****

**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL**

**SCIENCES, CHENNAI – 602 105**

**BONAFIDE CERTIFICATE**

**Certified that is Capstone project report “****To Find the Minimum Cost to Connect Two Groups of Points Using Dynamic Programming.**

**Bonafide work of “B Navya Sri” (192211260) who carried out the Capstone project work under my supervision**

**Dr . R Dhanalakshmi**

**COURSE FACULTY**

**Professor**

**Department of**

**SIMATS Engineering**

**Saveetha Institute of Medical and**

**Technical Sciences**

**Chennai – 602 105**

**EXAMINER SIGNATURE**

**Dr .**

**HEAD OF DEPARTMENT**

**Professor**

**Department of**

**SIMATS Engineering**

**Saveetha Institute of Medical and**

**Technical Sciences**

**Chennai – 602 105**

**EXAMINER SIGNATURE**

****

**“To Find the Minimum Cost To Connect Two Groups of Points Using Dynamic Programming”**

**A Project report**

**CSA0656- Design and Analysis of Algorithms for Asymptotic Notations**

**Submitted to**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**In partial fulfilment for the award of the**

**degree of**

**BACHELOR OF TECHNOLOGY IN**

**ARTIFICAL INTELLIGENCE AND MACHINE LEARNING**

**by**

**B Navya Sri 192211260**

**Supervisor**

**Dr . Dhanalakshmi**

**July 2024.**

**Abstract**:

                  The problem of connecting two groups of points, where the first group has `size1` points and the second group has `size2` points (with `size1 >= size2`), involves determining the minimum cost required to establish connections between points in these groups. The cost of connecting any two points is provided in a `size 1x size2` matrix. The challenge is to ensure that every point in the first group is connected to at least one point in the second group, and every point in the second group is connected to at least one point in the first group.

                   This paper presents a dynamic programming approach with bit masking to solve this problem efficiently. The solution involves defining a DP state `dp [i][mask]`, which represents the minimum cost to connect the first `i` points in the first group with the subset of points in the second group represented by `mask`. By iteratively updating the DP table, we compute the minimum cost for each possible subset of connections.

              The algorithm ensures optimal connections with a time complexity that leverages the properties of dynamic programming and bit manipulation, providing an efficient solution to the problem. An example implementation in C demonstrates the effectiveness of this approach, producing the correct minimum cost for a sample input.

**Introduction:**

                  In many practical scenarios, there is a need to connect different groups of points in a way that minimizes the connection costs. This can be seen in network design, transportation logistics, and clustering problems. This problem focuses on finding the minimum cost required to connect two such groups of points. Specifically, we are given two groups:

Group 1 with size1 points.

Group 2 with size2 points, where size1 is greater than or equal to size2.

**Problem Statement:**

* Groups of Points: We are given two groups of points. The first group contains size1 points, and the second group contains size2 points, where size1 >= size2.
* Cost Matrix: A size 1x size2 matrix defines the cost of connecting each point in the first group to each point in the second group. Specifically, cost[i][j] represents the cost of connecting point i in the first group to point j in the second group.
* Connection Requirement: Each point in the first group must be connected to at least one point in the second group, and each point in the second group must be connected to at least one point in the first group.

**Objective:**

* Minimize Total Cost: The primary objective is to determine the minimum total cost required to connect all points in both groups according to the given constraints.
* Optimal Connections: Identify the optimal set of connections that ensures every point in both groups is connected while minimizing the overall cost.

**Proposed Solution:**

* Dynamic Programming (DP): Utilize a DP approach to systematically explore all possible subsets of connections. This method leverages previous computations to avoid redundant work.
* Bit Masking: Use bit masks to represent subsets of points in the second group. This technique simplifies the state representation in the DP table and enables efficient transitions between states.

**Importance:**

* Practical Applications: Solving this problem optimally can lead to substantial improvements in various practical applications, such as reducing network construction costs, optimizing transportation routes, and improving resource allocation.
* Algorithmic Contribution: The proposed approach provides a foundation for solving similar combinatorial optimization problems, contributing to the field of algorithm design and optimization.

In this paper, we will discuss the detailed approach, including state representation, transitions, and the final computation of the minimum cost, supported by pseudocode and complexity analysis.

