ADA LAB extra programs

1. Pattern matching

#include <stdio.h>

#include <string.h>

void naivePatternSearch(char \*text, char \*pattern) {

int n = strlen(text);

int m = strlen(pattern);

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

int j;

for (j = 0; j < m; j++) {

if (text[i + j] != pattern[j])

break;

}

if (j == m) {

printf("Pattern found at index %d\n", i);

}

}

}

int main() {

char text[100], pattern[100];

printf("Enter text: ");

gets(text);

printf("Enter pattern: ");

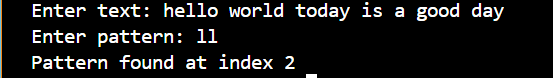
gets(pattern);

naivePatternSearch(text, pattern);

return 0;

}

**OUTPUT:**

****

1. Travelling salesman

#include <stdio.h>

#include <limits.h>

int n, cost[10][10];

int minCost = INT\_MAX;

int calculateCost(int tour[]) {

int totalCost = 0;

for (int i = 0; i < n - 1; i++) {

totalCost += cost[tour[i]][tour[i + 1]];

}

totalCost += cost[tour[n - 1]][tour[0]];

return totalCost;

}

void tsp(int tour[], int visited[], int count) {

if (count == n) {

int currentCost = calculateCost(tour);

if (currentCost < minCost) {

minCost = currentCost;

}

return;

}

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

if (!visited[i]) {

visited[i] = 1;

tour[count] = i;

tsp(tour, visited, count + 1);

visited[i] = 0;

}

}

}

int main() {

printf("Enter number of cities: ");

scanf("%d", &n);

printf("Enter cost matrix:\n");

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

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

scanf("%d", &cost[i][j]);

}

}

int tour[10] = {0};

int visited[10] = {0};

visited[0] = 1;

tour[0] = 0;

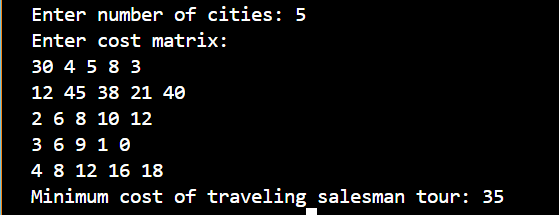
tsp(tour, visited, 1);

printf("Minimum cost of traveling salesman tour: %d\n", minCost);

return 0;

}

**OUTPUT:**

****

1. Assignment Problems

#include <stdio.h>

#include <limits.h>

int n, cost[10][10];

int minCost = INT\_MAX;

void assignment(int worker, int currentCost, int visited[10]) {

if (worker == n) {

if (currentCost < minCost) {

minCost = currentCost;

}

return;

}

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

if (!visited[task]) {

visited[task] = 1;

assignment(worker + 1, currentCost + cost[worker][task], visited);

visited[task] = 0;

}

}

}

int main() {

printf("Enter number of workers/tasks: ");

scanf("%d", &n);

printf("Enter cost matrix:\n");

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

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

scanf("%d", &cost[i][j]);

}

}

int visited[10] = {0};

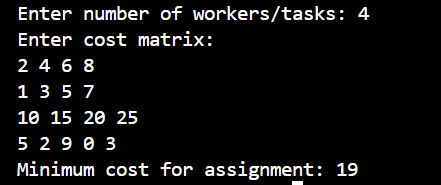
assignment(0, 0, visited);

printf("Minimum cost for assignment: %d\n", minCost);

return 0;

}

**OUTPUT:**

****

1. Knapsack using brute force

#include <stdio.h>

#include <math.h>

int max(int a, int b) {

return (a > b) ? a : b;

}

void evaluateSubset(int n, int weights[], int values[], int subset, int capacity, int \*maxValue) {

int totalWeight = 0;

int totalValue = 0;

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

if (subset & (1 << i)) {

totalWeight += weights[i];

totalValue += values[i];

}

}

if (totalWeight <= capacity && totalValue > \*maxValue) {

\*maxValue = totalValue;

}

}

int main() {

int n, capacity;

printf("Enter the number of items: ");

scanf("%d", &n);

int weights[n], values[n];

printf("Enter the weights of the items:\n");

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

scanf("%d", &weights[i]);

}

printf("Enter the values of the items:\n");

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

scanf("%d", &values[i]);

}

printf("Enter the capacity of the knapsack: ");

scanf("%d", &capacity);

int maxValue = 0;

int totalSubsets = pow(2, n);

for (int subset = 0; subset < totalSubsets; subset++) {

evaluateSubset(n, weights, values, subset, capacity, &maxValue);

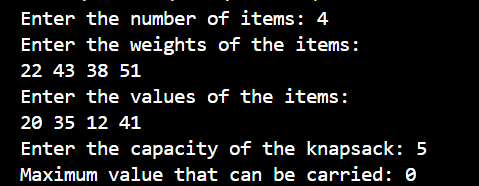
}

printf("Maximum value that can be carried: %d\n", maxValue);

return 0;

}

**OUTPUT:**

****