Lab 5:

Knapsack algorithm

#include <stdio.h>

#define MAX 100

int n, capacity;

int weights[MAX], values[MAX];

int maxValue = 0;

int selected[MAX], bestSelected[MAX];

int dp[MAX][MAX];

int max(int a, int b) {

return (a > b) ? a : b;

}

void knapsackBT(int index, int currWeight, int currValue) {

if (index == n) {

if (currWeight <= capacity && currValue > maxValue) {

maxValue = currValue;

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

bestSelected[i] = selected[i];

}

return;

}

selected[index] = 0;

knapsackBT(index + 1, currWeight, currValue);

if (currWeight + weights[index] <= capacity) {

selected[index] = 1;

knapsackBT(index + 1, currWeight + weights[index], currValue + values[index]);

}

}

void knapsackDP() {

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

for (int w = 0; w <= capacity; w++) {

if (i == 0 || w == 0)

dp[i][w] = 0;

else if (weights[i - 1] <= w)

dp[i][w] = max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);

else

dp[i][w] = dp[i - 1][w];

}

}

}

void printDPTable() {

printf("\nDP Table:\n");

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

for (int w = 0; w <= capacity; w++) {

printf("%2d ", dp[i][w]);

}

printf("\n");

}

printf("Maximum value in knapsack: %d\n", dp[n][capacity]);

}

int main() {

printf("Enter number of items: ");

scanf("%d", &n);

printf("Enter weights and values of each item:\n");

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

printf("Item %d - Weight: ", i + 1);

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

printf("Item %d - Value: ", i + 1);

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

}

printf("Enter knapsack capacity: ");

scanf("%d", &capacity);

knapsackDP();

printDPTable();

knapsackBT(0, 0, 0);

printf("\nItems selected (1 = included, 0 = excluded):\n");

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

printf("Item %d: %d\n", i + 1, bestSelected[i]);

}

return 0;

}

Output:

