National University of Computer and Emerging Sciences

Karachi Campus

Design and Analysis of Algorithms (CS2009)

Final Exam Paper Part-B

Total Time (Hrs): 3

Total Marks: 50

Total Questions: 8

Date: Dec 23rd 2024

Course Instructor(s)

Dr. Muhammad Atif Tahir, Dr. Nasir Uddin, Dr. Kamran, Dr. Fahad Sherwani, Dr. Farrukh Salim Shaikh, Ms. Anaum Hamid, Mr. Syed Faisal Ali, Mr. Sandesh Kumar, Mr. Minhal Raza, Mr. Abu Zohran

_____ _____ _____

Roll No Section Student Signature

Do not write below this line

Attempt all the questions.

CLO #1: To apply acquired knowledge to solve computing problems complexities and proofs

Question 3 [5 Marks]: A programmer is tasked with optimizing a computationally heavy data processing workflow. They are using System Z, which operates at a speed of 4 billion instructions per second (4 GIPS). The programmer has three software solutions to process a dataset of 1,000,000 records. Each solution has a different running time complexity as follows: The time complexity of Solution A is (n^2), the time complexity of Solution B is ($nlog_2\ n$) and the time complexity of Solution C is ($20\ n(log_2 n)^2$).

a) Calculate the number of instructions required to process the dataset for each solution. Show all your calculations. **[2.25 Marks]**

Task 1: Number of Instructions for Each Solution

1. Solution A: $T_A = n^2$

$$T_A = (10^6)^2 = 10^{12} \text{ instructions}$$

2. Solution B: $T_B = n \log_2 n$

First, compute $\log_2 n$ where $n = 10^6$:

$$\log_2(10^6) \approx \log_2 10 \times 6 \quad (\text{since } \log_2 10 \approx 3.32).$$

$$\log_2(10^6) \approx 3.32 \times 6 = 19.92 \,(\text{approximately } 20).$$

Now calculate T_B:

$$T_B = n \log_2 n = 10^6 \times 20 = 2 \times 10^7 \text{ instructions.}$$

3. Solution C: $T_C = 20n(\log_2 n)^2$

First, compute $(\log_2 n)^2$:

$$(\log_2 n)^2 = 20^2 = 400.$$

Now calculate T_C:

$$T_C = 20n(\log_2 n)^2 = 20 \times 10^6 \times 400.$$

$$T_C = 8 \times 10^9 \text{ instructions.}$$

b) Compute the execution time (in seconds) for each solution. **[1.5 Marks]**

Task 2: Execution Time for Each Solution

The formula to compute execution time is:

$$\text{Execution Time (seconds)} = \frac{\text{Number of Instructions}}{\text{System Speed (instructions per second)}}.$$

The system speed is 4×10^9 instructions per second.

1. Solution A:

$$\text{Time}_A = \frac{10^{12}}{4 \times 10^9} = 250 \text{ seconds.}$$

2. Solution B:

$$\text{Time}_B = \frac{2 \times 10^7}{4 \times 10^9} = 0.005 \text{ seconds.}$$

3. Solution C:

$$\text{Time}_C = \frac{8 \times 10^9}{4 \times 10^9} = 2 \text{ seconds.}$$

c) Compare the performance of the three solutions and explain the impact of running time complexity on execution time. **[1.25 Marks]**

Task 3: Performance Comparison and Impact of Complexity

Summary of Results:

Solution	Number of Instructions	Execution Time (seconds)
Solution A	10^{12}	250
Solution B	2×10^7	0.005
Solution C	8×10^9	2

Performance Comparison:

- **Solution A** has the worst performance with a **quadratic complexity** $O(n^2)$. The execution time of **250 seconds** is significantly higher, showing how inefficient quadratic algorithms become for large datasets.
- **Solution B** performs the best with **linearithmic complexity** $O(n \log n)$. It only takes **0.005 seconds**, demonstrating how well this complexity scales for large inputs.
- **Solution C** has a higher cost due to the $O(n(\log n)^2)$ complexity. While still much better than $O(n^2)$, its execution time of **2 seconds** reflects the impact of the squared logarithmic term.

CLO #1: To construct and analyze real world problems solutions using different algorithms design techniques. i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

Question 4 [5 Marks]: A shopping mall has introduced a reward system where customers can redeem reward coins of denominations1, 5, 6, and 8 to receive discounts. A customer has collected 20 coins in total and wants to redeem them in such a way that the total value equals 11, using the minimum number of coins.

a) Design an $O(n^2)$ dynamic programming based algorithm and write down its recurrence relation **[1 Mark]**

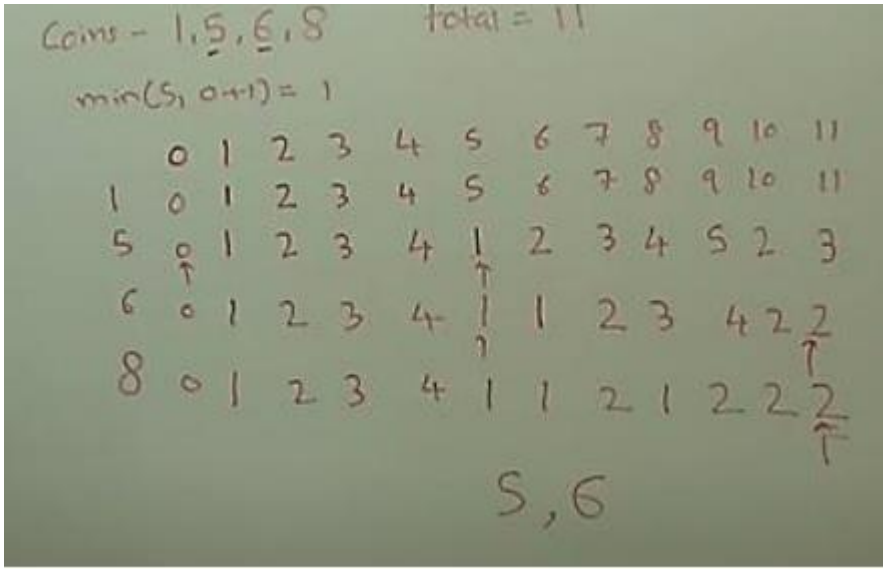
Solution:

```
function MIN_COINS(target_value):
    # Step 1: Initialize dp array where dp[i] is the minimum number of
coins to make i
    dp = array of size target_value + 1
    dp[0] = 0  # Base case: no coins needed for value 0

    # Step 2: Set all dp values for other indices to infinity initially
    for i = 1 to target_value:
        dp[i] = infinity

    # Step 3: Loop through all values from 1 to target_value
    for i = 1 to target_value:
        # Step 4: Try each denomination (1, 5, 6, 8)
        if i >= 1:
            dp[i] = min(dp[i], dp[i - 1] + 1)
        if i >= 5:
            dp[i] = min(dp[i], dp[i - 5] + 1)
        if i >= 6:
            dp[i] = min(dp[i], dp[i - 6] + 1)
        if i >= 8:
```

```
dp[i] = min(dp[i], dp[i - 8] + 1)

    # Step 5: Return the minimum number of coins for the target value
    return dp[target_value]
```

