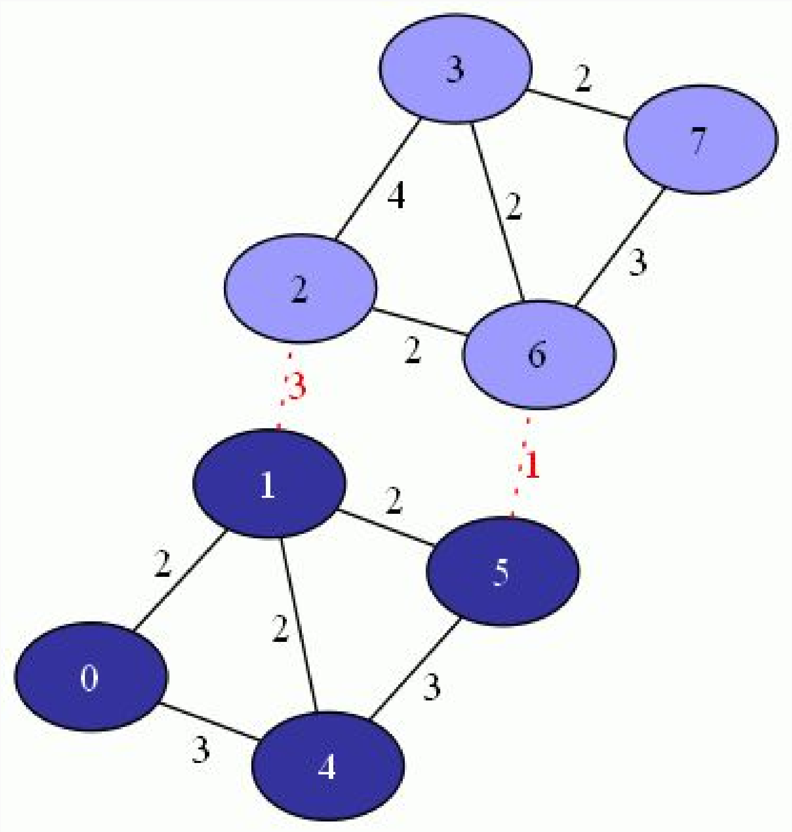
Given a Graph, Find the Minimal Cutset

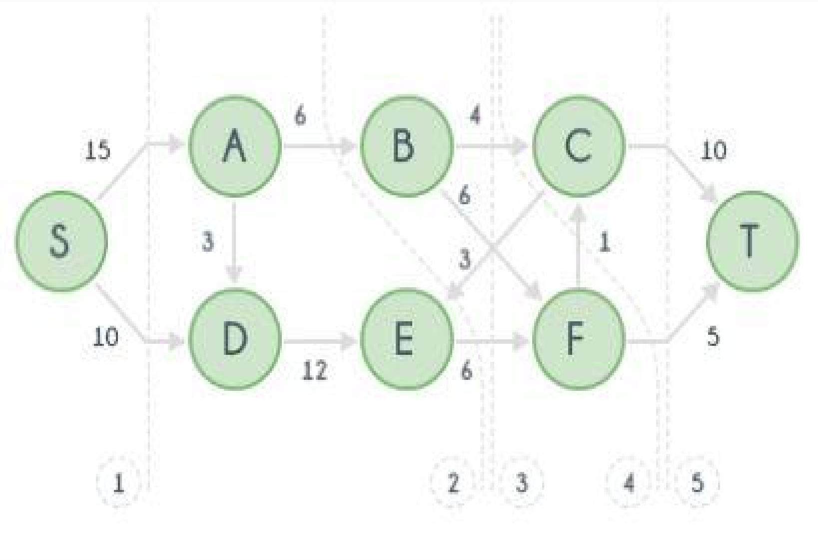
Ajay Saini (IRM2014003)
Ayonya Prabhakaran(IHM2014502)
Ankit Mund (IIT2013169)

Problem

- A graph is given. We have to find minimal cutset in that graph.
- A cut is a partition of the vertices of a graph into two disjoint subsets.
- Min-cut: Min-Cut of a weighted graph is defined as the minimum sum of weights of (at least one) edges that when removed from the graph divides the graph into two groups.
- For an unweighted graph, the minimum cut would simply be the cut with the least edges.

