|  |  |
| --- | --- |
| **NAME:** | Vinit Madhyan |
| **UID:** | 2021300070 |
| **SUBJECT** | DAA |
| **EXPERIMENT NO:** | 5 |
| **AIM:** | Experiment based on greedy approach (fractional knapsack problem). |
| **Algorithm:** | * Calculate the ratio(value/weight) for each item. * Sort all the items in decreasing order of the ratio. * Initialize res =0, curr\_cap = given\_cap. * Do the following for every item “i” in the sorted order:   + If the weight of the current item is less than or equal to the remaining capacity then add the value of that item into the result   + Else add the current item as much as we can and break out of the loop. * Return res. |
| **Code:** | *#include* <iostream>  using namespace std;  int knapsack(int n, int c, int we[], int pr[])  {      int curwe = 0;      int curr = 0;      float curpr = 0;      float frac[n];  *for* (int i = 0; i < n; i++)      {          frac[i] = ((float)pr[i] / (float)we[i]);      }      cout << "input data" << endl;      cout << "sr.\tweight\tprofit\tfraction" << endl;  *for* (int i = 0; i < n; i++)      {          cout << i + 1 << "\t" << we[i] << "\t" << pr[i] << "\t" << frac[i] << endl;      }  *for* (int i = 0; i < n; i++)      {          int m = i;          float temp;          int k;  *for* (int j = i + 1; j < n; j++)          {  *if* (frac[j] > frac[m])              {                  m = j;              }          }          temp = frac[i];          frac[i] = frac[m];          frac[m] = temp;          k = we[i];          we[i] = we[m];          we[m] = k;          k = pr[i];          pr[i] = pr[m];          pr[m] = k;      }      cout << "data after sorting" << endl;      cout << "sr.\tweight\tprofit\tfraction" << endl;  *for* (int i = 0; i < n; i++)      {          cout << i + 1 << "\t" << we[i] << "\t" << pr[i] << "\t" << frac[i] << endl;      }      for (int i = 0; i < n; i++)      {          curr = curr + we[i];          // cout << "main ";          if (curr <= c)          {              // cout << "if ";              curwe = curwe + we[i];              curpr = curpr + (float)pr[i];          }          else          {              // cout << "else ";              int w = c - curwe;              cout << w;              float z = ((float)w \* frac[i]);              cout << z;              curpr = curpr + z;              return curpr;          }      }      return curpr;  }  int main()  {      int n, capacity;      cout << "enter no of iteration : ";      cin >> n;      cout << "enter capacity : ";      cin >> capacity;      int we[n];      int pr[n];      cout << "enter weights for " << n << " iteration";      for (int i = 0; i < n; i++)      {          cin >> we[i];      }      cout << "enter profits for " << n << " iteration";      for (int i = 0; i < n; i++)      {          cin >> pr[i];      }      int ans = knapsack(n, capacity, we, pr);      cout << "ans is " << ans;  } |
| **Graphs and Observation:** | **Inference**  Thus, we observe that by using the greedy approach which allows us to get the best option possible without worrying about optimization, we can get the maximum profit possible for the total weight by calculating the value and weight ratio and sorting the table accordingly and also get the maximum number of items in the bag. |
| **Conclusion:** | With the help of this experiment, I was successfully able to understand and implement the concept of fractional knapsack problem. |