Recurrence Relation: dp[i] = min[dp[I - 1] + dp[i - 5] + dp[i – 6] + dp[i – 8]]
Time Complexity: O(n * m)

b) Use the dynamic programming approach to determine the minimum number of reward coins required to achieve a total value of 11. Provide the complete DP table for all amounts from 0 to 11. Identify the denominations of coins used in the optimal solution. **[3 Marks]**



c) What if you have to select maximum number of coins to make a total. **[1 Mark]**
Solution: If the objective changes from minimizing the number of coins used to maximizing the number of coins used to make a total, we need to adjust the dynamic programming approach accordingly. The goal now is to select the maximum number of coins that sum up to the target value, given the available coin denominations.

CLO #3: To evaluate generic algorithmic solutions such as sorting, searching and graphs applied to real-world problems

Question 5 [5 Marks]: A logistics company uses drones to deliver parcels between four warehouses: {A, B, C, D}. The energy cost for flying between these warehouses is shown in the graph in Figure 1. Some routes have negative energy costs due to favorable conditions like wind assistance or advanced energy saving technologies.

For instance, the drones are equipped with solar panels, and while they usually consume battery power during flights, certain routes benefit from optimal sunlight angles. On these routes, the solar panels charge the batteries instead of depleting them, resulting in a net energy gain. These scenarios are represented by negative energy costs in the graph. The company seeks answers to the following questions:

a) What is the minimum energy cost for transporting goods between all pairs of warehouses? **[2.5 Marks]**

	A	B	C	D
A	0	-1	-2	0
B	5	0	3	2
C	5	1	0	2
D	3	2	1	0

b) What are the most energy-efficient delivery routes between each pair of warehouses? **[2.5 Marks]**

	A	B	C	D
A	No Path	A -> C -> B	A -> C	A -> C -> D
B	B -> D -> A	No Path	B -> D -> A -> C	B -> D
C	C -> D -> A	C -> B	No Path	C -> D
D	D -> A	D -> A -> C -> B	D -> A -> C	No Path

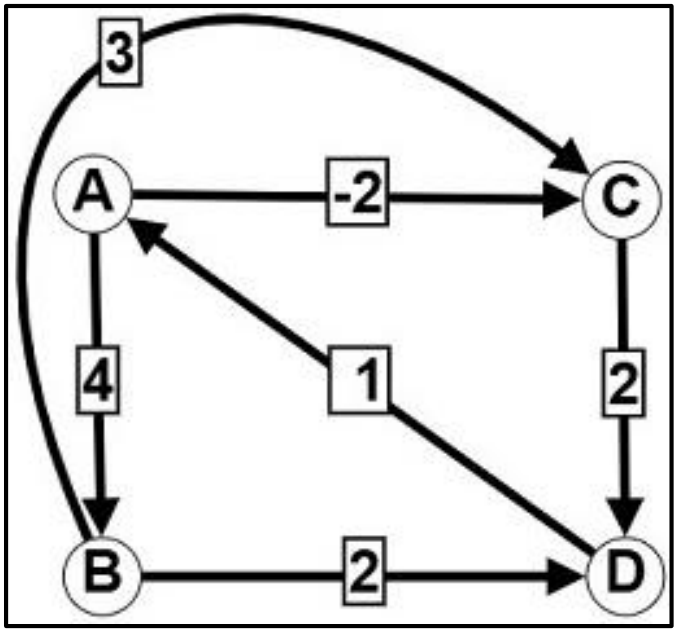


Figure 1 The directed graph showing the connections between the warehouses

CLO #4: To construct and analyze real world problems solutions using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

Question 6 [5 Marks]: Demand for multimedia, combining audio, video, and data streams over a network, is rapidly increasing. Some of the most popular uses of multimedia are real-time interactive applications such as desktop video and audio conferencing, collaborative engineering, shared white boards, transmission of university lectures to a remote audience, and animated simulations. With the advent of real-time interactive applications, **delay constraint (△)** is also an important objective along with minimizing bandwidth cost. Delay constraint means that a packet must be reached within that interval. Figure below shows an example graph. Each edge consists of 2 values (**first one is bandwidth cost** and **second one is delay**. Figure 1 shows the multicast graph where "**a**" is the source and **{b, c, d, f}** are the destinations. **{e, h, g}** are the intermediate nodes i.e. they can be used in multicast tree only if help in optimizing bandwidth cost under some delay constraint. Multicast communication means instead of sending packets to all destinations, packets will be send to only selected destinations. In the example below, main objective is to minimize the bandwidth cost under delay constraint. This problem is NP-Complete.