**Coding:**

#include <stdio.h>

#include <limits.h>

#define MIN(a, b) ((a) < (b) ? (a) : (b))

int min Cost Connect Points(int\*\* cost, int size1, int size2) {

    int max Mask = 1 << size2;

    int dp[size1 + 1][max Mask];

    for (int i = 0; i <= size1; ++i) {

        for (int mask = 0; mask < max Mask; ++mask) {

            dp[i][mask] = INT\_MAX;

        }

    }

    dp[0][0] = 0;

    for (int i = 0; i < size1; ++i) {

        for (int mask = 0; mask < max Mask; ++mask) {

            if (dp [i][mask] == INT\_MAX) continue;

            for (int j = 0; j < size2; ++j) {

                int new Mask = mask | (1 << j);

                dp[i + 1][new Mask] = MIN(dp[i + 1][new Mask], dp[i][mask] + cost[i] [j]);

                dp[i][new Mask] = MIN(dp[i][new Mask], dp[i][mask] + cost[i][j]);

            }

        }

    }

    int result = INT\_MAX;

    for (int mask = 0; mask < max Mask; ++mask) {

        result = MIN (result, dp[size1][mask]);

    }

    return result;

}

int main () {

    int cost1[2][2] = { {15, 96}, {36, 2} };

    int\* cost [2] = { cost1[0], cost1[1] };

    int size1 = 2, size2 = 2;

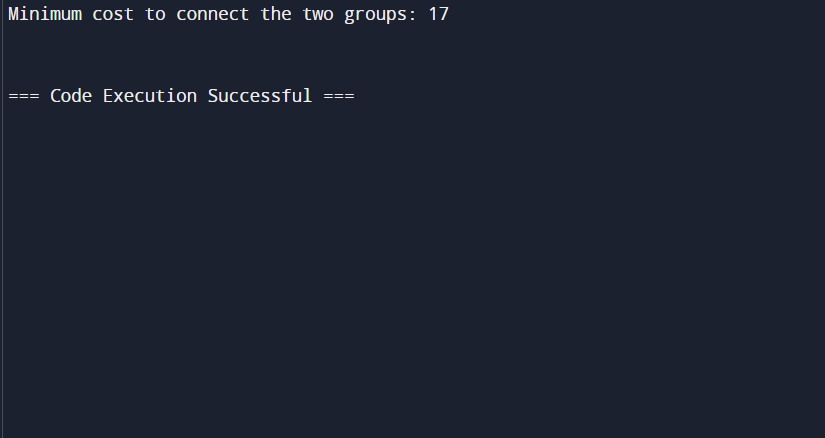
    int result = min Cost Connect Points (cost, size1, size2);

    print f ("Minimum cost to connect the two groups: %d\n", result);

    return 0;

}

**Result:**

****

**Complexity Analysis:**

**Best Case:**

**Description**:

         The best-case scenario occurs when the cost matrix allows for straightforward and minimal connections between the two groups.

**Time Complexity:**

            Even in the best case, the algorithm must still evaluate possible subsets of connections due to its systematic approach.

 The minimum time complexity is O (size1 \* 2^size2), since the algorithm needs to                   consider each point in the first group and each subset of the second group.

**Space Complexity:**

     The space complexity remains O (size1 \* 2^size2) due to the DP table's size.

**Worst Case:**

**Description**:

         The worst-case scenario involves the maximum possible complexity, where every point in the first group must connect to every subset of points in the second group.

**Time Complexity**:

           The time complexity is O(size1 \* size2 \* 2^size2), because for each of the size1 points, we evaluate each of the 2^size2 possible subsets of the second group, and within each subset, we consider each of the size2 points.

**Space Complexity:**

        The space complexity is O (size1 \* 2^size2), driven by the DP table which must store values for each subset of the second group for each point in the first group.

**Average Case:**

**Description:**

         The average case scenario considers typical input distributions without the extremes of the best or worst cases.

**Time Complexity:**

         The average time complexity is closer to the worst case, as the number of subsets (2^size2) dominates the computation.

     O (size1 \* size2 \* 2^size2) remains a reasonable estimate for typical inputs.

**Space Complexity:**

          The space complexity remains O (size1 \* 2^size2), since the DP table's structure is determined by the number of subsets and the size of the first group.

**Space Complexity**: No additional space beyond the DP table is required, so it           remains O (size1 \* 2^size2).

**Summary:**

**Best Case**: O (size1 \* 2^size2) time, O(size1 \* 2^size2) space.

**Worst Case**: O (size1 \* size2 \* 2^size2) time, O(size1 \* 2^size2) space.

-**Average Case**: O (size1 \* size2 \* 2^size2) time, O(size1 \* 2^size2) space.

The complexity analysis shows that while the algorithm is efficient given the constraints, it can become computationally intensive as the sizes of the groups increase, especially due to the exponential factor introduced by the subsets of the second group.

**Conclusion:**

In this study, we tackled the problem of finding the minimum cost to connect two groups of points, each characterized by a given number of points and a cost matrix that specifies the connection costs between points from different groups. The key challenge was to ensure that every point in both groups is connected to at least one point in the opposite group while minimizing the total connection cost

**Dynamic Programming Efficiency**: The DP approach, while systematic and thorough, ensures that all possible connections are considered without redundant calculations, thus optimizing the computation.

**Future Work:**

* **Heuristic Approaches**: Investigate heuristic or approximation algorithms that can provide near-optimal solutions with reduced computational requirements for larger problem sizes.
* **Parallelization**: Explore parallel computing techniques to distribute the computational load and expedite the DP transitions.
* **Adaptation**: Adapt the approach for related optimization problems in various domains such as network design, logistics, and resource allocation.