GOMORY-HU TREES : THEORY AND APPLICATIONS

Koutris Paraschos and Vasileios Syrgkanis

Outline

- Basic definitions
- Why needed?
- Gomory-Hu Construction Algorithm
- A Complete GH Tree Construction Example
- Proof Of Correctness
- Minimum K-Cut Problem
- Implementation

Basic Definitions

Cut Definition

- Let $G = (V,E)$ denote a graph and $c(e)$ a weight function on its edges.
- A **cut** is a partition of the vertices V into two sets S and T.
- Any edge $(u,v) \in E$ with $u \in S$ and $v \in T$ is said to be **crossing** the cut and is a **cut edge**.
- **The capacity of a cut** is the sum of weights of the edges crossing the cut.

U-V CUT

- A *u-v cut* is a split of the nodes into two disjoint sets U and V, such that u $\in$ U, v $\in$ V.

- **MINIMUM WEIGHT U-V CUT**

  Given a graph G = (V,E) and two terminals u,v $\in$ V, find the minimum u-v cut.

FLOW DEFINITION

- Given a directed graph *G(V,E)* in which every edge $(u,v) \in E$ has a **non-negative**, **real-valued** capacity $c(u,v)$.
- We distinguish two vertices: a **source s** and a **sink t**.
- A flow network is a real function $f: V \times V \rightarrow \mathbf{R}$ with the following properties for all nodes u and v:

1. **Capacity constraints**: $f(u,v) \leq c(u,v)$
2. **Skew symmetry**: $f(u,v) = - f(v,u)$
3. **Flow conservation**: ,unless u=s or u=t

MAX-FLOW

- The maximum flow problem is to find a feasible flow through a single-source, single-sink flow network that is maximum.
- Max-Flow can be computed in polynomial time (e.g. Edmonds-Karp algorithm).

- **MAX-FLOW MIN-CUT THEOREM**
  - The maximum amount of flow is equal to the capacity of the minimal cut.
  - Thus, the min s-t cut is also computed in polynomial time.

Important Publications On Max-Flow Min-Cut Problems

- Ford and Fulkerson, *Maximal Flow through a network* (1956).

  Introduction of basic concepts of flow and cut. Max flow min-cut theorem.

- Mayeda, *Terminal and Branch Capacity Matrices of a Communication Net* (1960).

  Multiterminal problem.

- Chien, *Synthesis of a Communication Net* (1960).

  Synthesis of multiterminal flow network.

WHY NEEDED ?

Basic Properties Of Cuts

- We are interested in finding maximal flow/minimal cut values between all pairs of nodes in a graph G = (V,E), where n = |V|. Any pair of nodes can serve as the source and the sink.

- How many min-cut computations are needed?

- You would think

- But in fact, n-1 computations are enough!!

  why? (PROOF #1)

Flow Equivalent Graphs

- Two graphs G = (V, E) and G' = (V, E') are said to be flow equivalent iff for each pair of vertices u,v ∈ V, the minimum u-v cut (maximal u-v flow) in G is the same as in G'.

- It turns out that there always exist a G' which is a tree (Gomory Hu Tree)!!

- Notice that the n-1 edges of the tree correspond to the n-1 distinct min-cuts in G.

Gomory-Hu (GH) Tree

R.E. Gomory and T.C. Hu, *Multi-terminal network flows* (1961).

- Given a graph G = (V,E) with a capacity function c, a cut-tree T = (V,F) obtained from G is a tree having the same set of vertices V and an edge set F with a capacity function c' verifying the following properties:
  1. **Equivalent flow tree**: for any pair of vertices s and t, $f_{s,t}$ in G is equal to $f_{s,t}$ in T , i.e., the smallest capacity of the edges on the path between s and t in T.
  2. **Cut property**: a minimum cut $C_{s,t}$ in T is also a minimum cut in G.