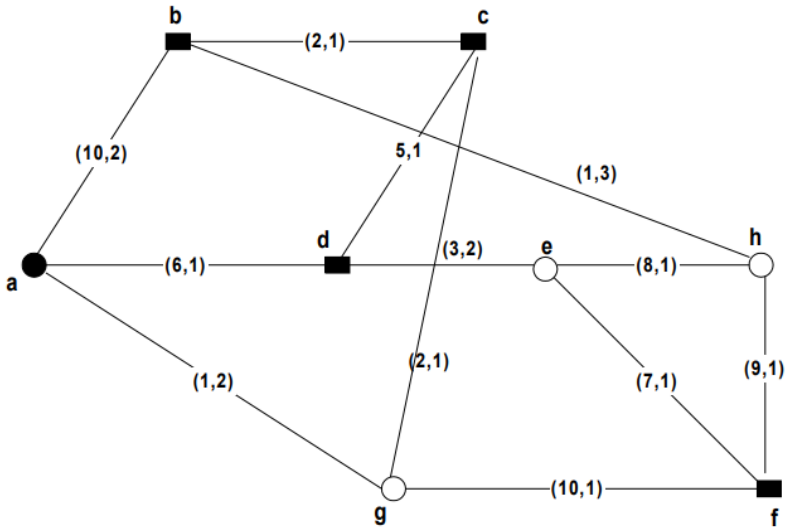


Figure 1: Multicast Graph

Algorithm proposed by V. P. Kompella, J. C. Pasquale and G. C. Polyzos, "Multicast Routing for Multimedia Communication," IEEE/ ACM Transaction on Networking is one of the most popular greedy algorithm to find the optimal bandwidth from source to selected destinations under some constraint. Algorithm 1 is the pseudocode of the algorithm and let's assume input graph of **Figure 1** with delay constraint △= 5. You are required to complete the following tasks

(a) What is the size of $\mathbf{V}'$ for input graph in Figure 1 and list the elements in $\mathbf{V}'$? Remember, $\mathbf{V}'$ is initialized in Algorithm 1. **[0.5 Mark]**

Solution: 5. {a,b,c,d,f} i.e. source and all multicast destinations.

Marking Scheme: Strict: 0 for wrong answer and 0.5 for correct answer

(b) Compute all pairs constrained least cost and delay paths i.e. $P_c\{v,w\}$ and $P_D\{v,w\}$. You can use compute shortest path from source "a" to other destinations using Dijkstra Algorithm using bandwidth cost. Rest of the values are pre-computed and given below. **[1.5 Marks]**

$P_c\{a,b\} = 5$	$P_D\{a,b\} = 4$	$\Pi(a,b) = \{a,g,c,b\}$
$P_c\{a,c\} = ?$	$P_D\{a,c\} = ?$	$\Pi(a,c) = ?$
$P_c\{a,d\} = ?$	$P_D\{a,d\} = ?$	$\Pi(a,d) = ?$
$P_c\{a,f\} = ?$	$P_D\{a,f\} = ?$	$\Pi(a,f) = ?$
$Pc(b,c) = 2$	$P_D(b,c) = 1$	$\Pi(b,c) = \{b,c\}$
$Pc(b,d) = 7$	$P_D(b,d) = 2$	$\Pi(b,d) = \{b,c,d\}$
$Pc(b,f) = 10$	$P_D(b,f) = 4$	$\Pi(b,f) = \{b,h,f\}$
$Pc(c,d) = 5$	$P_D(c,d) = 1$	$\Pi(c,d) = \{c,d\}$
$Pc(c,f) = 12$	$P_D(c,f) = 2$	$\Pi(c,f) = \{c,b,h,f\}$
$Pc(d,f) = 10$	$P_D(d,f) = 3$	$\Pi(d,f) = \{d,e,f\}$

Solution:

	a	b	c	D	E	f	g
	0	Inf	Inf	Inf	Inf	Inf	inf
		10	inf	6	Inf	Inf	**1**
		10	**3**	6	Inf	11	
		5		6	Inf	11	
				6	Inf	11	
					9	11	
						11	

	a	b	c	D	E	f	g
		c	g	A	D	g	A

$P_c\{a,b\} = 5$	$P_c\{a,b\} = 4$	$\Pi(a,b) = \{a,g,c,b\}$
$P_c\{a,c\} = 3$	**$P_c\{a,c\} = 3$**	**$\Pi(a,c) = \{a,g,c\}$**
$P_c\{a,d\} = 6$	**$P_c\{a,d\} = 1$**	**$\Pi(a,d) = \{a,d\}$**
$P_c\{a,f\} = 11$	**$P_c\{a,f\} = 3$**	**$\Pi(a,f) = \{a,g,f\}$**
Pc(b,c) = 2	Pc(b,c) = 1	$\Pi(b,c) = \{b,c\}$
Pc(b,d) = 7	Pc(b,d) = 2	$\Pi(b,d) = \{b,c,d\}$
Pc(b,f) = 10	Pc(b,f) = 4	$\Pi(b,f) = \{b,h,f\}$
Pc(c,d) = 5	Pc(c,d) = 1	$\Pi(c,d) = \{c,d\}$
Pc(c,f) = 12	Pc(c,f) = 2	$\Pi(c,f) = \{c,b,h,f\}$
Pc(d,f) = 10	Pc(d,f) = 3	$\Pi(d,f) = \{d,e,f\}$

Marking Scheme: Correct Solution: 1.5, Partial Solution 0-1.5, Incorrect: 0

(c) Floyd Warshall is another way to compute all pair shortest path. If no of multicast node (M) <<< no of non-multicast (NM) . Which one is preferred. Dijkstra algorithm run M times or Floyd Warshall. **[0.5 Mark]**

Solution: Since M<<<NM, Dijkstra algorithm is prefered as complexity would be V^2 as compared to V^3 of Floyd Warshall
Marking Scheme: Only Mention of Dikjstra = 0.25, Correct Reason = 0.5

(d) Compute $f_{c,d}(a,b)$, $f_{c,d}(a,c)$, $f_{c,d}(a,d)$, $f_{c,d}(a,f)$ from the equation mentioned in the pseudocode. **[1 Mark]**

