45) Introduction to Greedy Technique & Prims Algorithm to find MST

Solution to a problem
- Feasible - A solution that satisfies a given constraint
- Optimal - A feasible solution for which the optimisation function has the best possible value

In greedy technique, the algorithm always make a choice that looks best at that moment. The choice made in each step should have the following requirement

1) Feasible - The problem should satisfy the problem constraints

2) Locally optimal - It has to be the best choice among all feasible solutions

3) Irrevocable - Decision once made can't be changed (revoked) in the subsequent steps

A general algorithm for solving a problem using greedy technique

Algorithm Greedy (A, n)

// Solves a problem using greedy method

// A is the input, which is a list of n elements

Solution ← ϕ

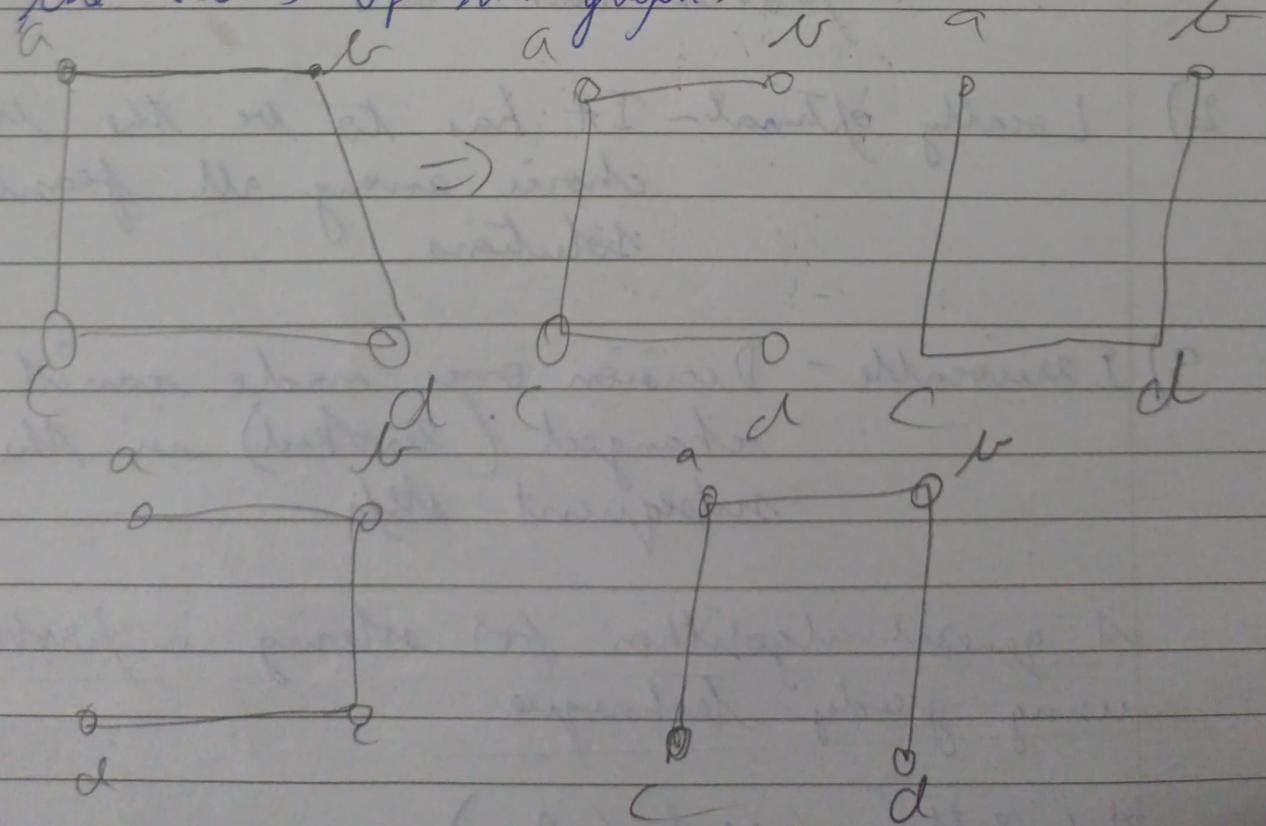
for i ← 1 to n-1
    x = choose(A)
    if (feasible(x))
        solution = Union(Solution, x)

