

Rajan Gautam

19BCP101

Div. II, CE 19
SOT, PDEU

Pandit Deendayal Energy University
School of Technology

Design & Analysis of Algorithm (20CP209P)

B. Tech - Computer Science & Engineering (Sem-IV)

Table of Contents

Lab 3 Assignment: Implementation of Greedy Algorithm	2
AIM 1: To write a C/C++ Program to implement Greedy Algorithm for given problem.	2
ALGORITHMS:	2
CODE:.....	3
OUTPUT:.....	4
AIM 2: To Implement Greedy Algorithm for computing Minimum Spanning Tree using Kruskal Algorithm.	5
Kruskal Algorithm:.....	5
CODE:.....	5
OUTPUT:	10
AIM 3: To solve the given problem.....	10
CODE:.....	11
OUTPUT 1:.....	11
OUTPUT 2:.....	13

Lab 3 Assignment: Implementation of Greedy Algorithm

AIM 1: To write a C/C++ Program to implement Greedy Algorithm for given problem.

Aim: Write a C/C++ Program (preferably) to implement Greedy Algorithm for given problem.

1. Consider the following 6 activities sorted by their finish time

Start[]	1	3	0	5	8	5
Finish[]	2	4	6	7	9	9

Print maximum set of activities that can be done by a single person, one at a time.

ALGORITHMS:

1. Insertion Sort Algorithm (Pseudocode)

INSERTION-SORT (A, n)

for j \leftarrow 2 to n

Do key \leftarrow A[j]

 i \leftarrow j - 1

while i > 0 and A[i] > key

Do A [i + 1] \leftarrow A[i]

 i \leftarrow i - 1

 A [i + 1] = key

2. Greedy Algorithm

Sort input activities in order by increasing finishing time.

n \leftarrow length[s]

A \leftarrow 1

j \leftarrow 1

for i \leftarrow 2 to n

if s_i \geq f_j **then**

 A \leftarrow A \cup (i)

 j \leftarrow i

return A

CODE:

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. // Algorithm to sort array
5. void insertion_sort(int A[], int n)
6. {
7.     int i, j, key;
8.     for(i = 1; i < n; i++)
9.     {
10.         key = A[i];
11.         j = i - 1;
12.         while (j >= 0 && A[j] > key)
13.         {
14.
15.             A[j+1] = A[j];
16.             j = j - 1;
17.         }
18.         A[j+1] = key;
19.     }
20. }
21.
22. // Algorithm for Greedy
23. int GreedyAlgorithm(int S[], int F[], int n)
24. {
25.     int A = 1;
26.     int j = 1;
27.     for (int i = 2; i <= n; i++)
28.     {
29.         if(S[i] >= F[j])
30.         {
31.             A++;
32.             j = i;
33.         }
34.     }
35.     return A;
36. }
37.
38. // Main Function
39. int main()
40. {
41.     // Scanning the total number of Activities
42.     int numActivity;
43.     printf("Enter total number of Activity: ");
44.     scanf("%d", &numActivity);
45.
46.     // Declaring two arrays
47.     int start[numActivity];
48.     int finish[numActivity];
49.
50.     // Getting the values of start Array
51.     printf("Enter starting time for %d Activities \n", numActivity);
```

```

52.
53.     for (int i = 0; i < numActivity; i++)
54.     {
55.         scanf("%d", &start[i]);
56.     }
57.
58.     // Getting the values for final Array
59.     printf("Enter finishing time for %d Activities \n", numActivity);
60.
61.     for (int i = 0; i < numActivity; i++)
62.     {
63.         scanf("%d", &finish[i]);
64.     }
65.
66.     // Sorting the finish array using Insertion Sort Algorithm
67.     insertion_sort(finish, numActivity);
68.
69.     // Getting the maximum set of activities that can be done by a single
    person
70.     int maxNum = GreedyAlgorithm(start, finish, numActivity);
71.
72.     // Printing maximum number returned by our GreedyAlgorithm Function
73.     printf("Maximum Set of Activities that can be done by a single
    person: %d\n", maxNum);
74.
75.     return 0;
76. }

```

OUTPUT:

The screenshot displays a C++ IDE with the following code and output:

```

main.c x
39 int main()
40 {
41     // Scanning the total number of Activities
42     int numActivity;
43     printf("Enter total number of Activity: ");
44     scanf("%d", &numActivity);
45
46     // Declaring two arrays
47     int start[numActivity];
48     int finish[numActivity];
49
50     // Getting the values of start Array
51     printf("Enter starting time for %d Activities \n", numActivity);
52
53     for (int i = 0; i < numActivity; i++)
54     {
55         scanf("%d", &start[i]);
56     }
57
58     // Getting the values for final Array
59     printf("Enter finishing time for %d Activities \n", numActivity);
60
61     for (int i = 0; i < numActivity; i++)
62     {
63         scanf("%d", &finish[i]);
64     }
65
66     // Sorting the finish array using Insertion Sort Algorithm
67     insertion_sort(finish, numActivity);
68
69     // Getting the maximum set of activities that can be done by a single person
70     int maxNum = GreedyAlgorithm(start, finish, numActivity);
71
72     // Printing maximum number returned by our GreedyAlgorithm Function
73     printf("Maximum Set of Activities that can be done by a single person: %d\n", maxNum);
74
75     return 0;
76 }

```

Output:

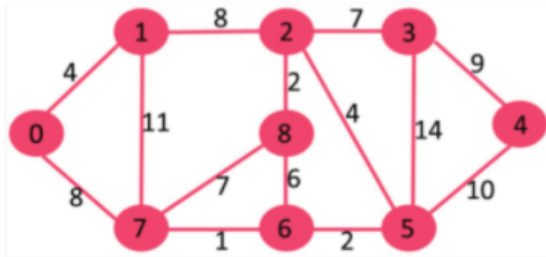
```

Enter total number of Activity: 6
Enter starting time for 6 Activities
1 3 0 5 8 5
Enter finishing time for 6 Activities
2 4 6 7 9 9
Maximum Set of Activities that can be done by a single person: 4
Process returned 0 (0x0)   execution time : 34.508 s
Press any key to continue.

```

AIM 2: To Implement Greedy Algorithm for computing Minimum Spanning Tree using Kruskal Algorithm.

2. Implement Greedy Algorithm for computing Minimum Spanning Tree Using (Kruskal/Prim's) Algorithm.



1. Print the number of edges and total cost of Minimum Spanning Tree.
2. Analyze the time complexity of your algorithm.

Kruskal Algorithm:

KRUSKAL(G):

$A = \emptyset$

For each vertex $v \in G.V$:

MAKE-SET(v)

For each edge $(u, v) \in G.E$ ordered by increasing order by weight (u, v) :

 if **FIND-SET**(u) \neq **FIND-SET**(v):

$A = A \cup \{(u, v)\}$

UNION (u, v)

CODE:

```
1. #include <stdio.h>
2. #define MAX 30
3.
4. typedef struct edge
5. {
6.     int u, v, w;
7. } edge;
8.
9. typedef struct edge_list
10. {
11.     edge data[MAX];
12.     int n;
```