Solution:

$$f_{CD}(a,c) = \frac{3}{5-(0+3)} = 1.5$$

$$f_{CD}(a,d) = \frac{6}{5-(0+1)} = 1.5$$

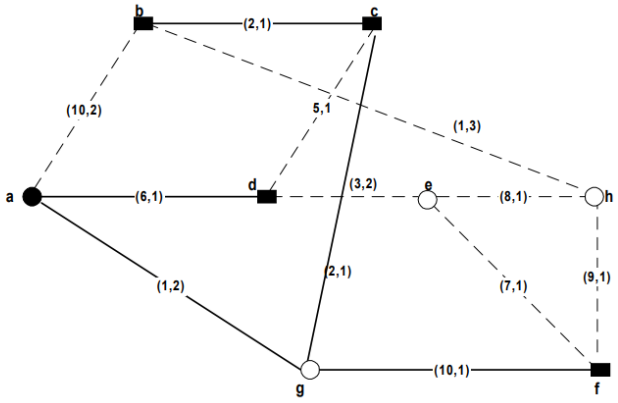
$$f_{CD}(a,f) = \frac{11}{5-(0+3)} = 5.5$$

$$f_{CD}(a,b) = \frac{5}{5-(0+4)} = 5$$

Marking Scheme: 0.25 Points each

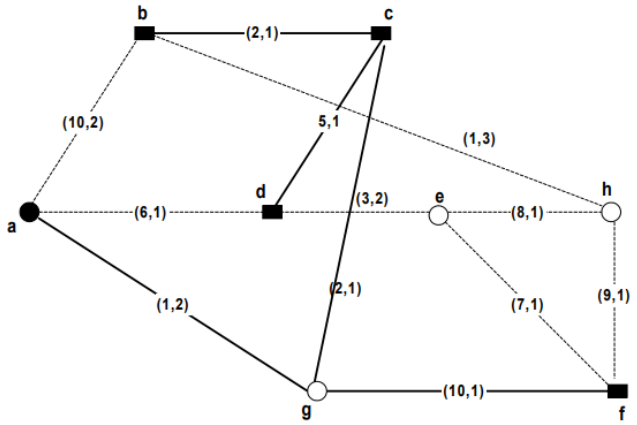
(e) Solution of Part (b) is enough to generate multicast tree with cost = 21 and △ = 5. Draw that multicast tree. **[0.5 Mark]**

Solution:



(f) Optimal multicast tree would have cost = 20 with △ = 5. Draw that multicast tree. No need to show steps. **[0.5 Mark]**

Solution:



1. /∗**CONSTRINT STEINER TREE ALGORITHM**
2. $*$ ***Let*** $\boldsymbol{G = (V, E)}$ ***describe the network topology***
3. $*$ $\boldsymbol{s}$ ***= source node***
4. $*$ $\boldsymbol{S}$ ***= multicast group***
5. $*\,\Delta =$ ***delay contraint */***
6. ***Multicast*** $\boldsymbol{(G(V, E), s, S, \Delta)}$
7. ***begin***
8. /$*$ ***Compute the cheapest constraint paths betwee all nodes in S U*** $\{s\}$ $*$/
9. $\boldsymbol{V' \leftarrow S\ U\ \{s\}}$
10. ***for each*** $\boldsymbol{v, w \in V'}$ ***do***
11. ***begin***
12. $\boldsymbol{P_C[v, w] \leftarrow}$ ***cost of cheapest constrained path from v to w***
13. $\boldsymbol{P_D[v, w] \leftarrow}$ ***path delay along the cheapest constrained path from v to w***
14. ***end***
15. /$*$ $\boldsymbol{C =}$ ***set of nodes already visited*** $*$/
16. /$*$ $\boldsymbol{P[v] =}$ ***path delay from s to v in the tree*** $*$/
17. /$*$ $\boldsymbol{T =}$ ***spanning tree on the closure graph*** $*$/
18. $\boldsymbol{C = \{s\}}$
19. $\boldsymbol{P[s] = 0}$
20. $\boldsymbol{T = \emptyset}$
21. /$*$ ***until all nodes in*** $\boldsymbol{V'}$ ***have been spanned*** $*$/
22. ***while*** $\boldsymbol{(C \neq V')}$ ***do***
23. ***begin***
24. $\boldsymbol{min = \infty}$
25. ***for each*** $\boldsymbol{v \in C}$ ***do***
26. ***begin***
27. ***for each*** $\boldsymbol{w \in V' \setminus C}$ ***do***
28. ***begin***
29. /$*$ ***if delay from s to w is within limits, consider***
30. $*$ $\boldsymbol{(v, w)}$***as a candidate and compute*** $\boldsymbol{f_s(v, w)}$
31. $*$ $\boldsymbol{f_s(v, w) = f_{CD}(v, w)}$ ***or*** $\boldsymbol{f_C(v, w)}$
32. $*$ $\boldsymbol{f_{CD}(v, w) = \frac{P_C(v,w)}{\Delta - [P(v) + P_D(v,w)]}}$
33. $*$ $\boldsymbol{f_C = P_C(v, w)}$
34. $*$ ***For further optimization use a better edge cost after conflict resolution*** $*$/
35. ***if*** $\boldsymbol{(f_s(v, w) <}$ **min**$\boldsymbol{)}$ ***then***

36. $\boldsymbol{begin}$
37. $\boldsymbol{nextedge} = (\boldsymbol{v}, \boldsymbol{w})$
38. $\mathbf{min} = \boldsymbol{f_s}(\boldsymbol{v}, \boldsymbol{w})$
39. $\boldsymbol{end}\ /* \boldsymbol{if} */$
40. $\boldsymbol{end}\ /* \boldsymbol{for} */$
41. $\boldsymbol{end}\ /* \boldsymbol{for} */$
42. $\boldsymbol{C} = \boldsymbol{C}\ \boldsymbol{U}\ \{\boldsymbol{w}\}$
43. $\boldsymbol{P}[\boldsymbol{w}] = \boldsymbol{P}[\boldsymbol{v}] + \boldsymbol{P_D}(\boldsymbol{v}, \boldsymbol{w})$
44. $\boldsymbol{T} = \boldsymbol{T}\ \boldsymbol{U}\ \{\boldsymbol{nextedge}\}$
45. $\boldsymbol{end}\ /* \boldsymbol{while} */$
46. $\boldsymbol{end}/* \boldsymbol{Multicast} */$

