39 Assignment

#include <stdio.h>

#include <limits.h>

#define MAX\_N 50 // Adjust this value as needed

// Function to find the minimum cost assignment for a given row

int minCostAssignment(int cost[][MAX\_N], int n, int row, bool assigned[]) {

int min\_cost = INT\_MAX;

int min\_idx = -1;

// Find the minimum element in the unassigned row

for (int col = 0; col < n; col++) {

if (!assigned[col] && cost[row][col] < min\_cost) {

min\_cost = cost[row][col];

min\_idx = col;

}

}

// Recursively call for the remaining rows

if (min\_idx != -1) {

assigned[min\_idx] = true; // Mark the selected column as assigned

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

if (!assigned[i]) {

cost[row][i] -= min\_cost; // Reduce cost by min\_cost to avoid redundant assignments

}

}

int min\_cost\_recurse = minCostAssignment(cost, n, row + 1, assigned);

min\_cost = min(min\_cost, min\_cost\_recurse); // Update minimum cost

// Backtrack: reset assigned state and cost reduction

assigned[min\_idx] = false;

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

if (!assigned[i]) {

cost[row][i] += min\_cost; // Undo cost reduction

}

}

}

return min\_cost;

}

// Function to find the minimum cost assignment for the entire problem

int assignmentProblem(int cost[][MAX\_N], int n) {

bool assigned[MAX\_N] = {false};

// Find the minimum cost for all rows

int min\_cost = minCostAssignment(cost, n, 0, assigned);

return min\_cost;

}

int main() {

/\*

Example cost matrix (modify as needed):

0 2 4

1 3 1

2 1 0

\*/

int cost[][MAX\_N] = {

{0, 2, 4},

{1, 3, 1},

{2, 1, 0}

};

int n = sizeof(cost[0]) / sizeof(cost[0][0]); // Number of rows (and columns)

int min\_cost = assignmentProblem(cost, n);

printf("Minimum cost of assignment: %d\n", min\_cost);

return 0;

}