```

13. }    edge_list;
14.
15. edge_list elist;
16.
17. int Graph[MAX][MAX], n;
18. edge_list spanlist;
19.
20. void kruskalAlgo();
21. int find(int belongs[], int vertexno);
22. void applyUnion(int belongs[], int c1, int c2);
23. void sort();
24. void print();
25.
26. // Applying Krushkal Algo
27. void kruskalAlgo()
28. {
29.     int belongs[MAX], i, j, cno1, cno2;
30.     elist.n = 0;
31.
32.     for (i = 1; i < n; i++)
33.         for (j = 0; j < i; j++)
34.             {
35.                 if (Graph[i][j] != 0)
36.                 {
37.                     elist.data[elist.n].u = i;
38.                     elist.data[elist.n].v = j;
39.                     elist.data[elist.n].w = Graph[i][j];
40.                     elist.n++;
41.                 }
42.             }
43.
44.     sort();
45.
46.     for (i = 0; i < n; i++)
47.         belongs[i] = i;
48.
49.     spanlist.n = 0;
50.
51.     for (i = 0; i < elist.n; i++)
52.     {
53.         cno1 = find(belongs, elist.data[i].u);
54.         cno2 = find(belongs, elist.data[i].v);
55.
56.         if (cno1 != cno2)
57.         {
58.             spanlist.data[spanlist.n] = elist.data[i];
59.             spanlist.n = spanlist.n + 1;
60.             applyUnion(belongs, cno1, cno2);
61.         }
62.     }
63. }
64.
65. int find(int belongs[], int vertexno)

```

```
66. {
67.     return (belongs[vertexno]);
68. }
69.
70. void applyUnion(int belongs[], int c1, int c2)
71. {
72.     int i;
73.
74.     for (i = 0; i < n; i++)
75.         if (belongs[i] == c2)
76.             belongs[i] = c1;
77. }
78.
79. // Sorting algo
80. void sort()
81. {
82.     int i, j;
83.     edge temp;
84.
85.     for (i = 1; i < elist.n; i++)
86.         for (j = 0; j < elist.n - 1; j++)
87.             if (elist.data[j].w > elist.data[j + 1].w)
88.             {
89.                 temp = elist.data[j];
90.                 elist.data[j] = elist.data[j + 1];
91.                 elist.data[j + 1] = temp;
92.             }
93. }
94.
95. // Printing the result
96. void print()
97. {
98.     int i, cost = 0;
99.     printf("Minimum Spanning Tree's Edges are: \n\n");
100.    for (i = 0; i < spanlist.n; i++)
101.    {
102.        printf("%d. Edge (%d , %d) : %d\n", i+1, spanlist.data[i].u,
            spanlist.data[i].v, spanlist.data[i].w);
103.        cost = cost + spanlist.data[i].w;
104.    }
105.
106.    printf("\n\nMinimum Spanning Tree Cost: %d\n\n", cost);
107. }
108.
109. int main()
110. {
111.     n = 9;
112.
113.     Graph[0][0] = 0;
114.     Graph[0][1] = 4;
115.     Graph[0][2] = 0;
116.     Graph[0][3] = 0;
117.     Graph[0][4] = 0;
```



```
118.     Graph[0][5] = 0;
119.     Graph[0][6] = 0;
120.     Graph[0][7] = 8;
121.     Graph[0][8] = 0;
122.
123.
124.     Graph[1][0] = 4;
125.     Graph[1][1] = 0;
126.     Graph[1][2] = 8;
127.     Graph[1][3] = 0;
128.     Graph[1][4] = 0;
129.     Graph[1][5] = 0;
130.     Graph[1][6] = 0;
131.     Graph[1][7] = 11;
132.     Graph[1][8] = 0;
133.
134.
135.     Graph[2][0] = 0;
136.     Graph[2][1] = 8;
137.     Graph[2][2] = 0;
138.     Graph[2][3] = 7;
139.     Graph[2][4] = 0;
140.     Graph[2][5] = 4;
141.     Graph[2][6] = 0;
142.     Graph[2][7] = 0;
143.     Graph[2][8] = 2;
144.
145.
146.     Graph[3][0] = 0;
147.     Graph[3][1] = 0;
148.     Graph[3][2] = 7;
149.     Graph[3][3] = 0;
150.     Graph[3][4] = 9;
151.     Graph[3][5] = 14;
152.     Graph[3][6] = 0;
153.     Graph[3][7] = 0;
154.     Graph[3][8] = 0;
155.
156.
157.     Graph[4][0] = 0;
158.     Graph[4][1] = 0;
159.     Graph[4][2] = 0;
160.     Graph[4][3] = 9;
161.     Graph[4][4] = 0;
162.     Graph[4][5] = 10;
163.     Graph[4][6] = 0;
164.     Graph[4][7] = 0;
165.     Graph[4][8] = 0;
166.
167.
168.     Graph[5][0] = 0;
169.     Graph[5][1] = 0;
170.     Graph[5][2] = 4;
```

