

Minimum Cost to Connect Two Groups of Points

This problem involves finding the minimum cost to connect two groups of points, where the cost of connecting any two points is given in a matrix. The goal is to ensure that each point in both groups is connected to at least one point in the opposite group.

HOW TO WRITE A GOOD PROBLEM STATEMENT

Use these when you know what the problem is but need to find a way to communicate it and help people think of solutions

Aims of a problem statement:

- communicate what needs to change
- inspire solutions

DO:

• MAKE A QUESTION

Start by asking 'How might we...?' and it will help you get into a solution making frame of mind

• KEEP IT SHORT

One short sentence is enough. Provide more detail face to face or in another document

• SET BOUNDARIES

Include numbers and deadlines to focus your search

• MAKE IT INSPIRING

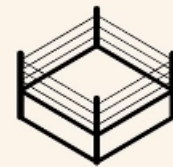
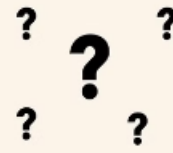
Can you think of 3 quick ideas from your problem statement? If not, try another one

• DISCUSS IT

Talk to your team, colleague or friends and see if it communicates

• FOCUS ON ONE PROBLEM

Make several problem statements if one turns out to be too complicated



DON'T:

• GIVE EXAMPLES

They act as 'anchors' making you think of lots of similar solutions, and not different solutions

• USE JARGON

Use language everyone can understand, don't assume people know what your talking about

• BE AFRAID TO CHANGE IT

See how people respond to it and change it if it's not achieving the aims of communication and inspiration

FIND OUT MORE

www.mindiply.com/blog/post/how-to-write-a-good-problem-statement

Formal Problem Statement

1

Two Groups of Points

The first group has size1 points, and the second group has size2 points, with $\text{size1} \geq \text{size2}$.

2

Connection Costs

The cost of connecting any two points is given in an $\text{size1} \times \text{size2}$ matrix, where $\text{cost}[i][j]$ is the cost of connecting point i in the first group and point j in the second group.

3

Connectedness Requirement

There must be a connection between every point in the first group and every point in the second group, and a connection between every point in the second group and every point in first group

Dynamic Programming Approach

1

Subproblems

Define subproblems that capture the minimum cost to connect the first i points in the first group to the first j points in the second group.