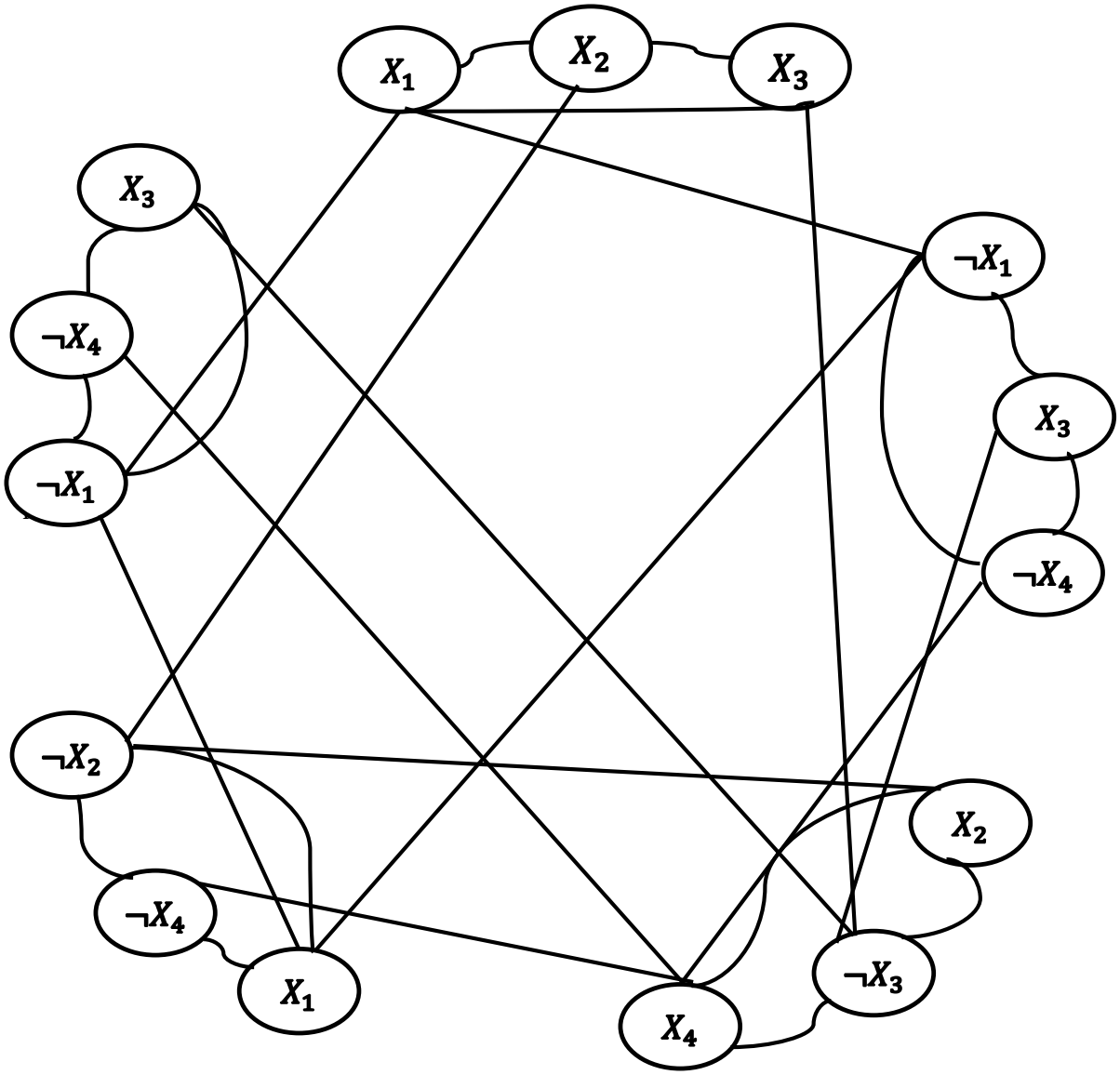
Algorithm 1: Multicast Tree Generation under Constraint △

CLO #4: To construct and analyze real world problems solutions using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