Return Solution

Spanning Tree

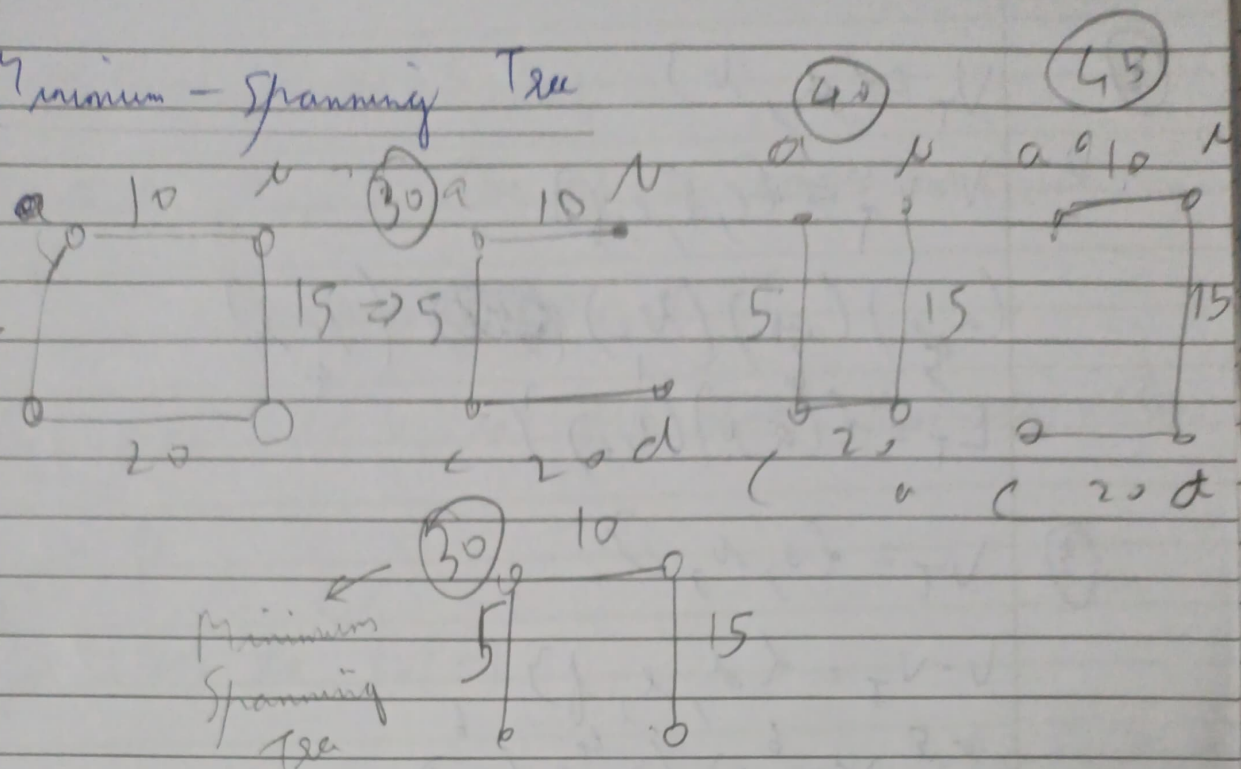
A spanning tree of a connected graph is a connected acyclic subgraph which contains all the vertex of the graph.

Eg:-

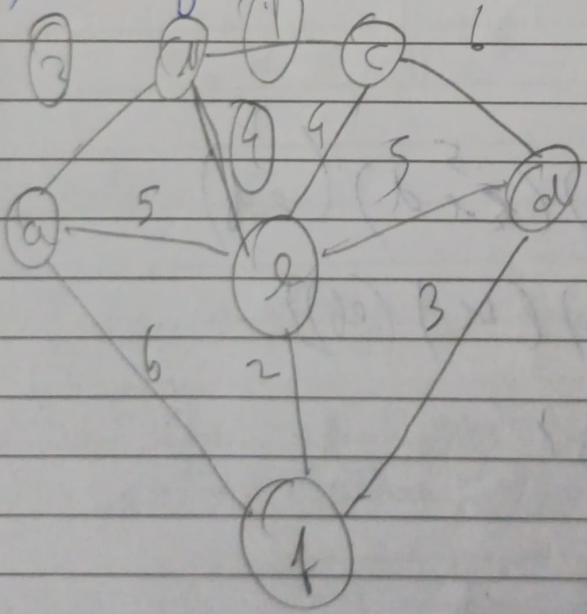


Minimum - Spanning Tree

Eg:-



A spanning tree with the minimum weights where weight is defined as the sum of weights of the edges used in the construction of spanning tree



Prims algorithm → Goal is to construct MST (min. spanning Tree)

$V = \{a, b, c, d, e, f\}$

$E_T = \phi$ (Should contain all edges used in the construction of MST)

① $V_T = \{a\}$

$V - V_T = \{b, c, d, e, f\}$

$E_T = \{(a,b)\}$

(ab) (ae) (af)
3 5 6

(2) $V_T = \{a, b\}$

$V - V_T = \{c, d, e, f\}$

$(a, e)\ (a, f)\ (b, c)$ [illegible] (b, e)

