

|  |
| --- |
| **Title: Implementation of Knapsack Problem** |



**Objective:** To learn the Greedy strategy of solving the problems for different types of problems

**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations. |
| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |



**Books/ Journals/ Websites referred:**

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran,” Fundamentals of computer algorithm”, University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein,” Introduction to algortihtms”,2nd Edition ,MIT press/McGraw Hill,2001**
3. **http://lcm.csa.iisc.ernet.in/dsa/node184.htm**
4. **http://students.ceid.upatras.gr/~papagel/project/kruskal.htm**
5. [**http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/kruskalAlgor.html**](http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/kruskalAlgor.html)
6. **http://lcm.csa.iisc.ernet.in/dsa/node183.html**
7. **http://students.ceid.upatras.gr/~papagel/project/prim.htm**
8. **http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf**



**Pre Lab/ Prior Concepts:**

Data structures, Concepts of algorithm analysis



**Historical Profile:**

The knapsack problem represents constraint satisfaction optimization problems’ family. Based on nature of constraints, the knapsack problem can be solved with various problem saolving strategies. Typically, these problems represent resource optimization solution.

Given a set of n inputs. · Find a subset, called feasible solution, of the n inputs subject to some constraints, and satisfying a given objective function. · If the objective function is maximized or minimized, the feasible solution is optimal. · It is a locally optimal method.

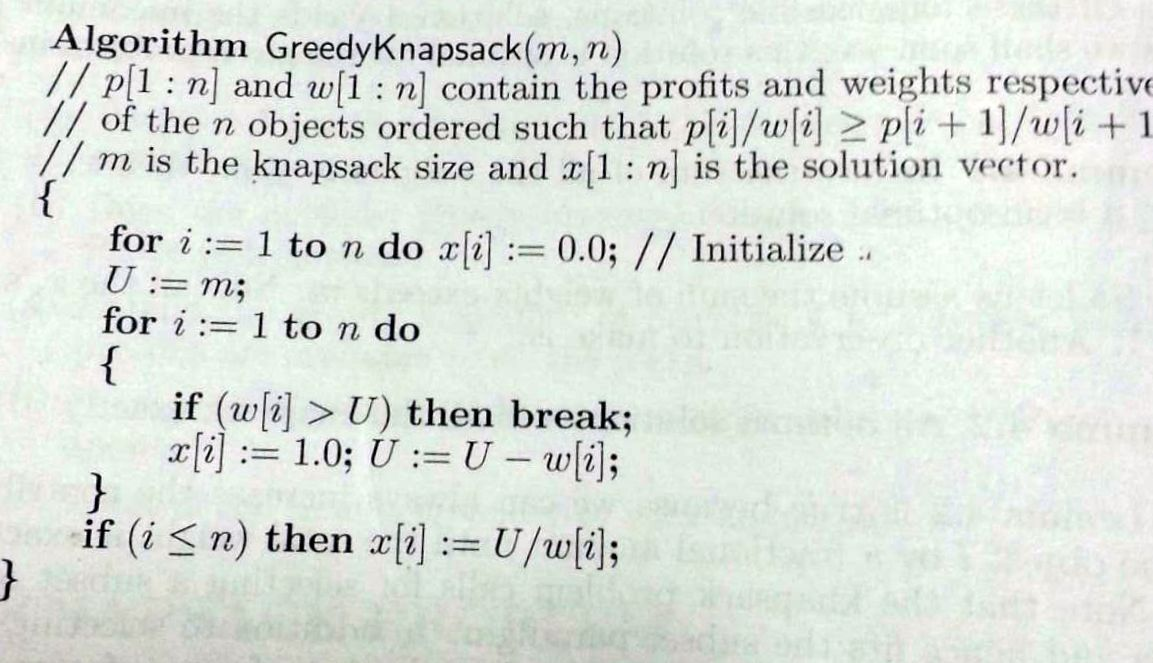


**New Concepts to be learned:**

Application of algorithmic design strategy to any problem, Greedy method of problem solving Vs other methods of problem solving, optimality of the solution, knapsack problem and their applications



**Knapsack Problem Algorithm**



**Example: Knapsack Problem**

Assume that we have a [knapsack](https://www.dyclassroom.com/greedy-algorithm/fractional-knapsack-problem#) with max weight capacity, **W = 16**.

Our objective is to fill the [knapsack](https://www.dyclassroom.com/greedy-algorithm/fractional-knapsack-problem#) with items such that the benefit (value or profit) is maximum.