Gomory-Hu Construction Algorithm

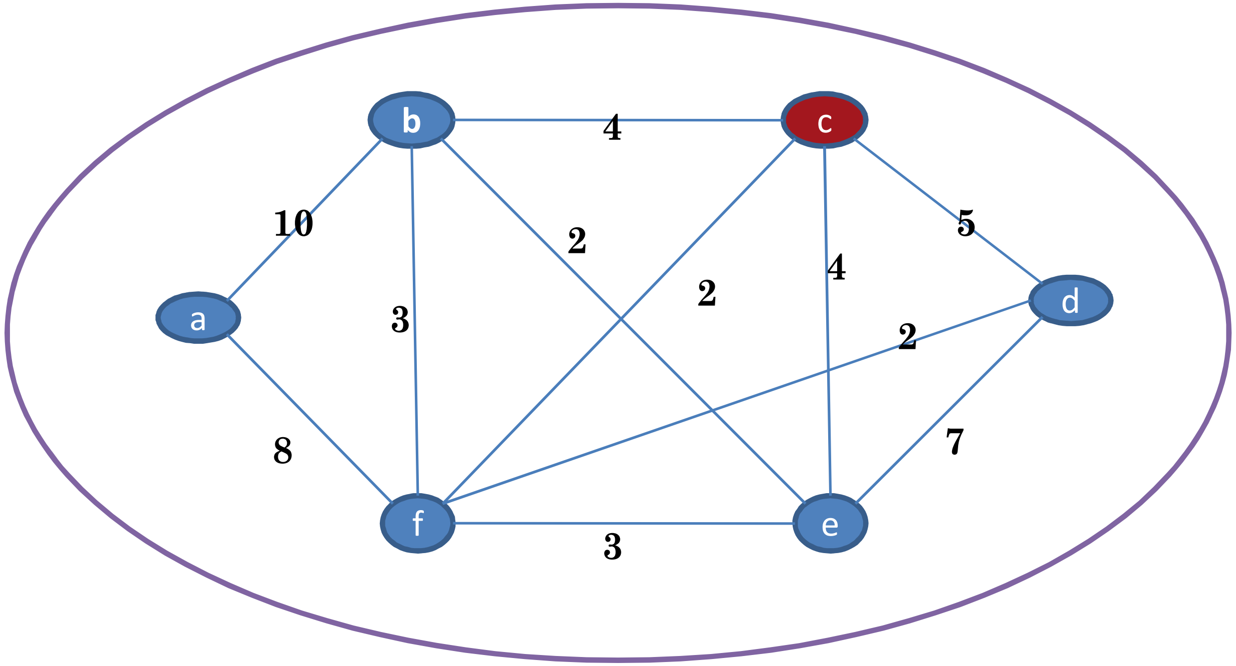
Outline

- The algorithm maintains a partition of V, (S_1, S_2, …, S_t) and a spanning tree T on the vertex set { S_1, S_2, …, S_t }.
- Let w' be the function assigning weights to the edges of T.
- On each iteration, T satisfies the following invariant :

  For any edge (S_i, S_j) in T, there are vertices a and b in S_i and S_j respectively such that w' (S_i, S_j) = f(a,b) and the cut defined by edge (S_i, S_j) is a minimum a-b cut in G.

INITIAL STEP

- The algorithm starts with a trivial partition V.
- Proceeds in n-1 iterations.

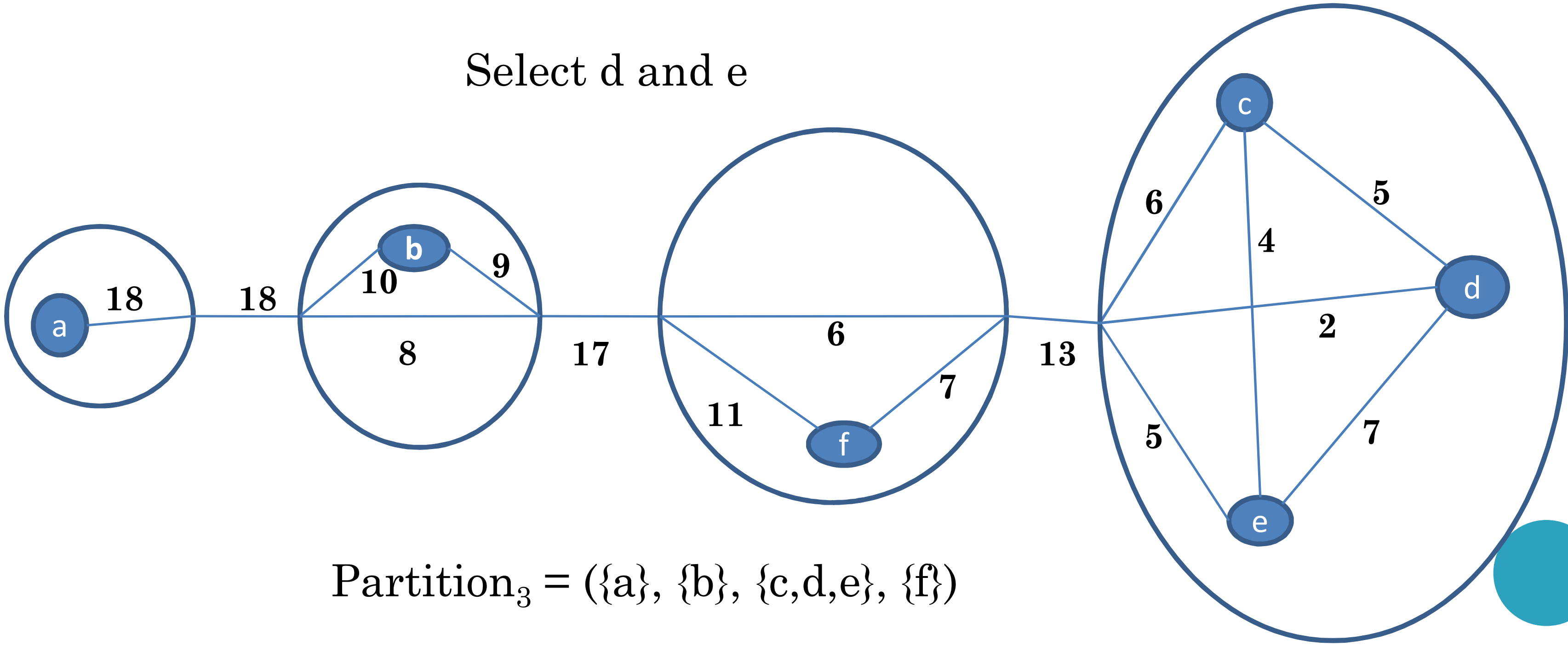


Initial Partition =
(V={a,b,c,d,e,f})

ITERATION (1)

- Select a set S_i in the partition such that $|S_i| \geq 2$.
- Let u and v be two distinct vertices of S_i.

