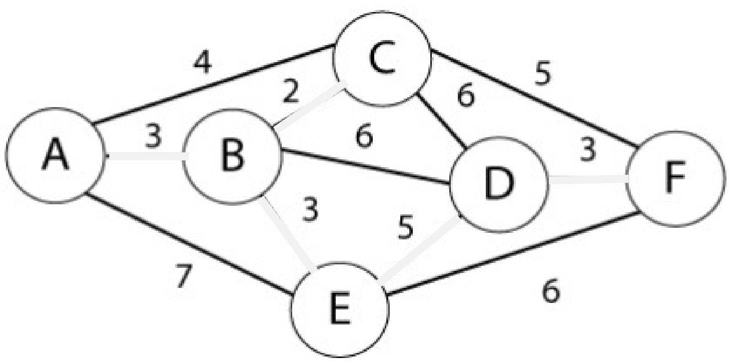
   = G2 {B-C, A-B, B-E, D-F, D -E}

Cost of tree G1 and G2 = 2+3+3+3+5 =16

G1:

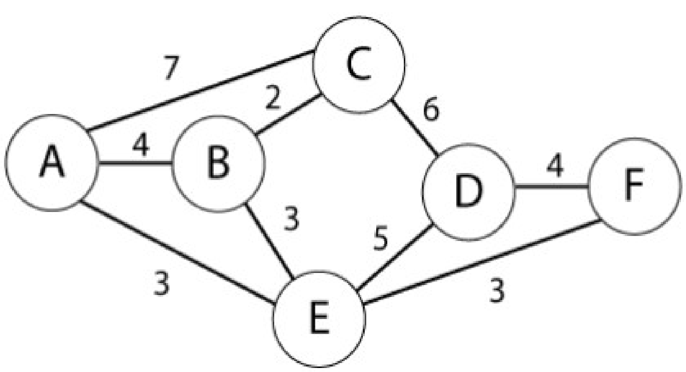



G2




**Q8 (5 Points)**

Use Prim's algorithm to find a minimum spanning tree for the connected weighted graph below. Show your work.



*Take the minimum edge of the cut-set each time.*

0: A S = {A}

1: A-E , A-B and A-C where A-E (3) is min-cut take A-E S = {A,E}

2: B-E, E-F and E-D where B-E (3) is min-cut take B-E S = {A, E, B}

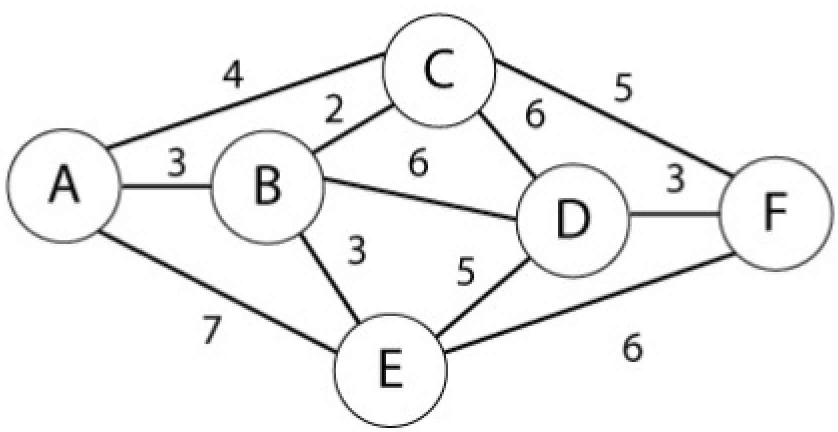
3. B-C (2) is min-cut B-C S = { A, B, C, E }
4. E-F and E-D where E-F (3) is min-cut take E-F S = {A, B, C, E, F}
5. D-F (4) is min-cut take D-F S = {A,B,C,E,F,D}

Done n-1 edges.

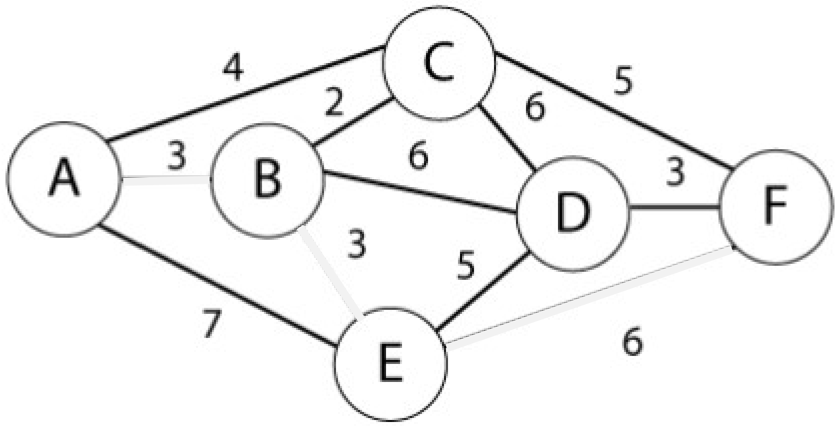
MST = {A-E, B-E ,E-F, D-F, C-B}

**Q9 (5 Points)**

Find shortest path from A to F in the graph below using Dijakshtra's algorithm. Show your steps.