2

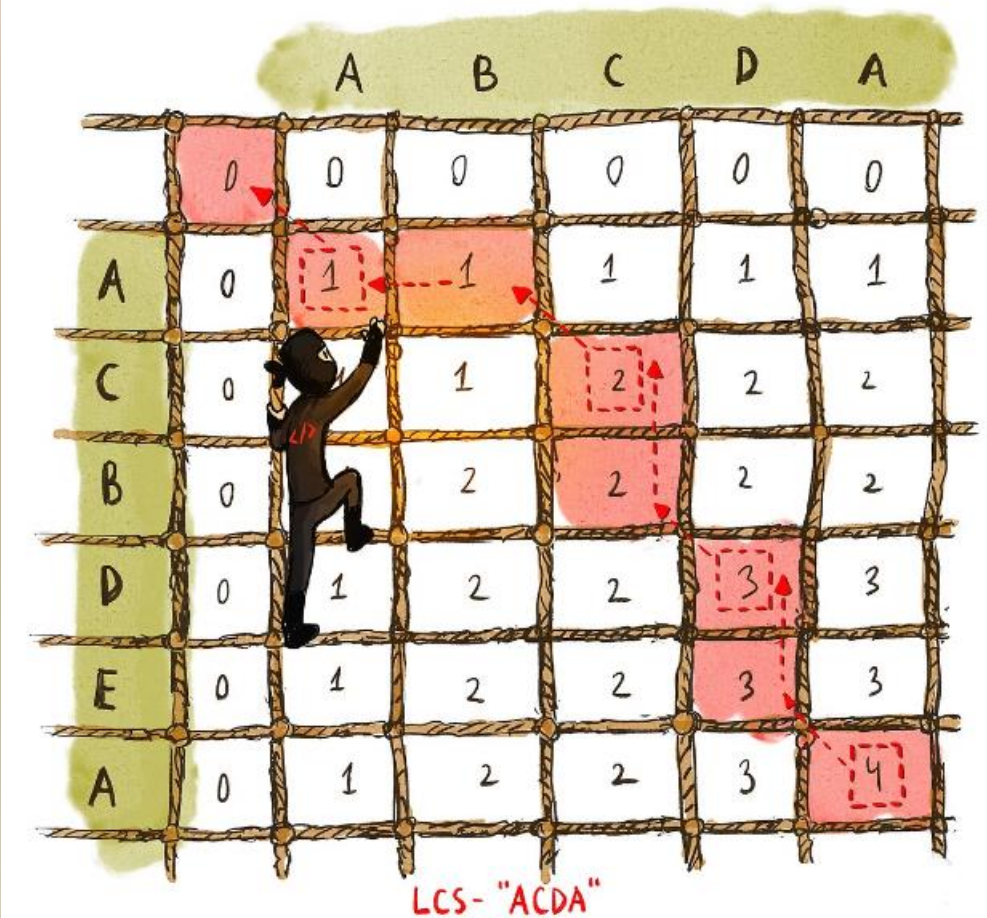
Recurrence Relation

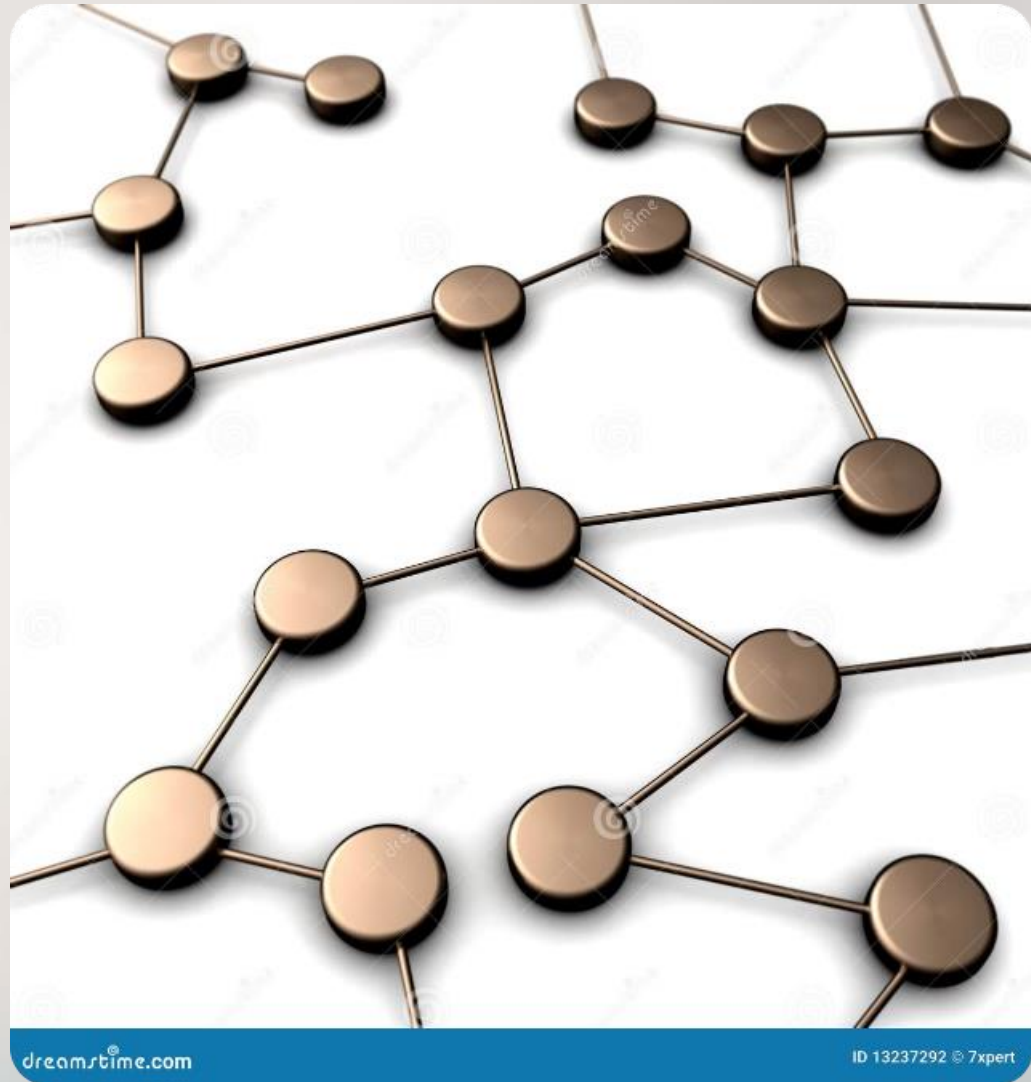
Develop a recurrence relation that expresses the minimum cost in terms of the minimum costs of smaller subproblems.