Select d and e

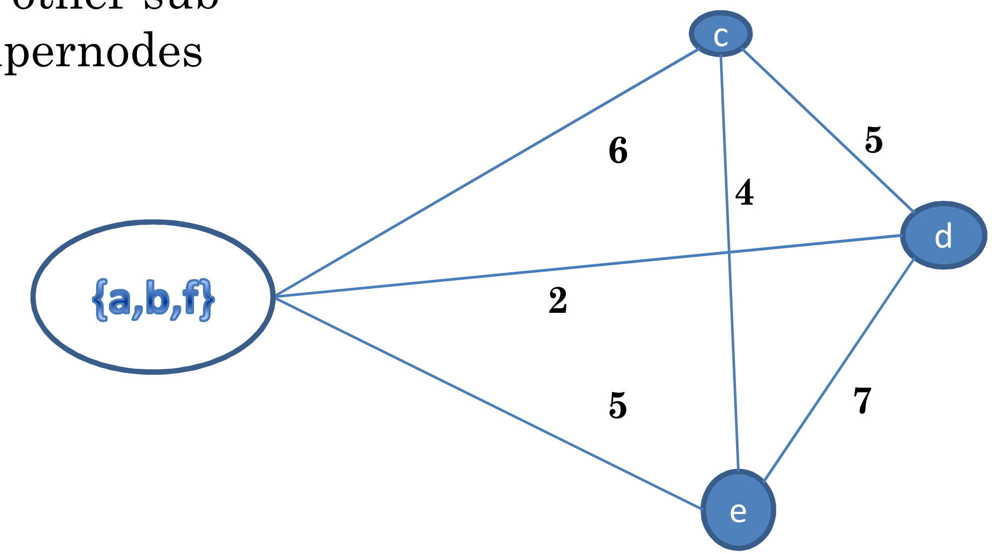


$\text{Partition}_3 = (\{a\}, \{b\}, \{c,d,e\}, \{f\})$

ITERATION (2)

- Root the current tree at S_i and consider the subtrees rooted at the children of S_i.
- Collapse each of the subtrees into a single vertex to obtain graph G' (G' also contains all vertices of S_i).

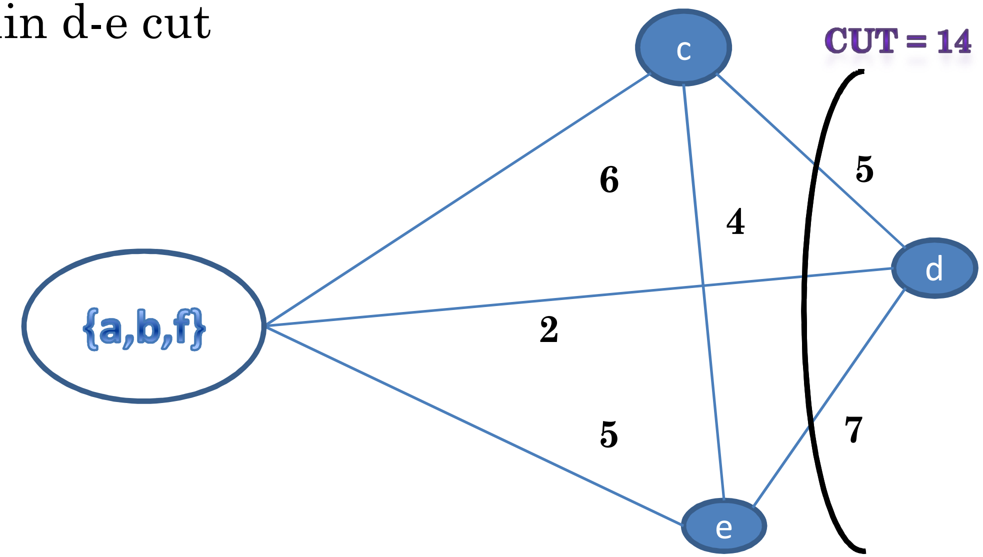
Collapse all other sub-trees to supernodes



ITERATION (3)

- Find a minimum u-v cut in G'.
- Let (A, B) the partition of the vertices of G' defining the cut, with u ∈ A, v ∈ B.

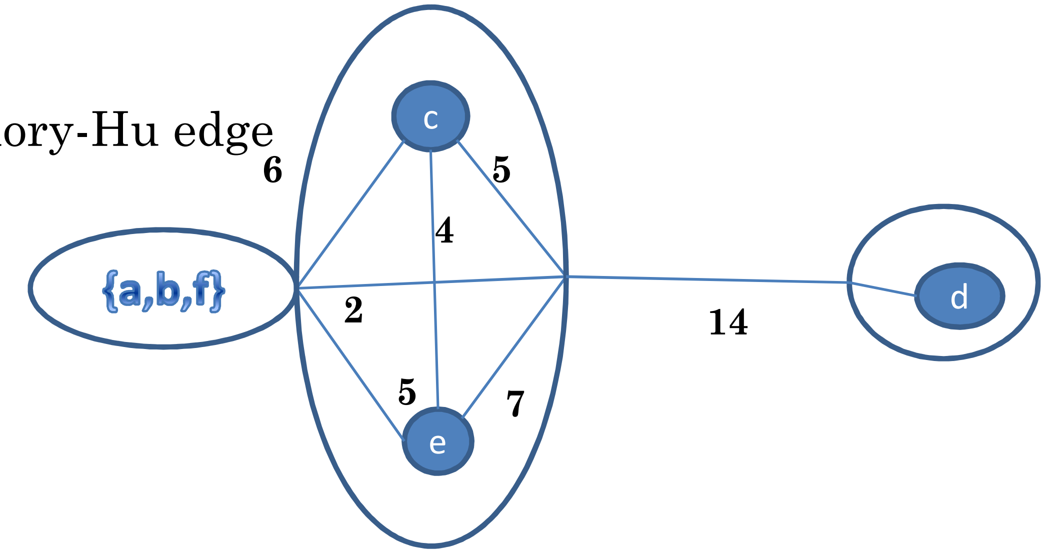
Compute min d-e cut



Iteration (4)

- Compute $S_i^u = S_i \cap A$ and $S_i^v = S_i \cap B$.
- Refine the current partition by replacing S_i with the two sets S_i^u and S_i^v.
- The new tree has an edge (S_i^u, S_i^v) with weight equal to the weight of the cut.