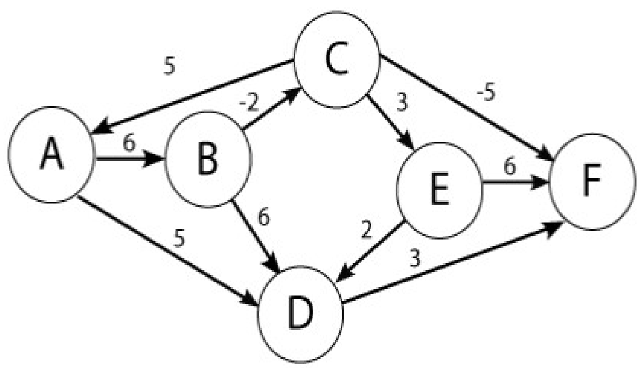



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A {A} | - | (3, A)* | (4, A) | INF | (7, A) | INF |
| B {A, B} | - | (3, A) | (4, A)* | (9, B) | (6, B) | INF |
| C {A, B, C} | - | (3, A) | (4, A) | (9, B) | (6, B)* | (9, C) |
| E {A, B, C, E} | - | (3, A) | (4, A) | (9, B)* | (6, B) | (9, C) |
| D {A, B, C, E, D} | - | (3, A) | (4, A) | (9, B) | (6, B) | (9, C)* |
| F {A, B, C, E, D, F} | - | (3, A) | (4, A) | (9, B) | (6, B) | (9, C) |




The shortest path from A to F {A, C, F} = 4 + 5 = 9

**Q10 (5 Points)**



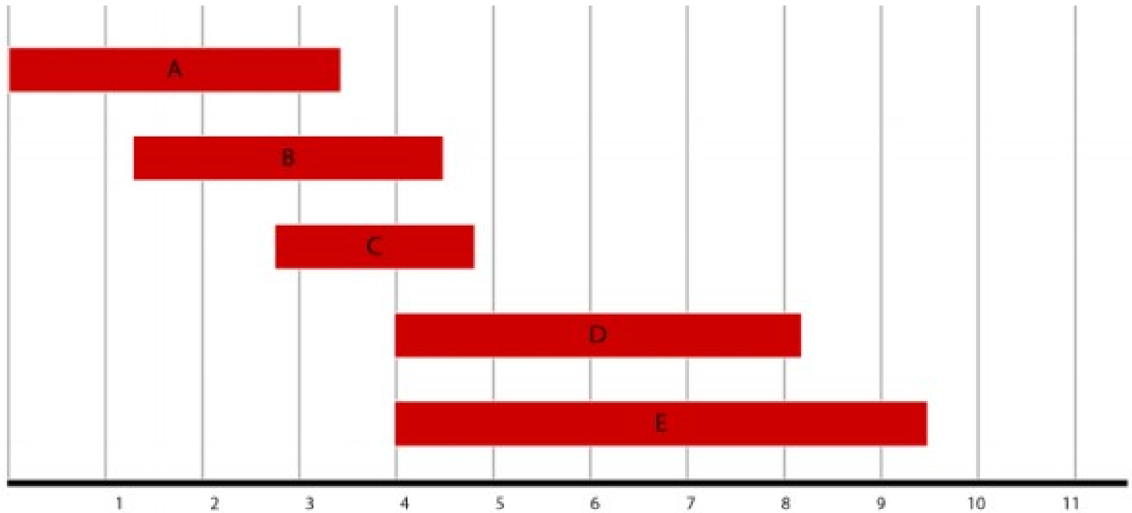

Use the Bellman-Ford algorithm to find the shortest path from node A to F in the weighted directed graph above. Show your work.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 0 | 0 | INF | INF | INF | INF | INF |
| 1 | 0 | 6 | INF | 5 | INF | INF |
| 2 | 0 | 6 | 4 | 5 | 7 | 13 |
| 3 | 0 | 6 | 4 | 5 | 7 | 8 |
| 4 | 0 | 6 | 4 | 5 | 7 | -1 |

The shortest path from A – F is A →B →C → F = -1

**Q11 (5 Points)**

Given the five intervals below, and their associated values; select a subset of non overlapping intervals with the maximum combined value. Use dynamic programming. Show your work.