5 6 1 4

$E_T = \{(ab)(bc)\}$

(3) $V_T = \{a, b, c\}$

$V - V_T = \{d, e, f\}$

5 6 4 6

$(ae)(af)(be)$ [illegible]

$E_T = \{(ab)(bc)(bd)\}$

2

(4) $V = \{a, b, e, f\}$

$V - V_T = \{d, f\}$

[illegible] $(af)(cd)(ed)(ef)$

6 6 5 2

$E_T = \{(ab)(bc)(be)(ef)\}$

(5) $V = \{a, b, c, e, f\}$

$V - V_T = \{d\}$

6 5 3

[illegible] $(cd)(ed)(fd)$

$E_T = \{(ab)(bc)(be)(ed)(fd)\}$

Algorithm Prims (C_1)

// output: E_T, i.e., the set of edges used in construction of MST

$V_T = \{v_0\}$ // v_0 is the start (source) vertex

$E_T = \emptyset$

for $i \in 1$ to $|V| - 1$

find a minimum weight edge $e^* = (v^*, u^*)$ among all the edges (v, u) such that v is in V_T & u is in $V - V_T$

$V_T \leftarrow V_T \cup u^*$

$E_T \leftarrow E_T \cup e^*$

return E_T

Time complexity of Prim's Algorithm

1) If the input graph G is represented as a weighted matrix, efficiency is $O|V|^2$

2) If the input graph is represented through adjacency list and a priority queue is implemented as a min-heap (on edges), the running time is $O(|E| \cdot \log |V|)$

we need to perform $|V|-1$ deletions of the smallest edge (element) & make $|E|$ verification & possibly changes the heap

$$\downarrow$$

$$O(\log |V|)$$

$$|V|+1+|E| \log |V| = O(|E| \log |V|)$$

46 C Program to find Minimum Spanning Tree using Prims Algorithm

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i, j, n, cost[10][10];
    printf("\n Read no. of nodes : ");
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
        {
            scanf("%d", &cost[i][j]);
            if (cost[i][j] == 0)
                cost[i][j] = 999;
        }
    }
    prims(n, cost);
    return 0;
}
```