3

Bottom-Up Computation

Use a bottom-up dynamic programming approach to compute the minimum cost, building up the solution from smaller subproblems.





Example

Input

$\text{cost} = [[15, 96], [36, 2]]$

Output

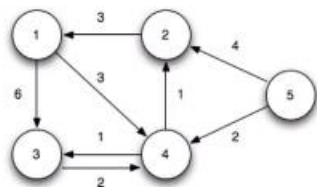
17

Explanation

The optimal way to connect the groups is: 1--A, 2--B, with a total a total cost of 17.

Dynamic Programming Solution

Dynamic Programming Introduction and examples



Before Text Justification
In computer science, mathematics, management science, economics and bioinformatics, dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions. The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of (it is hoped) a modest expenditure in storage space. (Each of the subproblem solutions is indexed in some way, typically based on the values of its input parameters, so as to facilitate its lookup.) The technique of storing solutions to subproblems instead of recomputing them is called "memoization".

Justify Text with Dynamic Programming
In computer science, mathematics, management science, economics and bioinformatics, dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions. The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of (it is hoped) a modest expenditure in storage space. (Each of the subproblem solutions is indexed in some way, typically based on the values of its input parameters, so as to facilitate its lookup.) The technique of storing solutions to subproblems instead of recomputing them is called "memoization".

1

Initialize

Create a 2D array dp to store the minimum costs for the subproblems.

2

Compute

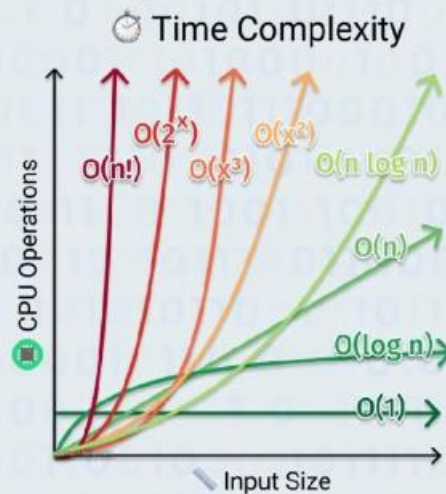
Fill in the dp array using the recurrence relation, starting from the base cases.

3

Return

The final answer is the value stored in dp[size1][size2].

Time and Space Complexity in Data Structure



Time and Space Complexity

1

Time Complexity

$O(\text{size1} * \text{size2})$, as we fill in the dp array using the recurrence relation.

2

Space Complexity

$O(\text{size1} * \text{size2})$, as we use a 2D array to store the subproblem subproblem solutions.

Optimizations and Extensions



Optimizations

Potential optimizations include using memoization or a sparse matrix representation to reduce the space complexity.

Extensions

The problem can be extended to consider other constraints, such as a maximum number of connections per point or a budget limit.



conclusion

Conclusion



Key Takeaways

The minimum cost to connect two groups of points can be solved using a dynamic programming approach, which involves defining subproblems and a recurrence relation.



Optimization Potential

There are opportunities to optimize the solution further, such as using memoization or sparse data structures.



Problem Extensions

The problem can be extended to consider additional constraints, making it a versatile and useful tool in various applications.