Create new Gomory-Hu edge



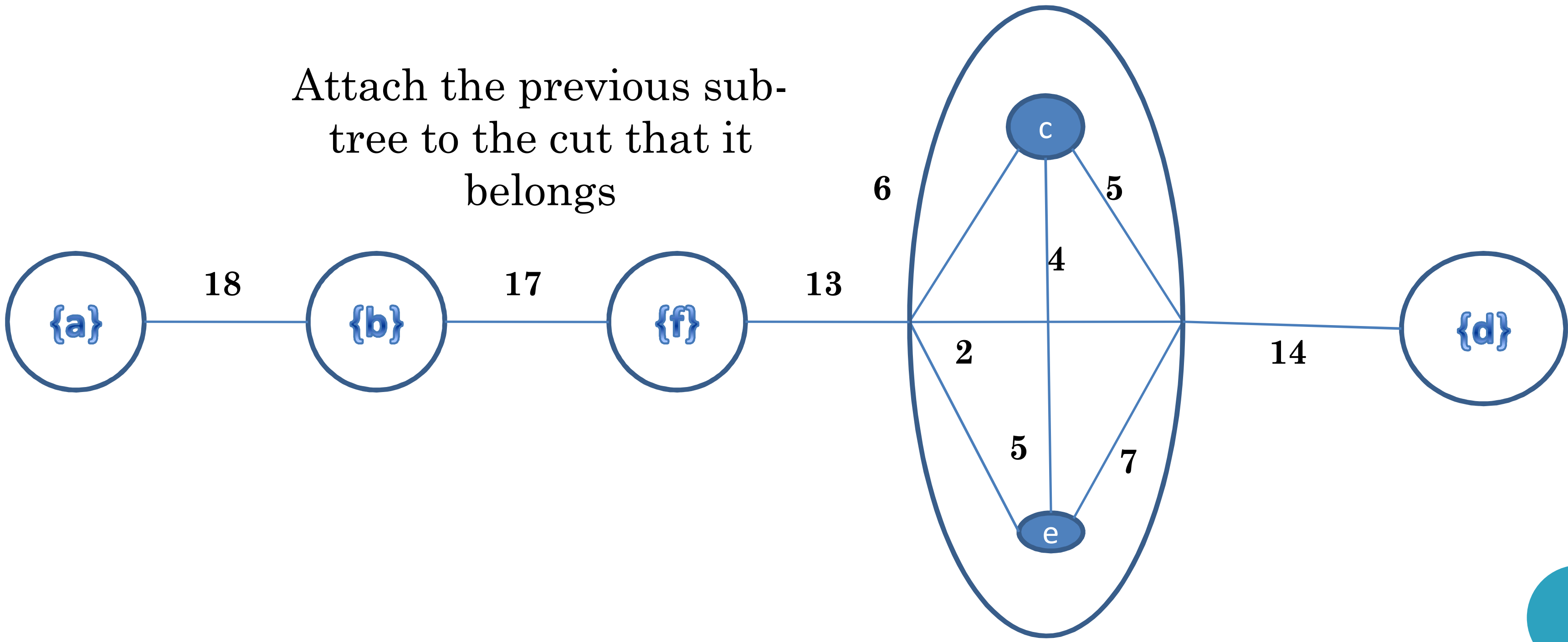
ITERATION (5)

- How are the other nodes aranged at the tree after the splitting?

- Consider a subtree T' incident at S_i in T. Assume that the collapsed node corresponding to T' lies in A.
- We connect T' by an edge with S_i^u.
- The weight of the edge is the same as the weight of the edge connecting T' to S_i.
- All the other edges retain their weights.

ITERATION (6)