| Interval | Value |
|---|---|
| A | 2 |
| B | 3 |
| C | 2 |
| D | 3 |
| E | 2 |

Solution :

| Interval | Value | Previous | Max |
|---|---|---|---|
| A | 2 | n/a | Max(4, 0) = 4 |
| B | 3 | n/a | Max(5, 0) = 5 |
| C | 2 | n/a | Max(5,0) = 5 |
| D | 3 | A | Max(9, 3+9) = 12 |
| E | 2 | B | Max(10, 2+10) = 12 |

**S = {E,B} or {A,D}**

**Q12 (5 Points)**

Given the weights and values of the four items in the table below, select a subset of items with the maximum combined value that will fit in a knapsack with a weight limit, W, of 6. Use dynamic programming. Show your work.

| Item $_i$ | Value $v_i$ | Weight $w_i$ |
|---|---|---|
| 1 | 3 | 4 |
| 2 | 2 | 3 |
| 3 | 4 | 2 |
| 4 | 4 | 3 |

Capacity of knapsack W=6

| | | | | | |
|---|---|---|---|---|---|
| **6** | 0 | 3 | 3 | 7 | 8 |
| **5** | 0 | 3 | 3 | 6 | 8 |
| **4** | 0 | 3 | 3 | 4 | 4 |
| **3** | 0 | 0 | 2 | 4 | 4 |
| **2** | 0 | 0 | 0 | 4 | 4 |
| **1** | 0 | 0 | 0 | 0 | 0 |
| **0** | 0 | 0 | 0 | 0 | 0 |
| | **0** | **1** | **2** | **3** | **4** |

We used items 2, 3 and 4 for a combined value of8 in the knapsack.

S={2,3,4}

**Q12 (40 Points) Search in Pacman**

## Question 12-1 (5 points):

Finding a Fixed Food Dot using Depth First Search

python pacman.py -l tinyMaze -p SearchAgent

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
[SearchAgent] using function tinyMazeSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 8 in 0.0 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python pacman.py -l mediumMaze -p SearchAgent

python pacman.py -l bigMaze -z .5 -p SearchAgent

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumMaze -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 130 in 0.0 seconds
Search nodes expanded: 144

(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l bigMaze -z .5 -p SearchAgent
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 390
```

Question 12-2 (5 points): Breadth First Search

python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 268
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win

(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.1 seconds
Search nodes expanded: 618
```

**Question 12-3 (5 points): Varying the Cost Function**

python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs

python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win

(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumDottedMaze -p StayEastSearchAgent
Path found with total cost of 1 in 0.1 seconds
Search nodes expanded: 186
Pacman emerges victorious! Score: 646
Average Score: 646.0
Scores:        646.0
Win Rate:      1/1 (1.00)
Record:        Win
```

python pacman.py -l mediumScaryMaze -p StayWestSearchAgent

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumScaryMaze -p StayWestSearchAgent
Path found with total cost of 68719479864 in 0.0 seconds
Search nodes expanded: 108
```

### Question 12-4 (5 points): A* search

python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.2 seconds
Search nodes expanded: 549
```

### Question 12-5 (5 points): Finding All the Corners

python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
[SearchAgent] using function bfs
[SearchAgent] using problem type CornersProblem
Path found with total cost of 28 in 0.0 seconds
Search nodes expanded: 410
Pacman emerges victorious! Score: 512
Average Score: 512.0
Scores:        512.0
Win Rate:      1/1 (1.00)
Record:        Win

(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
[SearchAgent] using function bfs
[SearchAgent] using problem type CornersProblem
Path found with total cost of 106 in 0.5 seconds
Search nodes expanded: 2381
```

### Question 12-6 (5 points): Corners Problem: Heuristic

python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
Path found with total cost of 106 in 0.2 seconds
Search nodes expanded: 901
```

### Question 12-7 (4 points): Eating All The Dots

python pacman.py -l trickySearch -p AStarFoodSearchAgent

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l trickySearch -p AStarFoodSearchAgent
Path found with total cost of 60 in 9.6 seconds
Search nodes expanded: 377
Pacman emerges victorious! Score: 570
Average Score: 570.0
Scores:        570.0
Win Rate:      1/1 (1.00)
Record:        Win
```

### Question 12-8 (5 points): Suboptimal Search

python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5

```
(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5
[SearchAgent] using function depthFirstSearch
[SearchAgent] using problem type PositionSearchProblem
Path found with cost 350.

(py27) C:\Users\mehta\Anaconda3\envs\py27\search>python pacman.py -l testSearch -p SearchAgent -afn=astar,prob=FoodSearchProblem,heuristic=foodHeuristic
[SearchAgent] using function astar and heuristic foodHeuristic
[SearchAgent] using problem type FoodSearchProblem
Path found with total cost of 7 in 0.0 seconds
Search nodes expanded: 7
Pacman emerges victorious! Score: 513
Average Score: 513.0
Scores:        513.0
Win Rate:      1/1 (1.00)
Record:        Win
```