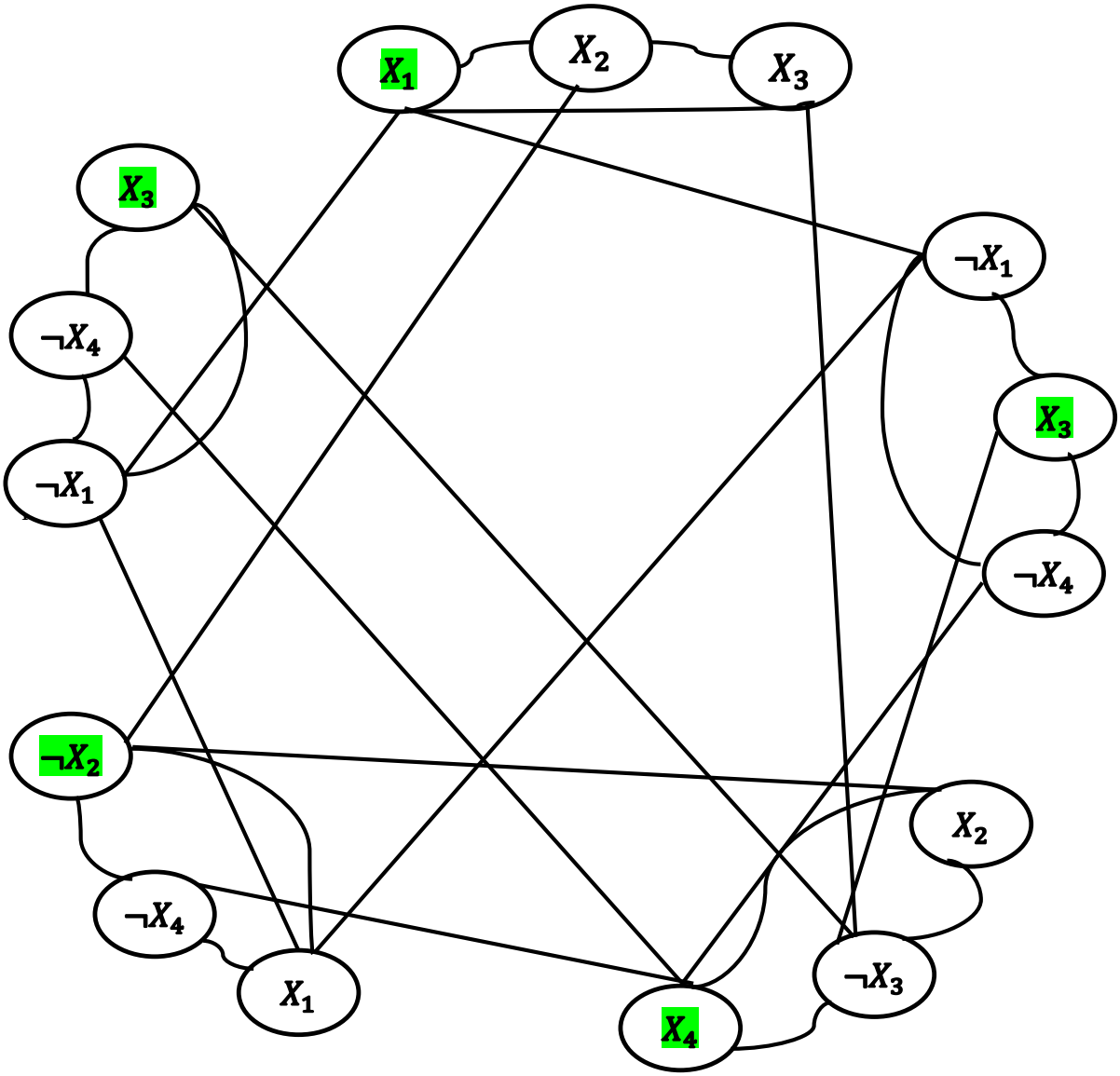
Question 7 [5 Marks]: Prove that $\varphi \in 3SAT \iff G_\varphi \in Clique \iff G_{\varphi'} \in Independent\ Set$, where $\boldsymbol{\varphi} = (\boldsymbol{X_1} \vee \boldsymbol{X_2} \vee \boldsymbol{X_3}) \wedge (\neg \boldsymbol{X_1} \vee \boldsymbol{X_3} \vee \neg \boldsymbol{X_4}) \wedge (\boldsymbol{X_2} \vee \neg \boldsymbol{X_3} \vee \boldsymbol{X_4}) \wedge (\neg \boldsymbol{X_2} \vee \neg \boldsymbol{X_4} \vee \boldsymbol{X_1}) \wedge (\boldsymbol{X_3} \vee \neg \boldsymbol{X_4} \vee \neg \boldsymbol{X_1})$.

Let ϕ be a 3-CNF formula with four variables and five clauses. Also specify a satisfying assignment with at least one literal false and corresponding clique G_φ. **Note: don't need to show all edges while reducing it to clique only show the edges which are contributing in formation of clique.**

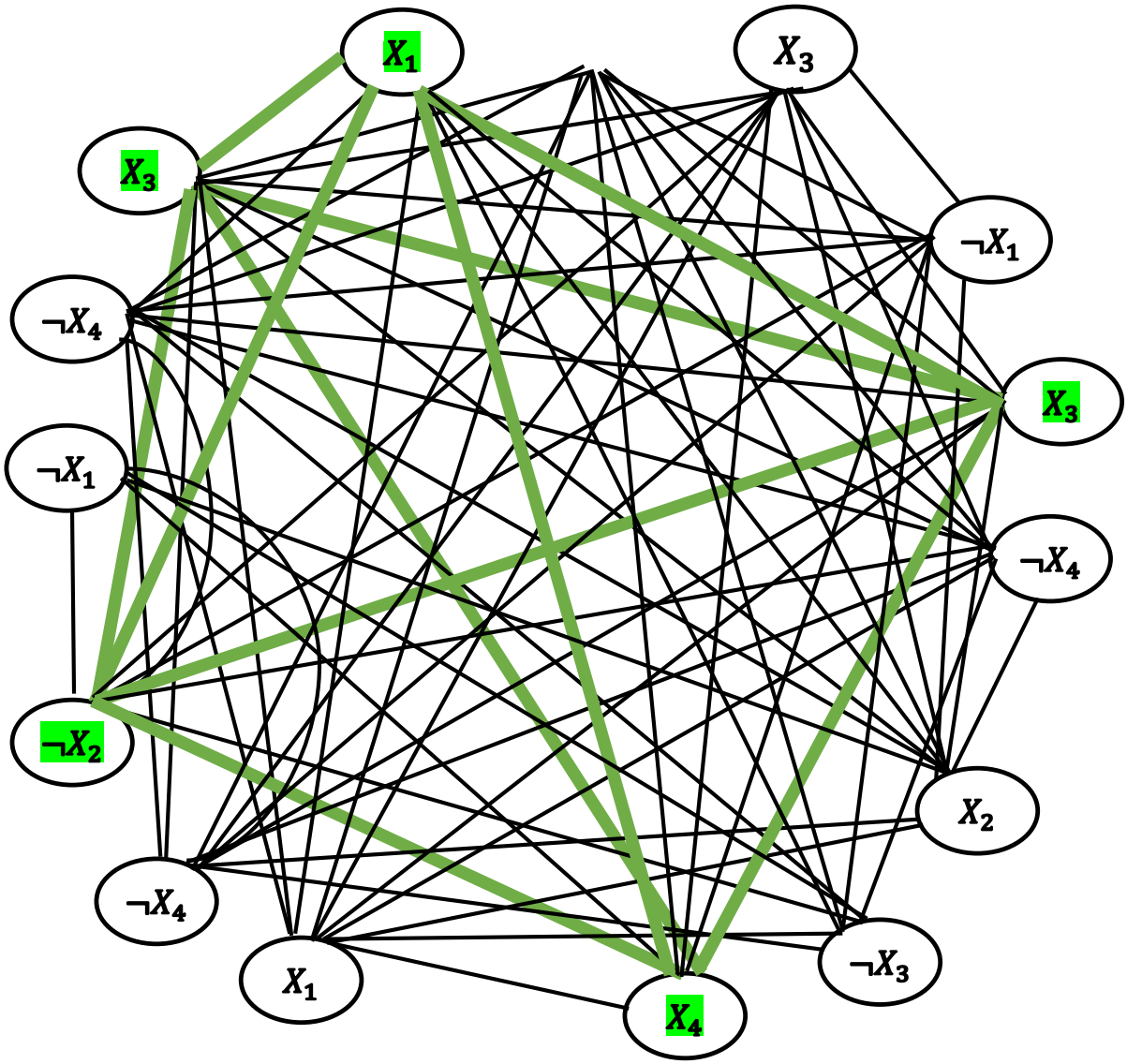
Solution: Satisfying assignment with at least one literal false is for $\varphi\ is\ X_1 = 1,\ X_2 = 0,\ X_3 = 1\ and\ X_4 = 1$
The following graph shows the Graph G corresponding to given 3-SAT formula φ.