```
171.     Graph[5][3] = 14;
172.     Graph[5][4] = 10;
173.     Graph[5][5] = 0;
174.     Graph[5][6] = 2;
175.     Graph[5][7] = 0;
176.     Graph[5][8] = 0;
177.
178.
179.     Graph[6][0] = 0;
180.     Graph[6][1] = 0;
181.     Graph[6][2] = 0;
182.     Graph[6][3] = 0;
183.     Graph[6][4] = 0;
184.     Graph[6][5] = 2;
185.     Graph[6][6] = 0;
186.     Graph[6][7] = 1;
187.     Graph[6][8] = 6;
188.
189.
190.     Graph[7][0] = 8;
191.     Graph[7][1] = 11;
192.     Graph[7][2] = 0;
193.     Graph[7][3] = 0;
194.     Graph[7][4] = 0;
195.     Graph[7][5] = 1;
196.     Graph[7][6] = 0;
197.     Graph[7][7] = 0;
198.     Graph[7][8] = 7;
199.
200.
201.     Graph[8][0] = 0;
202.     Graph[8][1] = 0;
203.     Graph[8][2] = 2;
204.     Graph[8][3] = 0;
205.     Graph[8][4] = 0;
206.     Graph[8][5] = 0;
207.     Graph[8][6] = 6;
208.     Graph[8][7] = 7;
209.     Graph[8][8] = 0;
210.
211.
212.     kruskalAlgo();
213.     print();
214. }
```

- (i) Print the number of edges and total cost of Minimum Spanning Tree.

OUTPUT:

```

main.c [Lab_3_Kruskal] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
kruskalAlgo(): void

main.c
1 #include <stdio.h>
2 #define MAX 30
3
4 typedef struct edge
5 {
6     int u, v, w;
7 }
8 edge;
9
10 typedef struct edge_list
11 {
12     edge data[MAX];
13     int n;
14 }
15 edge_list;
16
17 edge_list elist;
18
19 int Graph[MAX][MAX], n;
20 edge_list spanlist;
21
22 void kruskalAlgo();
23 int find(int belongs[], int vertexno);
24 void applyUnion(int belongs[], int c1, int c2);
25 void sort();
26 void print();
27
28 // Applying Kruskal Algo
29 void kruskalAlgo()
30 {
31     int belongs[MAX], i, j, cno1, cno2;
32     elist.n = 0;
33     for (i = 1; i < n; i++)
34         for (j = 0; j < i; j++)
35             if (Graph[i][j] != 0)
36                 elist.data[elist.n].u = i;
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

"F:\Programs\CodeBlocks\Projects\Design and Analysis of Algorithm\Lab_3_Kruskal\bin\Debug\Lab_3_Kruskal.exe"
Minimum Spanning Tree's Edges are:
1. Edge (7, 5) : 1
2. Edge (6, 5) : 2
3. Edge (8, 2) : 2
4. Edge (1, 0) : 4
5. Edge (5, 2) : 4
6. Edge (3, 2) : 7
7. Edge (2, 1) : 8
8. Edge (4, 3) : 9

Minimum Spanning Tree Cost: 37

Process returned 0 (0x0)   execution time : 0.091 s
Press any key to continue.

```

(ii) **Analyze the time complexity of your algorithm.**

- The Time Complexity of Kruskal Algorithm is $O(E \log E)$.

AIM 3: To solve the given problem.

3. A thief enters a house for robbing it. He can carry a maximal weight of 60 kg into his bag. There are 4 items in the house with the following weights and values.

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24

1. Perform the 0-1 Knapsack and print the maximum weight and value of Knapsack.
2. Perform Fractional Knapsack and print the maximum value of Knapsack.

1. Perform the 0-1 Knapsack and print the maximum weight and value of Knapsack.

CODE:

```

1. #include <bits/stdc++.h>
2. using namespace std;
3. int max(int a, int b)
4. {
5.     return (a > b) ? a : b;
6. }
7. int knapSack(int W, vector<int>wt, vector<int>val, int n)
8. {
9.     if (n == 0 || W == 0)
10.        return 0;
11.     if (wt[n - 1] > W)
12.        return knapSack(W, wt, val, n - 1);
13.     else
14.        return max(val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),
15.                  knapSack(W, wt, val, n - 1));
16. }
17. // Driver code
18. int main()
19. {
20.     vector<int> profit= {280,100,120,120};
21.     vector<int> weight= {40,10,20,24};
22.     int W = 60;
23.     int n = profit.size();
24.     cout<<"Maximum Value of Knapsack is " << knapSack(W, weight, profit,
25. n)<<endl;
26.     return 0;
27. }

```

OUTPUT 1:

The screenshot shows a C++ IDE with the code from the previous block. The output window displays the result of the program execution:

```

Maximum Value of Knapsack is 400
Process returned 0 (0x0)   execution time : 0.101 s
Press any key to continue.

```

The status bar at the bottom indicates the file path: F:\Programs\CodeBlocks\Projects\Design and Analysis of Algorithm\Knapsack\main.cpp, and the current line is 18, column 2, position 429.

2. Perform Fractional Knapsack and print the maximum value of Knapsack.

```

1. #include<bits/stdc++.h>
2. using namespace std;
3. struct Item
4. {
5.     int profit;
6.     int weight;
7.
8. };
9. bool comp(Item a,Item b)
10. {
11.     double i1 = (double)a.profit/double(a.weight);
12.     double i2 = (double)b.profit/double(b.weight);
13.     return i1>i2;
14. }
15. double fractionalKnapsack(int W,Item arr[],int n)
16. {
17.
18.     sort(arr,arr+n,comp);
19.
20.
21.     int currentWeight=0;
22.     double finalProfit=0.0;
23.
24.     for(int i=0; i<n; i++)
25.     {
26.         if(currentWeight+arr[i].weight<=W)
27.         {
28.             currentWeight+=arr[i].weight;
29.             finalProfit+=arr[i].profit;
30.         }
31.         else
32.         {
33.             int remaining=W-currentWeight;
34.
35.             finalProfit+=(arr[i].profit*((double)remaining)/(double)arr[i].weight);
36.             break;
37.         }
38.     }
39.
40.     return finalProfit;
41.
42. }
43. int main()
44. {
45.
46.     int W=60;
47.     int n=4;
48.     Item arr[4]= {{280,40},{100,10},{120,20},{120,24}};
49.     cout<< "Maximum Value of Knapsack is " <<fractionalKnapsack(W,arr,n)
<<endl;

```

```

50.     return 0;
51. }

```

OUTPUT 2:

The screenshot shows a C++ IDE with the following code in `main.cpp`:

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  struct Item
4  {
5      int profit;
6      int weight;
7  };
8
9  bool comp(Item a,Item b)
10 {
11     double i1 = (double)a.profit/double(a.weight);
12     double i2 = (double)b.profit/double(b.weight);
13     return i1>i2;
14 }
15 double fractionalKnapsack(int W,Item arr[],int n)
16 {
17
18     sort(arr,arr+n,comp);
19
20     int currentWeight=0;
21     double finalProfit=0.0;
22
23     for(int i=0; i<n; i++)
24     {
25         if(currentWeight+arr[i].weight<=W)
26         {
27             currentWeight+=arr[i].weight;
28             finalProfit+=arr[i].profit;
29         }
30         else
31         {
32             int remaining=W-currentWeight;

```

The output window shows the following text:

```

Maximum Value of Knapsack is 440
Process returned 0 (0x0)   execution time : 0.090 s
Press any key to continue.

```

Link: https://github.com/rgautam320/Design-and-Analysis-of-Algorithm-Lab/tree/master/Lab_3_Greedy