Consider the following items and their associated weight and value

|  |  |  |
| --- | --- | --- |
| **ITEM** | **WEIGHT** | **VALUE** |
| i1 | 6 | 6 |
| i2 | 10 | 2 |
| i3 | 3 | 1 |
| i4 | 5 | 8 |
| i5 | 1 | 3 |
| i6 | 3 | 5 |

## 

## Steps

* Calculate value per weight for each item (we can call this value density)
* Sort the items as per the value density in descending order
* Take as much item as possible not already taken in the [knapsack](https://www.dyclassroom.com/greedy-algorithm/fractional-knapsack-problem#)

Compute density = (value/weight)

|  |  |  |  |
| --- | --- | --- | --- |
| **ITEM** | **WEIGHT** | **VALUE** | **DENSITY** |
| i1 | 6 | 6 | 1.000 |
| i2 | 10 | 2 | 0.200 |
| i3 | 3 | 1 | 0.333 |
| i4 | 5 | 8 | 1.600 |
| i5 | 1 | 3 | 3.000 |
| i6 | 3 | 5 | 1.667 |

**Sort the items as per density in descending order.**

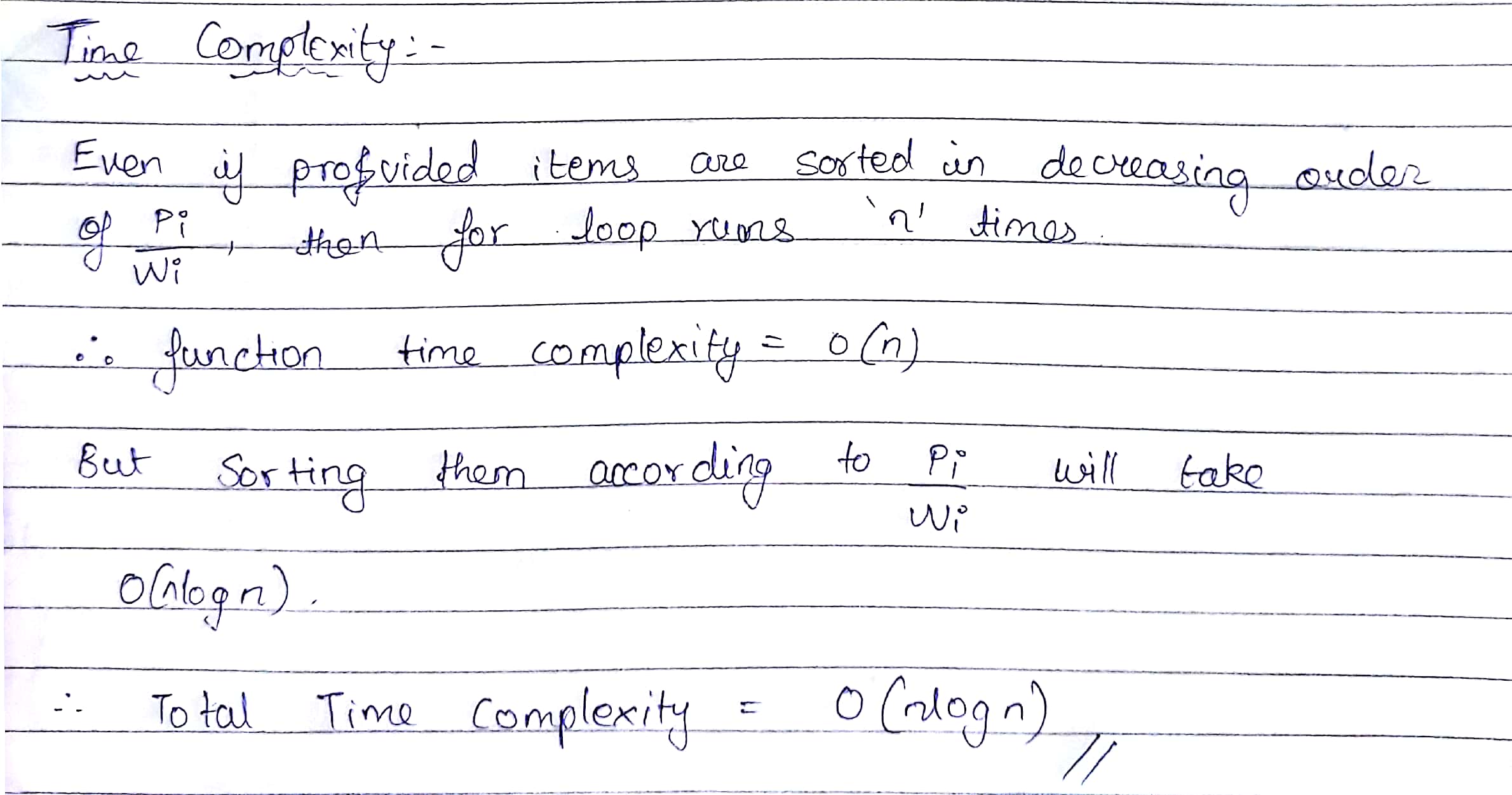
|  |  |  |  |
| --- | --- | --- | --- |
| **ITEM** | **WEIGHT** | **VALUE** | **DENSITY** |
| i5 | 1 | 3 | 3.000 |
| i6 | 3 | 5 | 1.667 |
| i4 | 5 | 8 | 1.600 |
| i1 | 6 | 6 | 1.000 |
| i3 | 3 | 1 | 0.333 |
| i2 | 10 | 2 | 0.200 |

