|  |  |
| --- | --- |
| **NAME:** | Akshaya Karande |
| **UID:** | 2021300056 |
| **SUBJECT** | Design and Analysis of Algorithms |
| **EXPERIMENT NO:** | 5 |
| **AIM:** | To implement Knapsack Problem |
| **Theory:** | The **knapsack problem** is the following problem in [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization):  *Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.* |
| **Algorithm:** | **Algorithm GREEDY\_FRACTIONAL\_KNAPSACK(X, V, W, M)**  **Description: Solve the knapsack problem using greedy approach**  **Input:**  X: An array of n items  V: An array of profit associated with each item  W: An array of weight associated with each item  M: Capacity of knapsack  **Output:**  SW: Weight of selected items  SP: Profit of selected items  Items are sorted in decreasing order of pi = vi / wi ratio  S ← Φ // Set of selected items, initially empty  SW ← 0 // weight of selected items  SP ← 0 // profit of selected items  i ← 1  while i ≤ n do  if (SW + w[i]) ≤ M then  S ← S ∪ X[i]  SW ← SW + W[i]  SP ← SP + V[i]  else  frac ← (M - SW) / W[i]  S ← S ∪ X[i] \* frac // Add fraction of item X[i]  SP ← SP + V[i] \* frac // Add fraction of profit  SW ← SW + W[i] \* frac // Add fraction of weight  end  i ← i + 1  end |
| **CODE:** | #include <bits/stdc++.h>  #include <iostream>  #include <algorithm>  #include <string>  #include <cmath>  using namespace std;  void printarr(double \*\*arr,int n) {  cout << "Item \t\t\t Weight \t\t Value \t\t Value/Weight\n";  for (int i = 0; i < n; i++) {  for (int j = 0; j < 4; j++) {  cout << arr[i][j] << "\t\t\t";  }  cout << "\n";  }  }  int main()  {  int c, n;  double profit = 0.0,weight = 0.0;  cout << "\nEnter the weight of the sack: ";  cin >> c;  cout << "\nEnter the no of items: ";  cin >> n;  cout << "\nEnter weight and value of each item: \n\n";  vector<string> s (n);  double \*\*arr = new double\*[n];  for (int i = 0; i < n; i++) {  arr[i] = new double[4];  arr[i][0]=i+1;  for (int j = 1; j < 4; j++) {  if (j == 3)  {  arr[i][j] = arr[i][2] / arr[i][1];  }  else {  cout << "Enter weight and value for [" << i << "][" << j << "]: ";  cin >> arr[i][j];  }  }  }  cout<<endl;  printarr(arr,n);  cout << "\nSorted based on ratio: \n" << endl;  sort(arr, arr + n, [](const double\* a, const double\* b) {  return a[3] > b[3];  });  printarr(arr,n);  int remain = 0;  double remain\_pro = 0.0;  string coco = "";  ostringstream ss;  for (int i = 0; i < n; i++) {  if (c >= weight + arr[i][1]){  weight += arr[i][1];  s[i] = to\_string(lround(arr[i][0]));  profit += arr[i][2];  }  else  {  remain = c - weight;  weight += remain;  remain\_pro = (remain \* arr[i][2]) / arr[i][1];  profit += remain\_pro;  ss << remain << "/" << arr[i][1];  coco = ss.str();  s[i] = to\_string(lround(arr[i][0])) + " (" + coco + ")";  break;  }  }  cout << "\nTotal weight: " << weight << endl;  cout << "\nTotal profit: " << profit << endl;  cout << "\nAll items in the bag: {";  for (int i = 0; i < s.size(); i++)  {  cout << s[i] << ",";  }  cout << "}";  cout<<endl;  cout<<endl;  return 0;  } |
| **Output:** |  |
| **Conclusion:** | In conclusion, the Fractional Knapsack Problem is a classic problem in computer science that involves maximizing the value of items that can be placed into a knapsack with limited capacity. The greedy approach is a popular and effective method for solving this problem, which involves selecting items with the highest value-to-weight ratio at each step.The experiment using the greedy approach for the Fractional Knapsack Problem has shown that this algorithm can provide efficient and near-optimal solutions in a reasonable amount of time. The algorithm was able to solve various instances of the problem with different item values and weights and produced solutions that were close to the optimal solutions.The success of the greedy algorithm in solving the Fractional Knapsack Problem highlights the importance of using greedy strategies in solving real-life problems. |