1. **Knapsack using greedy techniques**

**Code:**

#include <stdio.h>

#include <stdlib.h>

// Structure to represent items

struct Item {

int weight;

int value;

};

// Function to compare items based on their value/weight ratio

int compare(const void \*a, const void \*b) {

double ratio1 = (double)(((struct Item\*)a)->value / (double)((struct Item\*)a)->weight);

double ratio2 = (double)(((struct Item\*)b)->value / (double)((struct Item\*)b)->weight);

if (ratio1 < ratio2)

return 1;

else if (ratio1 > ratio2)

return -1;

else

return 0;

}

// Function to solve fractional knapsack problem

double fractionalKnapsack(int capacity, struct Item items[], int n) {

// Sort items based on value/weight ratio

qsort(items, n, sizeof(struct Item), compare);

double totalValue = 0.0; // Total value of items in knapsack

int currentWeight = 0; // Current weight in knapsack

// Iterate through sorted items and add to knapsack as much as possible

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

if (currentWeight + items[i].weight <= capacity) {

// If the whole item can be added

currentWeight += items[i].weight;

totalValue += items[i].value;

} else {

// Otherwise, add a fraction of the item

int remainingWeight = capacity - currentWeight;

totalValue += items[i].value \* ((double)remainingWeight / items[i].weight);

break; // No more items can be added

}

}

return totalValue;

}

int main() {

int capacity; // Capacity of knapsack

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

scanf("%d", &capacity);

int n; // Number of items

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

scanf("%d", &n);

struct Item items[n]; // Array of items

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

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

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

printf("Weight: ");

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

printf("Value: ");

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

}

double maxValue = fractionalKnapsack(capacity, items, n);

printf("Maximum value in Knapsack = %.2lf\n", maxValue);

return 0;

}

**Output:**

Enter the capacity of the knapsack: 30

Enter the number of items: 4

Enter the weight and value of each item:

Item 1:

Weight: 3

Value: 40

Item 2:

Weight: 15

Value: 20

Item 3:

Weight: 10

Value: 10

Item 4:

Weight: 15

Value: 40

Maximum value in Knapsack = 96.00

--------------------------------

Process exited after 53.62 seconds with return value 0

Press any key to continue . . .

