Exp No.3

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

double fractional\_knapsack(const vector<double>& weights, const vector<double>& values, double capacity) {

if (weights.size() != values.size()) {

throw invalid\_argument("Weights and values must be of the same length.");

}

double res = 0.0;

vector<pair<double, double>> items;

for (size\_t i = 0; i < weights.size(); ++i) {

items.push\_back({weights[i], values[i]});

}

sort(items.begin(), items.end(), [](const pair<double, double>& a, const pair<double, double>& b) {

return (b.second / b.first) < (a.second / a.first);

});

for (const auto& item : items) {

if (capacity <= 0) break;

if (item.first > capacity) {

res += capacity \* (item.second / item.first);

capacity = 0;

} else {

res += item.second;

capacity -= item.first;

}

}

return res;

}

int main() {

int num\_items;

cout << "Enter the number of items: ";

cin >> num\_items;

vector<double> weights(num\_items);

vector<double> values(num\_items);

cout << "Enter the weights (space-separated): ";

for (int i = 0; i < num\_items; ++i) {

cin >> weights[i];

}

cout << "Enter the values (space-separated): ";

for (int i = 0; i < num\_items; ++i) {

cin >> values[i];

}

if (weights.size() != num\_items || values.size() != num\_items) {

throw invalid\_argument("The number of weights and values must match the number of items.");

}

double capacity;

cout << "Enter the capacity of the knapsack: ";

cin >> capacity;

double max\_value = fractional\_knapsack(weights, values, capacity);

cout << "Maximum value in the knapsack: " << max\_value << endl;

return 0;

}

OUTPUT: -

