LAB TASK – 7

1. Greedy Approach: Implementation of Fractional Knapsack

**Method 1 : In Descending order of their Profits**

**Code :**

#include <stdio.h>

struct Item {

    int profit;

    int weight;

};

void sorting(struct Item a[], int n) {

    for (int i = 0; i < n - 1; i++) {

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

            if (a[i].profit < a[j].profit) {

                struct Item t = a[i];

                a[i] = a[j];

                a[j] = t;

            }

        }

    }

}

double knapsack(struct Item a[], int n, int w) {

    double total = 0.0;

    for (int i = 0; i < n; i++) {

        if (a[i].weight <= w) {

            w = w-a[i].weight;

            total=total+a[i].profit;

        } else {

            total=total+a[i].profit\*((double)w/a[i].weight);

            break;

        } }

    return total;

}

int main() {

    int n, W;

    printf("Enter number of items: ");

    scanf("%d", &n);

    printf("Enter knapsack capacity: ");

    scanf("%d", &W);

    struct Item items[n];

    printf("Enter weight and profit of each item:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d %d",&items[i].weight, &items[i].profit);

    }

    sorting(items, n);

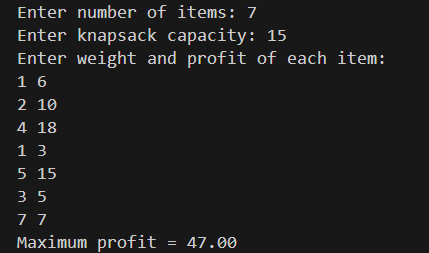
    double max = knapsack(items, n, W);

    printf("Maximum profit = %.2f", max);

    return 0;

}

**Output :**



**Method 2 : In Ascending order of their Weights**

**Code :**

#include <stdio.h>

struct Item {

    int profit;

    int weight;

};

void sorting(struct Item a[], int n) {

    for (int i = 0; i < n - 1; i++) {

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

            if (a[i].weight > a[j].weight) {

                struct Item t = a[i];

                a[i] = a[j];

                a[j] = t;

            }

        }

    }

}

double knapsack(struct Item a[], int n, int w) {

    double total = 0.0;

    for (int i = 0; i < n; i++) {

        if (a[i].weight <= w) {

            w = w-a[i].weight;

            total=total+a[i].profit;

        } else {

            total=total+a[i].profit\*((double)w/a[i].weight);

            break;

        }

    }

    return total;

}

int main() {

    int n, W;

    printf("Enter number of items: ");

    scanf("%d", &n);

    printf("Enter knapsack capacity: ");

    scanf("%d", &W);

    struct Item items[n];

    printf("Enter weight and profit of each item:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d %d",&items[i].weight, &items[i].profit);

    }

    sorting(items, n);

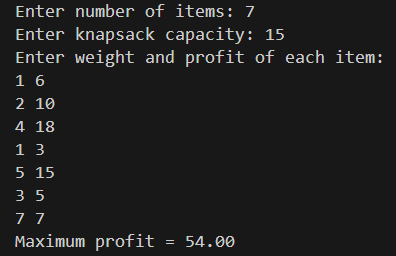
    double max = knapsack(items, n, W);

    printf("Maximum profit = %.2f", max);

    return 0;

}

**Output :**

****

**Method 3 : In Descending order of Profit/Weight Ratio**

**Code :**

#include <stdio.h>

struct Item {

    int profit;

    int weight;

    double ratio;

};

void sort(struct Item arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

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

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

                struct Item t = arr[i];

                arr[i] = arr[j];

                arr[j] = t;

            }

        }

    }

}

double knapsack(struct Item arr[], int n, int W) {

    double total = 0.0;

    for (int i = 0; i < n; i++) {

        if (arr[i].weight <= W) {

            W = W-arr[i].weight;

            total=total+arr[i].profit;

        } else {

            total=total+arr[i].profit \*((double)W/arr[i].weight);

            break;

        }

    }

    return total;

}

int main() {

    int n, W;

    printf("Enter number of items: ");

    scanf("%d", &n);

    printf("Enter knapsack capacity: ");

    scanf("%d", &W);

    struct Item items[n];

    printf("Enter weight and profit of each item:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d %d", &items[i].weight, &items[i].profit);

        items[i].ratio = (double)items[i].profit / items[i].weight;

    }

    sort(items, n);

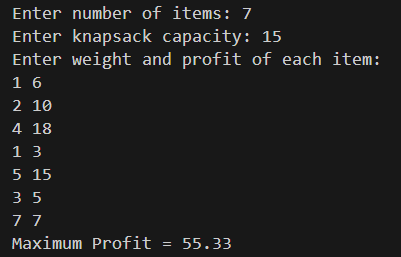
    double max = knapsack(items, n, W);

    printf("Maximum Profit = %.2f\n", max);

    return 0;

}

**Output :**

****