Attach the previous sub-tree to the cut that it belongs

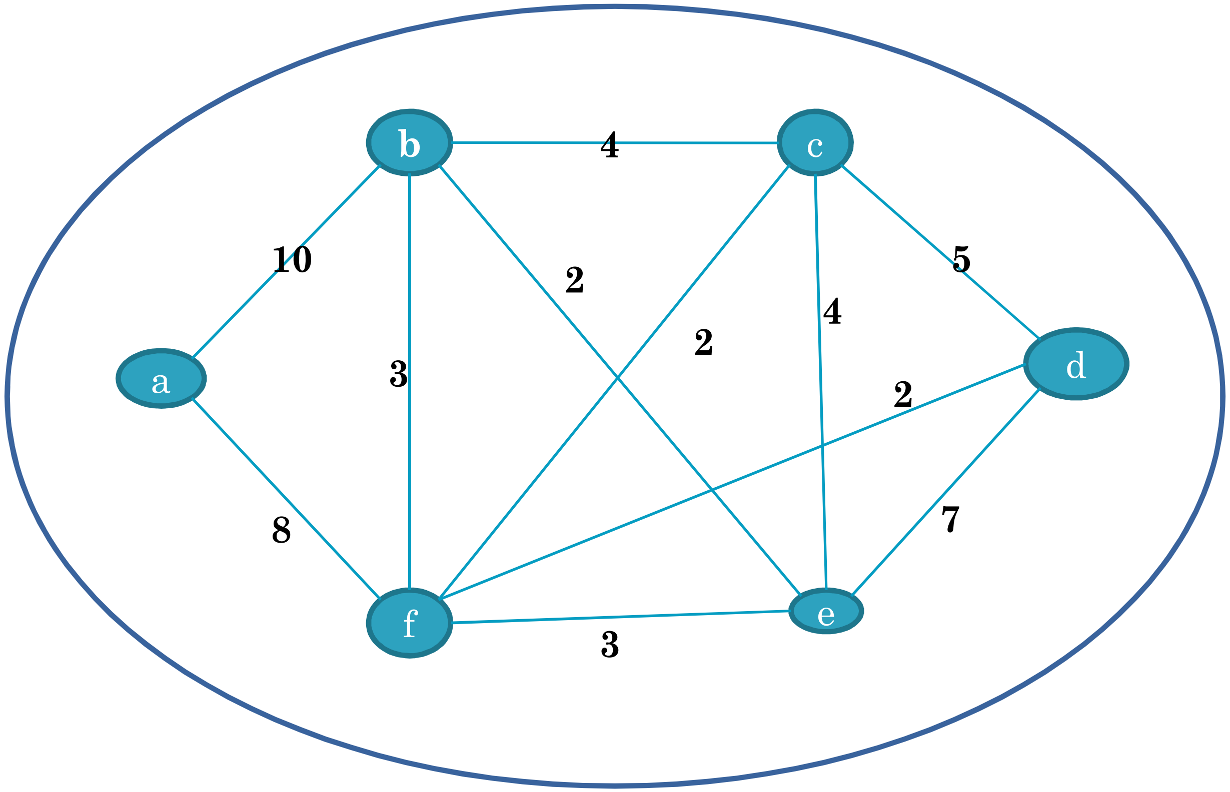


TERMINATION

- The algorithm terminates when the partition consists of singleton vertices.
- Thus, after exactly n-1 iterations!