```
void prims(int n, int cost[10][10])
{
int i, j, u, v, min, mincost=0, visited[10],
ne=1;
for(i=1;i<=n;i++)
    visited[i]=0;
printf("\n The edges considered for MST
are \n");
visited[1]=1;
while(ne<n)
{
  for(i=1,min=999;i<=n;i++)
      for(j=1;j<=n;j++)
        if(cost[i][j]<min)
        {
          if(visited[i]==0)
            continue;
          else
          {
            min=cost[i][j];
            u=i;
            v=j;
          }
        }
  if(visited[u]==0||visited[v]==0)
  {
    printf("%d Edge (%d,%d)=%d",ne++,
    u,v,min);
    mincost=mincost+min;
    visited[v]=1;
  }
  cost[u][v]=cost[v][u]=999;
}printf("\n Cost of constructing MST is %d",
```

mincost(c);
}

So Huffman Coding

↓

data encoding ← Data Compression (Reduce the size)

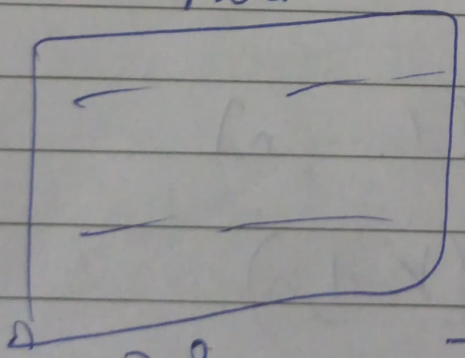
Data integrity

Encoding are of 2 types

1) Fixed length encoding

2) Variable length encoding

1)



2) With encoding is 8 X 20 = 160 bits

Out

With encoding (fixed) = 3 X 20 = 60 bits

2) Huffman coding is an example for variable length encoding

↓

We take in account the frequency of characters

↓

Higher frequency characters will have lesser size

Huffman coding algorithm

↓

We build a binary tree which gives the codeword for the characters

1) Initialise n (no. of characters to be encoded) 0-1 tree (binary tree - left child marked with 0 & right child is marked with 1) and label with the appropriate characters as well as record the frequency of that character in the tree root to indicate the tree weight

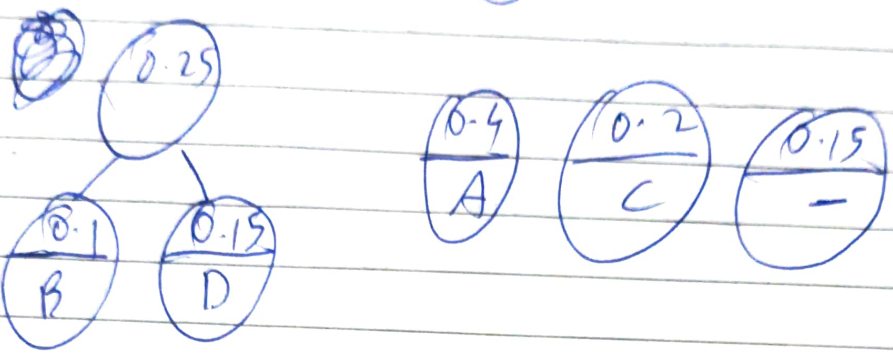
2) Repeat the following until a single binary tree is obtained - find 2 trees with smallest weight & create a 0-1 tree with the smallest as left child and second smallest as right child. Record the sum of weights in the root of the new tree

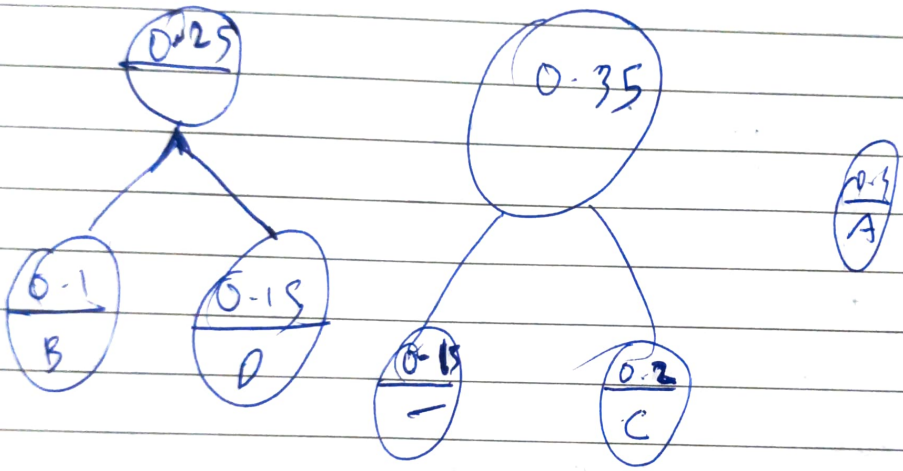
20 characters

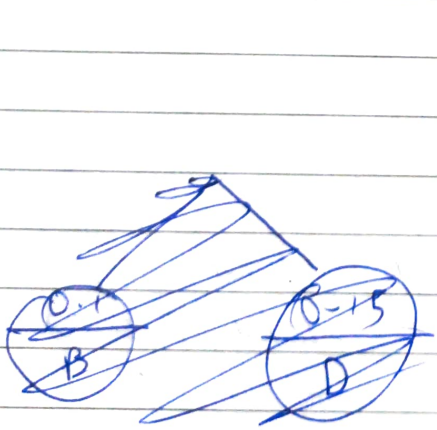
A = 8 = 40%

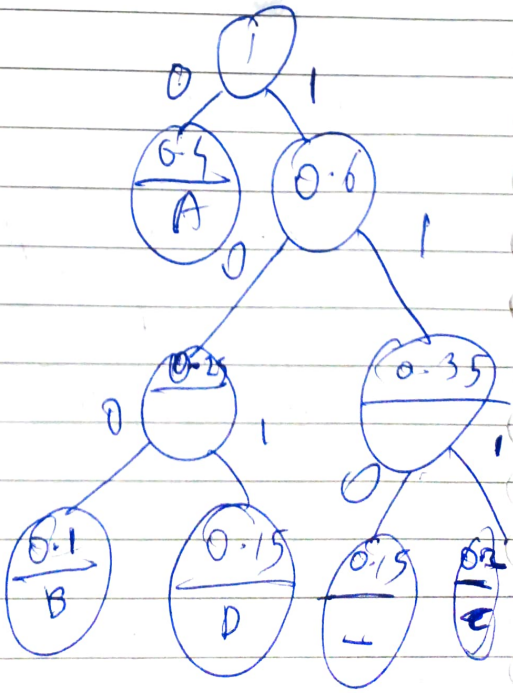
B = 2 = 10%

C = 4 = 20%
D = 3 = 15%
– = 3 = 15%









A = 0
B = 100
C = 111
D = 101
< = 110

No. of

Avg no. of bits = 1×0.4 + 3×0.1 + 3×0.2 + 3×0.15 + 3×0.15.

= 0.4 + 0.3 + 0.6 + 0.45 + 0.45

= 2.2

Compression ratio = $\left(\frac{3 - 2.2}{3}\right) \times 100$ = 26%.

1) Encode BAD - DAA

1000101 | 1010 | 00