1. Greedy Approach: Implementation of Fractional Knapsack

Method1:

#include <stdio.h>

void swap(float \*a, float \*b) {

float temp = \*a;

\*a = \*b;

\*b = temp;

}

float greedyMaxProfit(int n, float w[], float p[], float capacity) {

float maxProfit = 0;

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

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

if (p[i] < p[j]) {

swap(&p[i], &p[j]);

swap(&w[i], &w[j]);

}

}

}

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

if (capacity == 0) break;

if (w[i] <= capacity) {

maxProfit += p[i];

capacity -= w[i];

} else {

maxProfit += (p[i] / w[i]) \* capacity;

capacity = 0;

}

}

return maxProfit;

}

int main() {

int n;

float capacity;

printf("Enter number of items: ");

scanf("%d", &n);

float w[n], p[n];

printf("Enter weights of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &w[i]);

printf("Enter profits of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &p[i]);

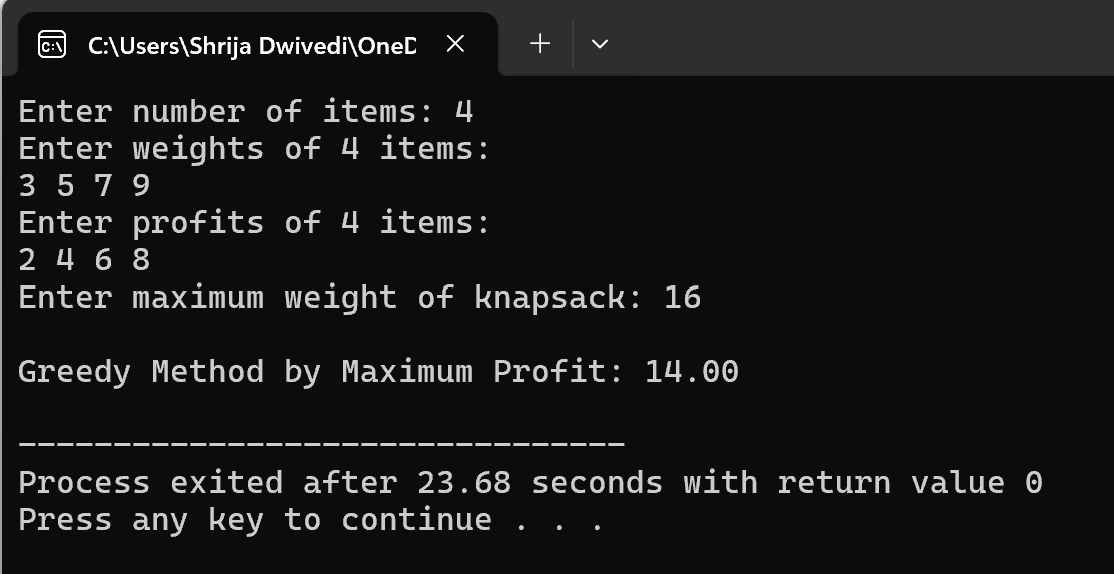
printf("Enter maximum weight of knapsack: ");

scanf("%f", &capacity);

printf("\nGreedy Method by Maximum Profit: %.2f\n", greedyMaxProfit(n, w, p, capacity));

return 0;

}



Method2:

#include <stdio.h>

void swap(float \*a, float \*b) {

float temp = \*a;

\*a = \*b;

\*b = temp;

}

float greedyMinWeight(int n, float w[], float p[], float capacity) {

float maxProfit = 0;

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

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

if (w[i] > w[j]) {

swap(&p[i], &p[j]);

swap(&w[i], &w[j]);

}

}

}

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

if (capacity == 0) break;

if (w[i] <= capacity) {

maxProfit += p[i];

capacity -= w[i];

} else {

maxProfit += (p[i] / w[i]) \* capacity;

capacity = 0;

}

}

return maxProfit;

}

int main() {

int n;

float capacity;

printf("Enter number of items: ");

scanf("%d", &n);

float w[n], p[n];

printf("Enter weights of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &w[i]);

printf("Enter profits of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &p[i]);

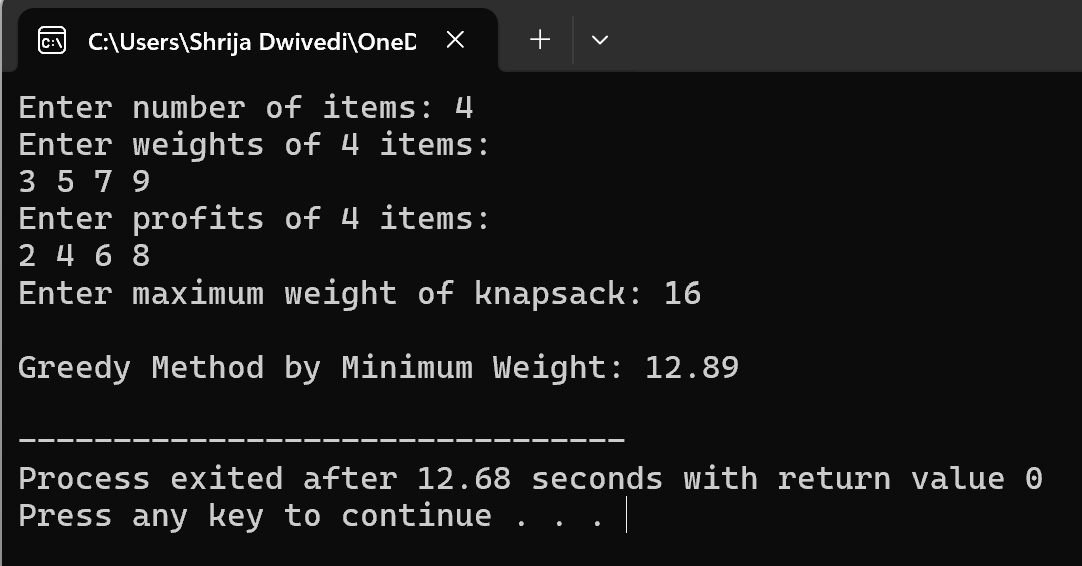
printf("Enter maximum weight of knapsack: ");

scanf("%f", &capacity);

printf("\nGreedy Method by Minimum Weight: %.2f\n", greedyMinWeight(n, w, p, capacity));

return 0;

}



Method3:

#include <stdio.h>

void swap(float \*a, float \*b) {

float temp = \*a;

\*a = \*b;

\*b = temp;

}

float greedyRatio(int n, float w[], float p[], float capacity) {

float ratio[n];

float maxProfit = 0;

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

ratio[i] = p[i] / w[i];

}

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

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

if (ratio[i] < ratio[j]) {

swap(&ratio[i], &ratio[j]);

swap(&p[i], &p[j]);

swap(&w[i], &w[j]);

}

}

}

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

if (capacity == 0) break;

if (w[i] <= capacity) {

maxProfit += p[i];

capacity -= w[i];

} else {

maxProfit += (p[i] / w[i]) \* capacity;

capacity = 0;

}

}

return maxProfit;

}

int main() {

int n;

float capacity;

printf("Enter number of items: ");

scanf("%d", &n);

float w[n], p[n];

printf("Enter weights of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &w[i]);

printf("Enter profits of %d items:\n", n);

for (int i = 0; i < n; i++) scanf("%f", &p[i]);

printf("Enter maximum weight of knapsack: ");

scanf("%f", &capacity);

printf("\nGreedy Method by Profit/Weight Ratio: %.2f\n", greedyRatio(n, w, p, capacity));

return 0;

}