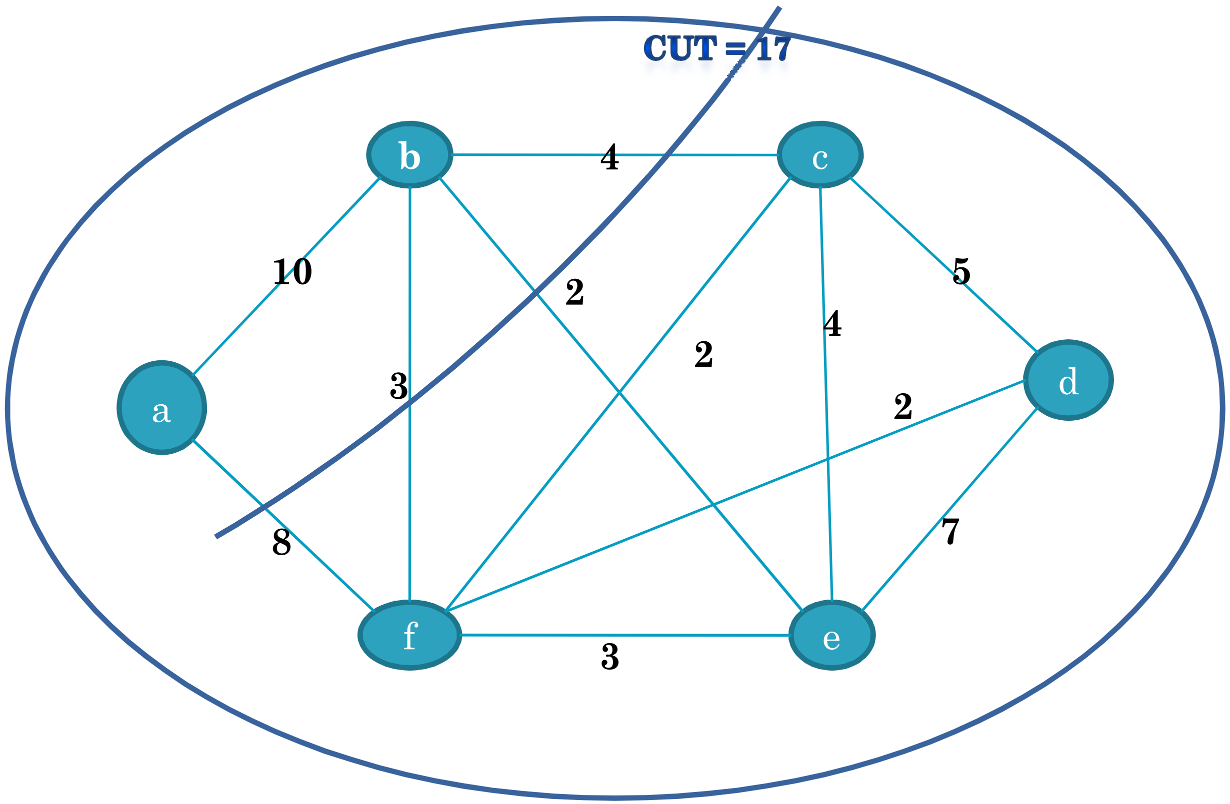
A Complete GH Tree Construction Example

INITIALIZATION



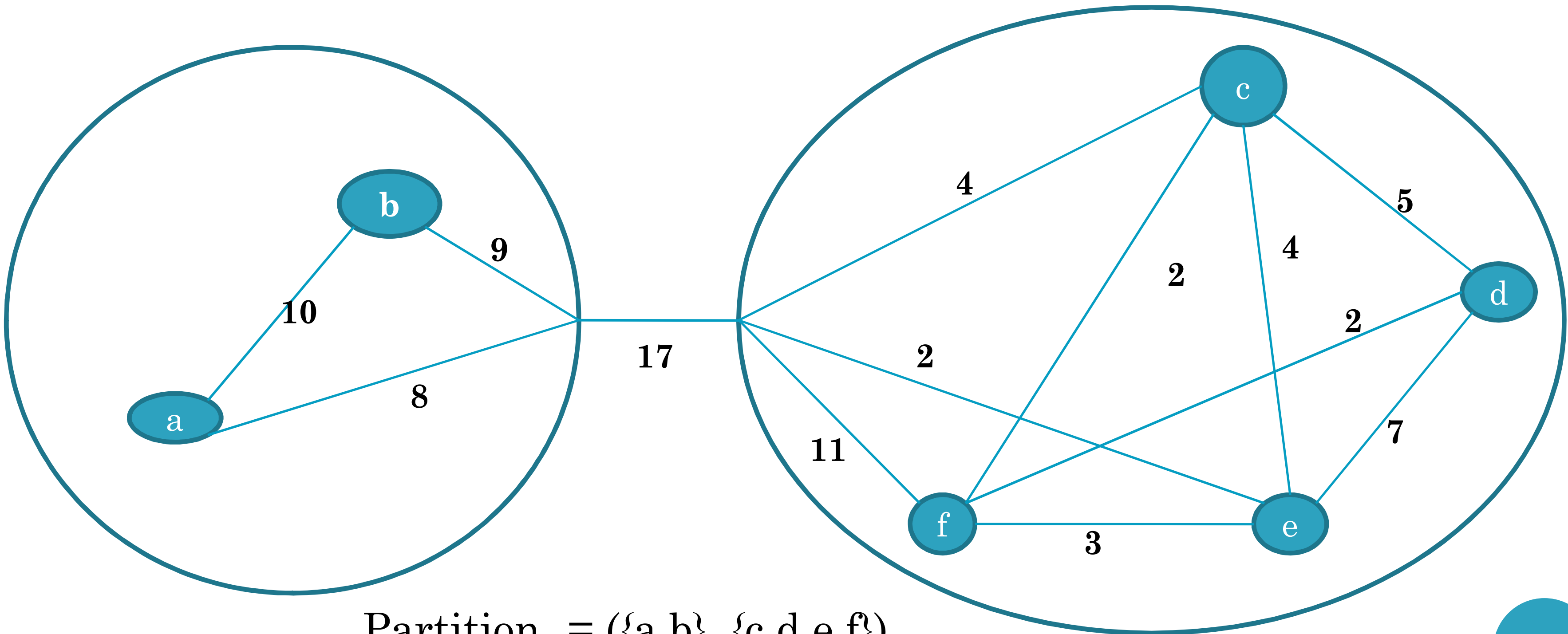
Initial Partition = (V={a,b,c,d,e,f})

