SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – III

GREEDY TECHNIQUE

Topic

Greedy Technique: Kruskal's algorithm

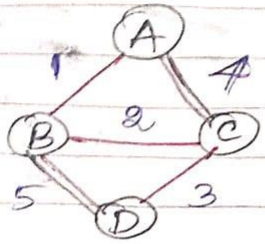
Kruskal's Algorithm

Kruskal Algorithm find MST.

Steps to find MST:

1. Sort all edges in increasing order by weight.
2. Pick Smallest edge. check any cycles are formed.
3. If cycle is formed, Discard it.
4. If cycle is not formed, include that edge in MST.
5. Repeat until (n-1) edges are formed.

Example :



Finding MST : (By Kruskal method)

Step 1: Sort the edges by their weight in Ascending order.

* A – B : 1
* B – C : 2
* C – D : 3
* A – C : 4
* B – D : 5

Step 2: Add edges to MST that forms No cycle.

A – B : 1
B – C : 2
C – D : 3

4 Vertices
edges = |V| – 1 = 4 – 1 = 3
Three edges with no cycles formed.

Step 3: Add the weight : 1 + 2 + 3 = 6

MST for the graph = 6

Example : 2

Graph 'G' : Vertices A, B, C, D, E

Edges 'E' : A-B

(i) (A-B ; 2)

(ii) (A-C, 3)

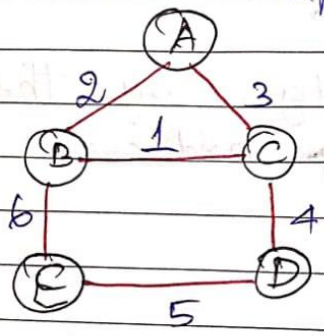
(iii) (B-C, 1)

(iv) (C-D, 4)

(v) (D-E, 5)

(vi) (B-E, 6)

Step1: Form the Graph:



Step2: Add Edges to MST one by one, that forms No cycle in Ascending order.

(i) B-C : 1

(ii) A-B : 2

(iii) A-C : 3

(iv) C-D : 4

(v) D-E : 5

(vi) B-E : 6

Step 3: Add Edges to MST one by one that forms no cycle.

(i). B-C : No cycle ; Add to MST

(ii) A-B : No cycle ; Add to MST

(iii) A-C : cycle formed i.e. A is already Connected to C through (A-B) & (B-C). So Skip this edge.

(iv) C-D : No cycle, Edge added to MST

(v) D-E : No cycle, Edge added to MST

(vi) B-E : Cycle formed (B-A-C-D-E) So skip edge.

Final MST :

* Edges 'E' : (B-C ; 1), (A-B, 2), (C-D, 4) (D-E, 5)

∴ Total weight = (1 + 2 + 4 + 5) = 12

5 vertices & 4 edges formed.

Thus MST for Graph is created.

Algorithm :

```
Algorithm kruskal ()
// problem description : Find MST using kruskal's
// Input : Graph 'G'
// output : MST with its total cost.
count = 0
k = 0
sum = 0
for (i = 0 to totnodes)
parent [i] = i
while (count != totnodes - 1) do
{
pos = minimum ( totedges ); // Find min cost edge
if (pos == 1)
break;
v1 = G [pos]. v1
v2 = G [pos]. v2
i = find ( v1, parent)
j = find (v2, parent)
if ( i != j) then
{
tree [k][0] = v1 // stores edges of spanning tree [][]
// (storing min edge in tree[]).
tree [k][1] ← v2
k++
count ++ ;
sumt = G[pos]. cost // computing total cost of all the minimum distance.
```

// accumulating total cost of MST
union (i, j, parent)

if (cost = INFINITY

if (count = totnodes - 1) then

for (i := 0 to totnodes - 1) do

{
write (tree [i][0], tree [i][1])

}
write ("cost of spanning Tree ", sum)

}

Analysis: Time complexity:

1. Sorting all Edges : O(E log E) times
2. Union - Find operations : O(E log V) times
3. Final complexity :

$O(E \log E) + E \log V) = O(E \log E)$

∵ E log E dominates E log V.

Best & Average case:

$O(E \log E)$

Worst case:

$O(E \log V)$ (occurs when sorting is dominant)

Space Complexity:

1. Store E edges → O(E)
2. Store V vertices → O(V)
3. MST stores V-1 edges → O(V)

Total space complexity = $O(V + E)$