The independent sat G_φ corresponding to satisfying assignment $X_1 = 1,\ X_2 = 0,\ X_3 = 1$ and $X_4 = 1$ are shown in green below.


X_1
X_2
X_3
X_3
$\neg X_1$
$\neg X_4$
X_3
$\neg X_1$
$\neg X_4$
$\neg X_2$
X_2
$\neg X_4$
$\neg X_3$
X_1
X_4

The Clique corresponding to is obtained by taking complement of the graph G. The clique is highlighted in green in the following figure



CLO#3 To evaluate geometric algorithmic solutions applied to solve real-life problems

Question 8 [5 Marks]: You are assisting LakeNav Corp., an autonomous boat company, in optimizing its navigation route around Lake Blue. The boat must inspect various critical locations in the lake area and return to the base station after completing its route. The company's goal is to minimize travel distance while ensuring all important locations are covered. You are given the coordinates (in kilometers) of each location as follows:

Base Station: (4, 5)
North Buoy: (2,10)
North-East Buoy: (7,9)
East Buoy: (9, 7)
South Buoy: (9, 3)
West Buoy: (1, 2)
Fishing Spot Alpha: (0, 5)
Birdwatching Platform: (6, 1)
Fishing Spot Beta: (3, 3)

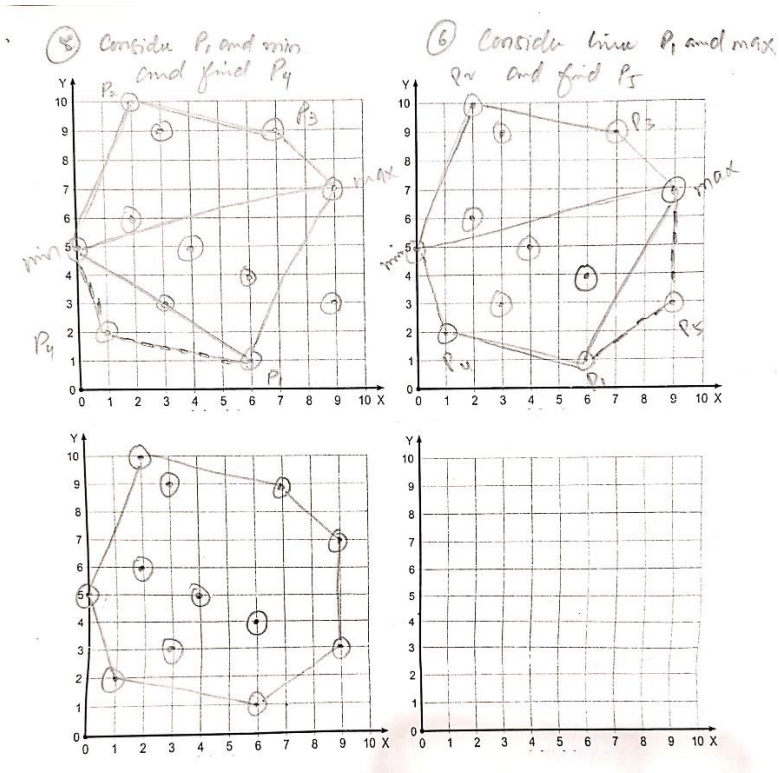
View Point = (6,4)
Rock Cluster Charlie: (2, 6)
Rock Cluster Delta: (3, 9)

a) Your task is to help LakeNav Corp. optimize the route by applying the quick elimination method. Plot all 12 points on a 2D plane, identifying which points are internal or dominated by others. Determine which points are essential for forming the convex hull **(use any algorithm)** and compute it accordingly. (Show all steps) **[2 Marks]**

Solution:

After the elimination process, the points that remain and form the convex hull are:

- **Fishing Spot Alpha:** (0, 5)
- **North Buoy:** (2, 10)
- **North-East Buoy:** (7, 9)
- **East Buoy:** (9, 7)
- **South Buoy:** (9, 3)
- **West Buoy:** (1, 2)
- **Birdwatching Platform:** (6, 1)



Final Convex Hull Points:

- (0, 5), (2, 10), (7, 9), (9, 7), (9, 3), (6, 1), and (1, 2).

(b) To calculate the perimeter, we'll measure the Euclidean distance between consecutive points on the hull.