Example





Cut1:25,Cut2:15,Cut3:19,Cut4:10,Cut5:15

Algorithm 1 Karger’s algorithm

1. Initialize contracted graph CG as copy of original graph.
2. While there are more than two vertices.

   a) Pick a random edge (u, v) in the contracted graph.

   b) Merge (or contract) u and v into a single vertex (update the contracted graph).

   c) Remove self-loops

3. Return cut represented by two vertices.

Pseudocode 1 Karger's algorithm

```
Procedure Main
        Input graph G
        Print( kargerMinCut(G) )

Function kargerMinCut (graph)
        V = graph->V, E = graph->E
        Initialize 2d array of size V: subsets
        for v=0 to V
                subsets[v].parent = v
                subsets[v].rank = 0
        vertices = V
        while vertices>2 do
                i = rand() % E
                subset1 = find(subsets, edge[i].src)
                subset2=find(subsets,  edge[i].dest)
                if (subset1 == subset2)
                        continue
                else
                    vertices--
                    Union(subsets, subset1, subset2)
        cutedges = 0
        for  i=0 toE
            subset1 = find(subsets, edge[i].src)
            subset2 = find(subsets, edge[i].dest)
            if (subset1 != subset2)
                  cutedges++
        return cutedges

function find(subset, i)
        if (subsets[i].parent != i)
                Subsets[i].parent=find(subsets,
                subsets[i].parent)
  return subsets[i].parent

function Union(subsets, x,  y)
   x = find(subsets, x)
   y = find(subsets, y)
   if (subsets[x].rank < subsets[y].rank)
         subsets[x].parent = y
   else if (subsets[x].rank) >    subsets[y].rank)
         subsets[y].parent = x
    else
         subsets[y].parent = x
         Subsets[x].rank++
```

Time complexity: Karger's algorithm can be implemented in $O(E) = O(V^2)$ time.

Algorithm 2 Stoer–Wagner algorithm

MinCut(G,w):

    w(minCut) ← ∞

    While |V| > 1

        s-t-phaseCut ← MinCutPhase(G,w)

        if w(s-t-phaseCut) < w(minCut)

            minCut ← s-t-phaseCut

    Return minCut

MinCutPhase(G, w):

    a ← arbitrary vertex of G

    A ← (a)

    While A ≠ V

        v ← vertex most tightly connected to A

        A ← A U (v)

    s and t are the last two vertices (in order) added to A

    Merge(G,s,t)

    Return cut(A-t,t)

Pseudocode 2 Stoer–Wagner algorithm

```
int minCutPhase(int V){
	int i = 0, j = 0;
	int s[2];
	if(V  == 2) {
		for( i = 0; i  <  n;  i++){
			if(Del[i] == FALSE){
				s[j] = i; j++;
			}
		}
		return W[s[0]][s[1]];
	}
	int L[n], T[n];
	memset(L, 0, n*sizeof(int));
	memset(T, FALSE, n*sizeof(int));
	i = 1;  // the number of vertices in the tree T
	j = 0;
	int v,u;
	while( i <= V){
		v = maxStickiness(T,L);
		T[v] = TRUE;
		for(u = 0; u < n; u++){
			if(W[v][u] != 0 && Del[u] == FALSE && T[u] ==
FALSE){
				L[u] = L[u] + W[u][v];
			}
		}
		if( i >= V-1){
			s[j] = v;  j++;
		}
		i++;
	}
	merge(s[0], s[1]);
	return L[s[1]];
}
```

```
void merge(int s, int t){
	int v = 0;
	for(v = 0; v < n; v++){
		if(Del[v] == FALSE && v != s && v!= t){
			W[s][v] = W[s][v] + W[v][t];
			W[v][s] = W[s][v];
		}
	}
	Del[t] = TRUE;
}

int maxStickiness(int *T, int *L){
	int i = 0;
	int v = 0;
	int max = 0;
	for(i = 0; i < n; i++){
		if(Del[i] == FALSE && T[i] == FALSE && max < L[i]){
			v = i;
			max = L[i];
		}
	}
	return v;
}
int StoerWagner(){
	int V  = n;
	int C = INFTY;
	memset(Del, FALSE, n*sizeof(int));
	for(V = n; V > 1; V--){
		int cutValue = minCutPhase(V);
		C = (C < cutValue ? C: cutValue);
	}
	return C;
}
```

Time complexity:

For Stoer–Wagner algorithm

- For **MinCutPhase:**

a single run of it needs at most ***O(|E|+|V|log |V|)*** time.

- **MinCut:**

MinCut calls MinCutPhase V times

So running time will be

O(|V||E|+|V|^2log |V|)

Thank you