ITERATION 1



Select b and f

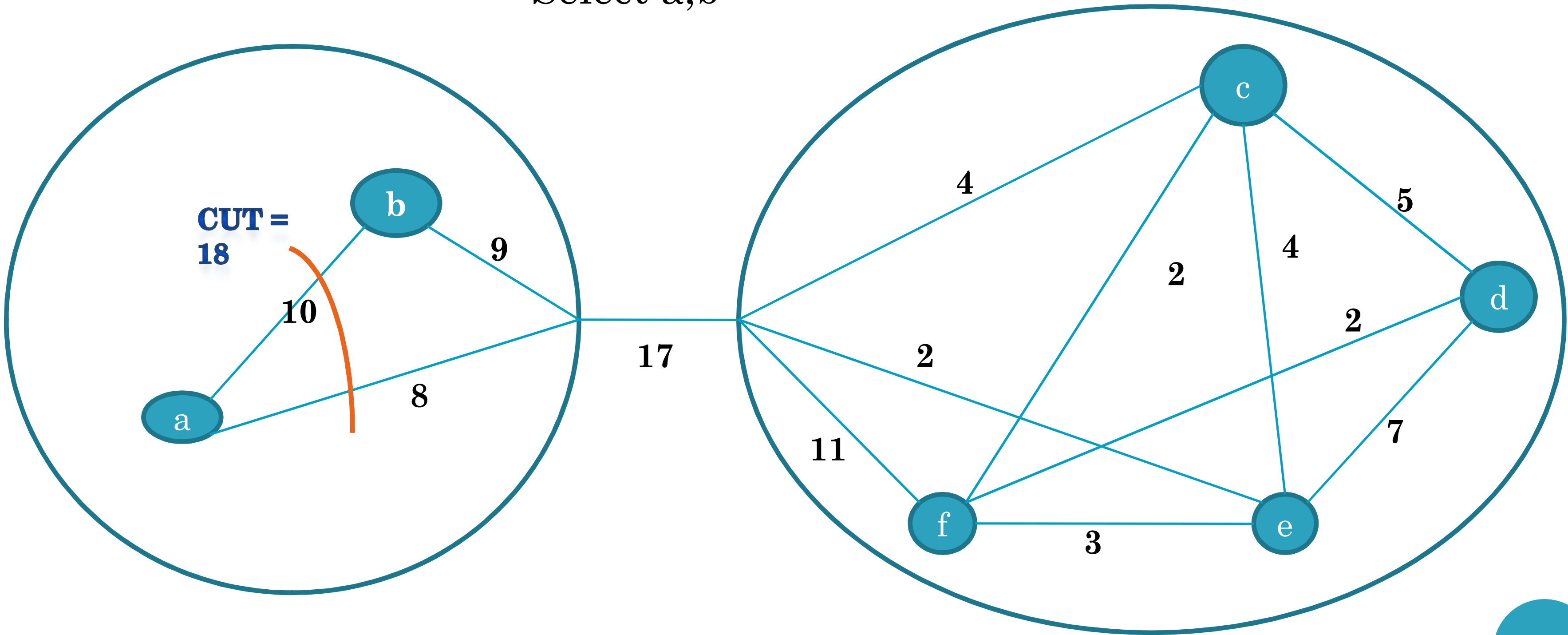
Iteration 1



Partition$_1$ = ({a,b}, {c,d,e,f})

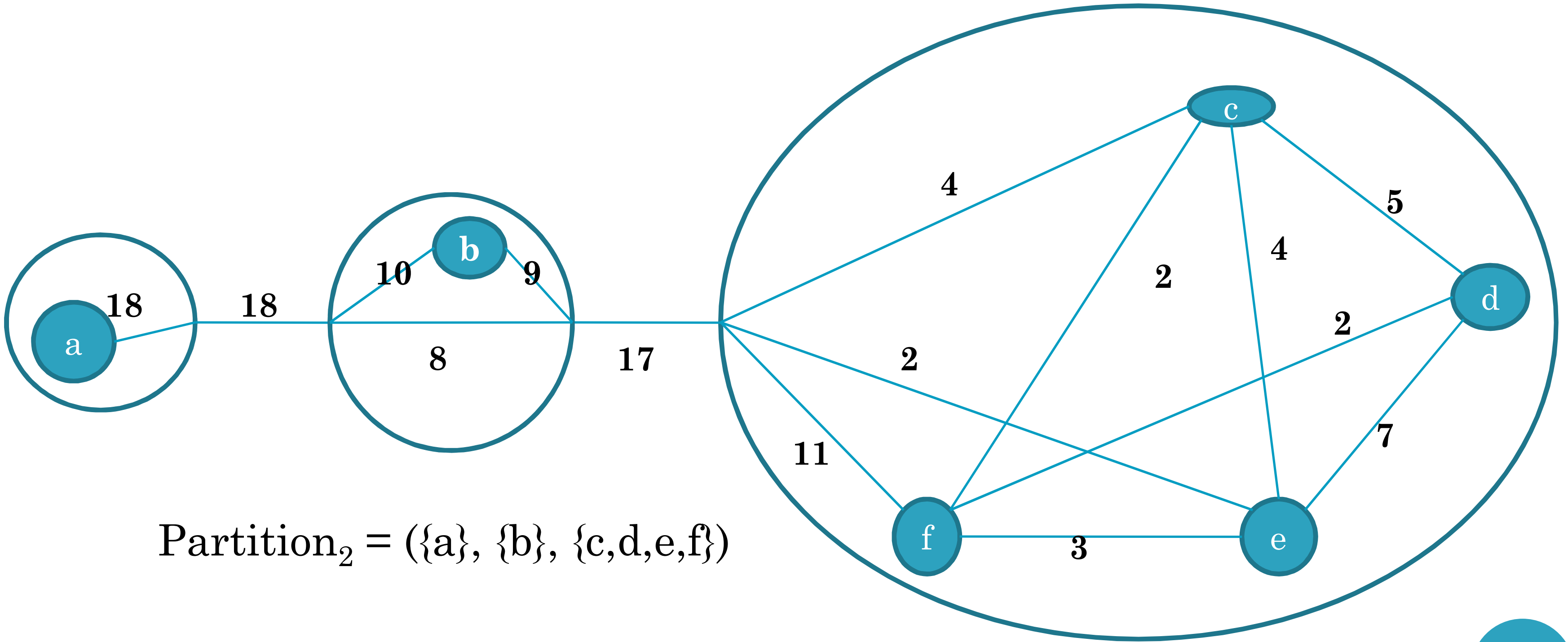
ITERATION 2

Select a,b



Partition$_1$ = ({a,b}, {c,d,e,f})