1. (0, 5) to (2, 10):

$$\text{Distance} = \sqrt{(2-0)^2 + (10-5)^2} = \sqrt{4+25} = \sqrt{29} \approx 5.39$$

2. (2, 10) to (7, 9):

$$\text{Distance} = \sqrt{(7-2)^2 + (9-10)^2} = \sqrt{25+1} = \sqrt{26} \approx 5.10$$

3. (7, 9) to (9, 7):

$$\text{Distance} = \sqrt{(9-7)^2 + (7-9)^2} = \sqrt{4+4} = \sqrt{8} \approx 2.83$$

4. (9, 7) to (9, 3):

$$\text{Distance} = \sqrt{(9-9)^2 + (3-7)^2} = \sqrt{0+16} = 4$$

5. (9, 3) to (6, 1):

$$\text{Distance} = \sqrt{(6-9)^2 + (1-3)^2} = \sqrt{9+4} = \sqrt{13} \approx 3.61$$

6. (6, 1) to (1, 2):

$$\text{Distance} = \sqrt{(1-6)^2 + (2-1)^2} = \sqrt{25+1} = \sqrt{26} \approx 5.10$$

7. (1, 2) to (0, 5):

$$\text{Distance} = \sqrt{(0-1)^2 + (5-2)^2} = \sqrt{1+9} = \sqrt{10} \approx 3.16$$

Total Perimeter:

$$\text{Perimeter} = 5.39 + 5.10 + 2.83 + 4 + 3.61 + 5.10 + 3.16 \approx 29.19 \text{ kilometers}$$

C) If the boat must avoid Rock Cluster Charlie and Rock Cluster Delta, how would you modify the route based on the convex hull? **[1 Mark]**

Solution: Since Rock Cluster Charlie (2, 6) and Rock Cluster Delta (3, 9) are internal points that were eliminated, they do not interfere with the convex hull path itself. The boat's route, following the perimeter of the convex hull, already avoids these points. If additional clearance is necessary around these locations, the boat could take a slightly wider path around the hull points near these rocks (such as the path between (2, 10) and (7, 9)), but the primary convex hull would remain the same.

CLO #1: To apply acquired knowledge to solve computing problems complexities and proofs

Question 9 [5 Marks]: Briefly answer the following questions

a. Why it is important to find approximate solutions for NP Complete Problems? **[1 Mark]**
Approximation algorithms provide a way to obtain near-optimal solutions efficiently, allowing for practical applications even when exact solutions are computationally expensive or infeasible

b. Write down any one real life application of Maximum sub-array. **[1 Mark]**
financial analysis.

c. Suppose you want to lay pipelines in a building such that minimum pipe and other materials are consumed covering every desired location. Which technique from algorithms course will you employ and why? **[1 Mark]**

The problem is essentially about connecting all the desired locations (nodes) in the building with the minimum total cost (representing the minimum pipe length and material usage). This is a classic **MST problem**, where you want to connect all nodes (locations) with the minimum total edge (pipe) weight.

d. Differentiate between decision and optimization based problems? **[1 Mark]**
A decision problem is a problem where the goal is to answer a **yes** or **no** question whereas an optimization problem seeks to find the best solution from a set of possible solutions, usually maximizing or minimizing some objective function.

e. Does there any chance exist of solving an NP-Hard problem by deterministic polynomial time algorithm? **[1 Mark]**
No, there is no known deterministic polynomial-time algorithm for solving NP-Hard problems

CLO #4: To construct and analyze real world problems solutions using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

Question 10 [5 Marks]: Write pseudocode for a divide-and-conquer algorithm **MATRIX-ADD-RECURSIVE** that sums two n x n matrices A and B by partitioning each of them into **four $n/2$ x $n/2$** submatrices and then recursively summing corresponding pairs of sub-matrices. Assume that matrix partitioning uses $\theta(1)$-time index calculations.

a. Write a recurrence for the worst-case running time of MATRIX-ADD-RECURSIVE, and solve your recurrence. **[2.5 Marks]**
b. What happens if you use $\theta(n^2)$-time copying to implement the partitioning instead of index calculations? **[2.5 Marks]**

SOLUTION:

PSEUDOCODE:

```
MATRIX-ADD-RECURSIVE(A, B):

    Input: Two n x n matrices A and B

    Output: Matrix C = A + B


    If n = 1:

        Return A[1][1] + B[1][1]   -> O(1)


    Else:

        // Partition A, B into four n/2 x n/2 submatrices

        A11, A12, A21, A22 = Partition(A)  // A is partitioned into four submatrices -

        B11, B12, B21, B22 = Partition(B)  // B is partitioned into four submatrices


        // Recursively add corresponding submatrices
```

```
C11 = MATRIX-ADD-RECURSIVE(A11, B11) -> O(n/2)
C12 = MATRIX-ADD-RECURSIVE(A12, B12) -> O(n/2)
C21 = MATRIX-ADD-RECURSIVE(A21, B21) -> O(n/2)
C22 = MATRIX-ADD-RECURSIVE(A22, B22) -> O(n/2)

// Combine the submatrices into the result matrix C
C = Combine(C11, C12, C21, C22) -> O(1)
Return C
```

(a) Recurrence for the Worst-Case Running Time

Matrix is divided into four n/2 * n/2 submatrices and a constant time work for partitioning (index calculations) and combining (partitioning takes $\theta(1)$ time as assumed).

$$T(n) = 4T\left(\frac{n}{2}\right) + O(1)$$

Solving the Recurrence using the Master Theorem

The recurrence is of the form: $T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$ where a = 4, b = 2, and d = 0. Now compare a with b^d.

a = 4, $b^d = 2^0 = 1$. Now compare a with b^d

- If a > b^d the time complexity is $T(n) = O(n^{log b^a})$. Here we have 4 > 1, so we use the case where a > b^d and time complexity is $T(n) = O(n^{log 2^4}) = O(n^2)$

(b) What Happens if Partitioning Takes $O(n^2)$ Time?

If the partitioning step takes $O(n^2)$ time (i.e., if copying the submatrices instead of just calculating indices requires $\boldsymbol{O(n^2)}$ time), the recurrence becomes:

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n^2)$$