**Now we will pick items such that our benefit is maximum and total weight of the selected items is at most W.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ITEM** | **WEIGHT** | **VALUE** | **TOTAL WEIGHT** | **TOTAL BENEFIT** |
| i5 | 1 | 3 | 1.000 | 3.000 |
| i6 | 3 | 5 | 4.000 | 8.000 |
| i4 | 5 | 8 | 9.000 | 16.000 |
| i1 | 6 | 6 | 15.000 | 22.000 |
| i3 | 1 | 0.333 | 16.000 | 22.333 |

**So, total weight in the** [**knapsack**](https://www.dyclassroom.com/greedy-algorithm/fractional-knapsack-problem#) **= 16 and total value inside it = 22.333336.**

**Analysis of Knapsack Problem algorithm:**

****

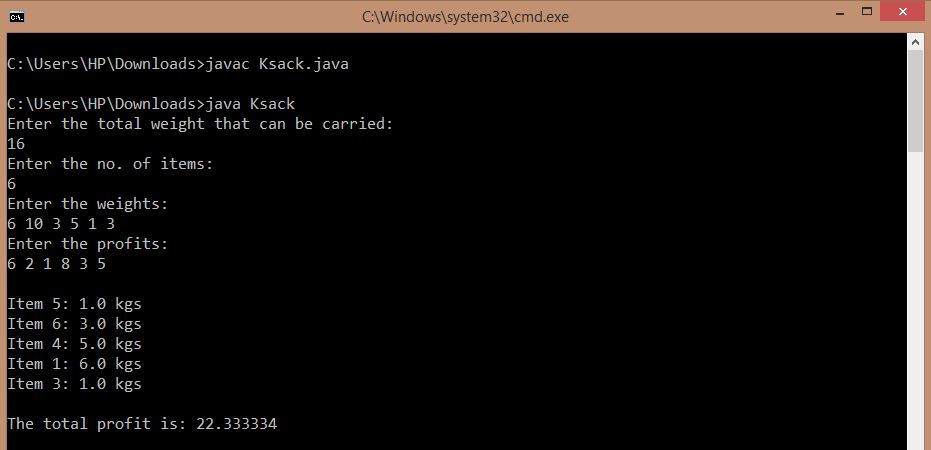
**CONCLUSION:**

Therefore, greedy strategy and its application for various problems has been studied and a program for the implementation of fractional knapsack problem has been implemented using this strategy. Further, the time and space complexity of the fractional knapsack algorithm was also calculated.

**Program**

import java.util.\*;  
  
public class Ksack  
{  
 public static void main(String[] args)  
 {  
 Scanner sc=new Scanner(System.in);  
 System.out.println("Enter the total weight that can be carried: ");  
 float weight=sc.nextFloat();  
 System.out.println("Enter the no. of items: ");  
 int n=sc.nextInt();  
 float w[]=new float[n];  
 float p[]=new float[n];  
 float p2[]=new float[n];  
 float r[]=new float[n];  
 int i;  
 System.out.println("Enter the weights: ");  
 for(i=0;i<n;i++)  
 {  
 w[i]=sc.nextFloat();  
 }  
 System.out.println("Enter the profits: ");  
 for(i=0;i<n;i++)  
 {  
 p[i]=sc.nextFloat();  
 }  
 for(i=0;i<n;i++)  
 {  
 p2[i]=p[i];  
 }  
 for(i=0;i<n;i++)  
 {  
 r[i]=p[i]/w[i];  
 }  
 int j,k,l;  
 float t1,t2;  
 float temp;  
 for(i=1;i<n;i++)  
 {  
 temp=r[i];  
 t1=p[i];  
 t2=w[i];  
 j=i-1;  
 while(j>=0&&r[j]<temp)  
 {  
 r[j+1]=r[j];  
 w[j+1]=w[j];  
 p[j+1]=p[j];  
 j--;  
 }  
 k=j+1;  
 r[k]=temp;  
 w[k]=t2;  
 p[k]=t1;  
 }  
 System.out.println("");  
 float profit=0;  
 for(i=0;i<n;i++)  
 {  
 if(w[i]<=weight)  
 {  
 profit=profit+(float)p[i];  
 weight=weight-w[i];  
 j=0;  
 while(p2[j]!=p[i])  
 {  
 j++;  
 }  
 p2[j]=0;  
 System.out.println("Item "+(j+1)+": "+w[i]+" kgs");  
 }  
 else if(w[i]>weight&&weight!=0)  
 {  
 float temp2=weight;  
 profit=profit+r[i]\*temp2;  
 weight=weight-temp2;  
 j=0;  
 while(p2[j]!=p[i])  
 {  
 j++;  
 }  
 p2[j]=0;  
 System.out.println("Item "+(j+1)+": "+temp2+" kgs");  
 }  
 }  
 System.out.println("\nThe total profit is: "+profit);  
 }  
}

**Output**

****