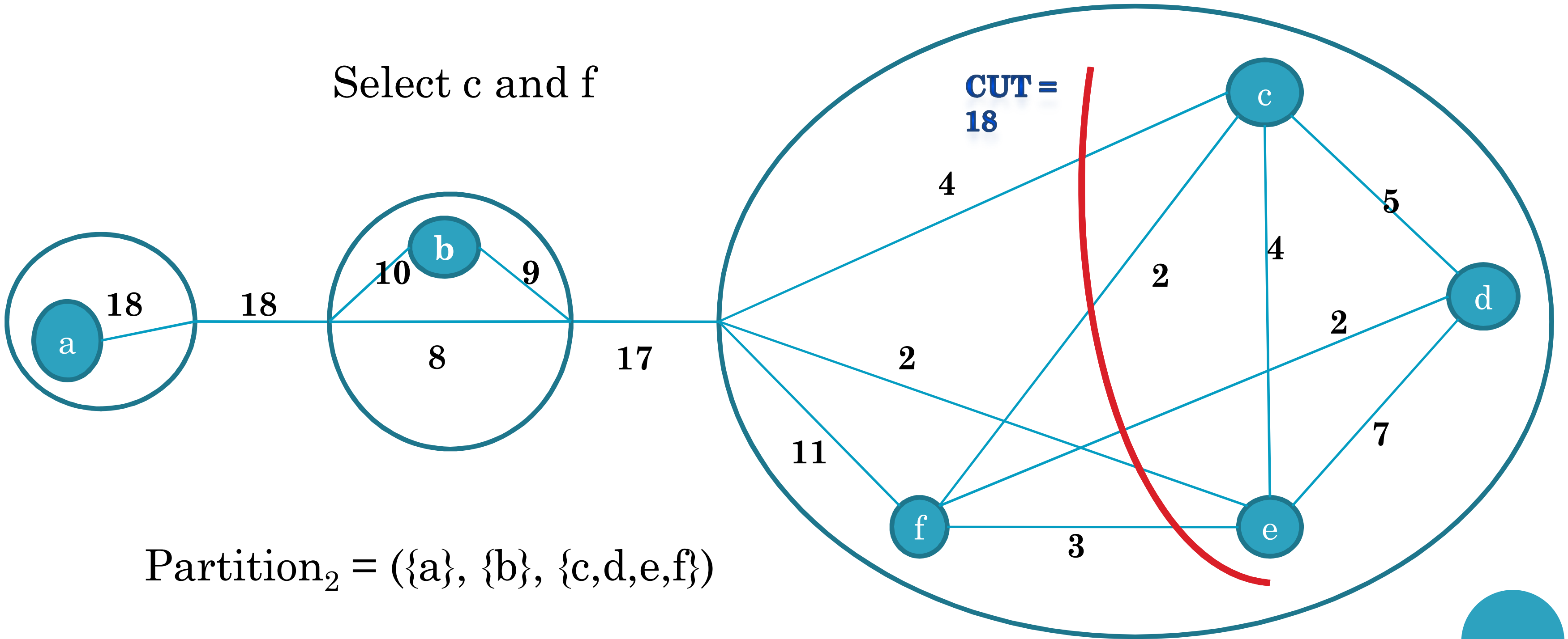
ITERATION 2



Partition$_2$ = ({a}, {b}, {c,d,e,f})

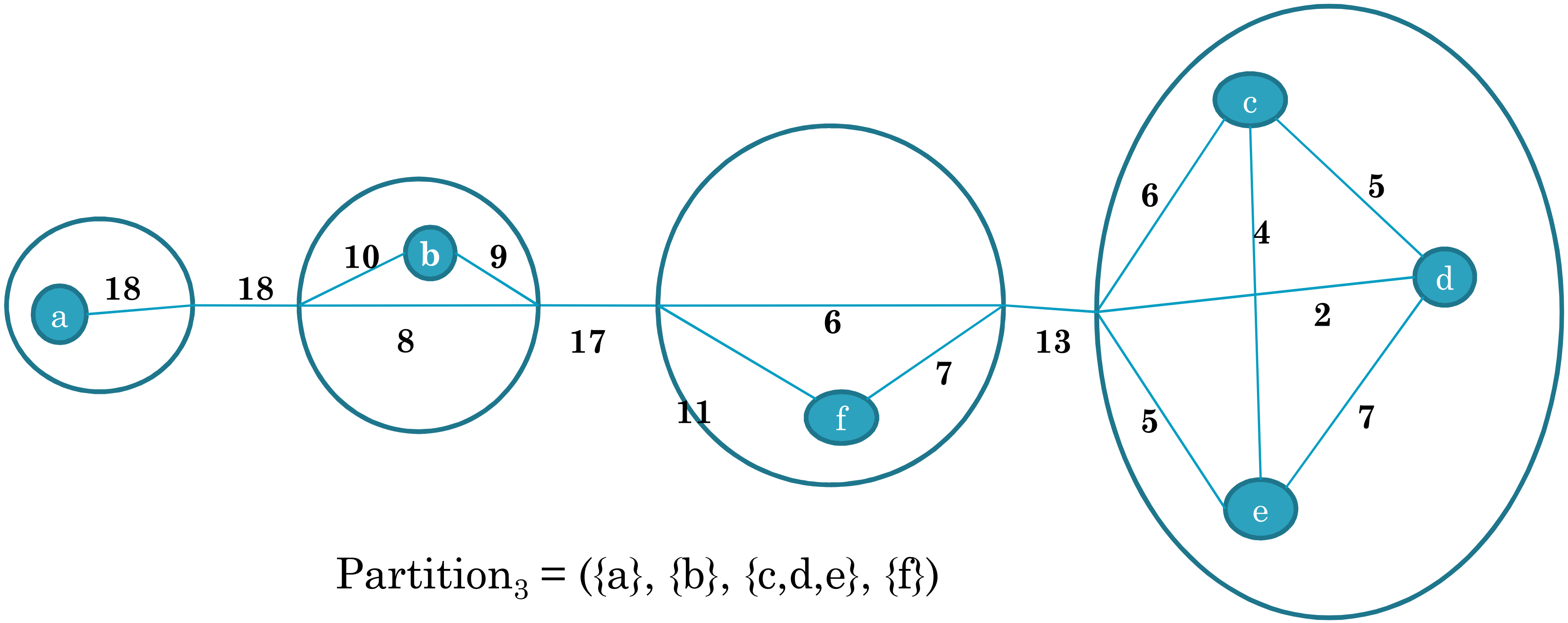
ITERATION 3

Select c and f



Partition$_2$ = ({a}, {b}, {c,d,e,f})

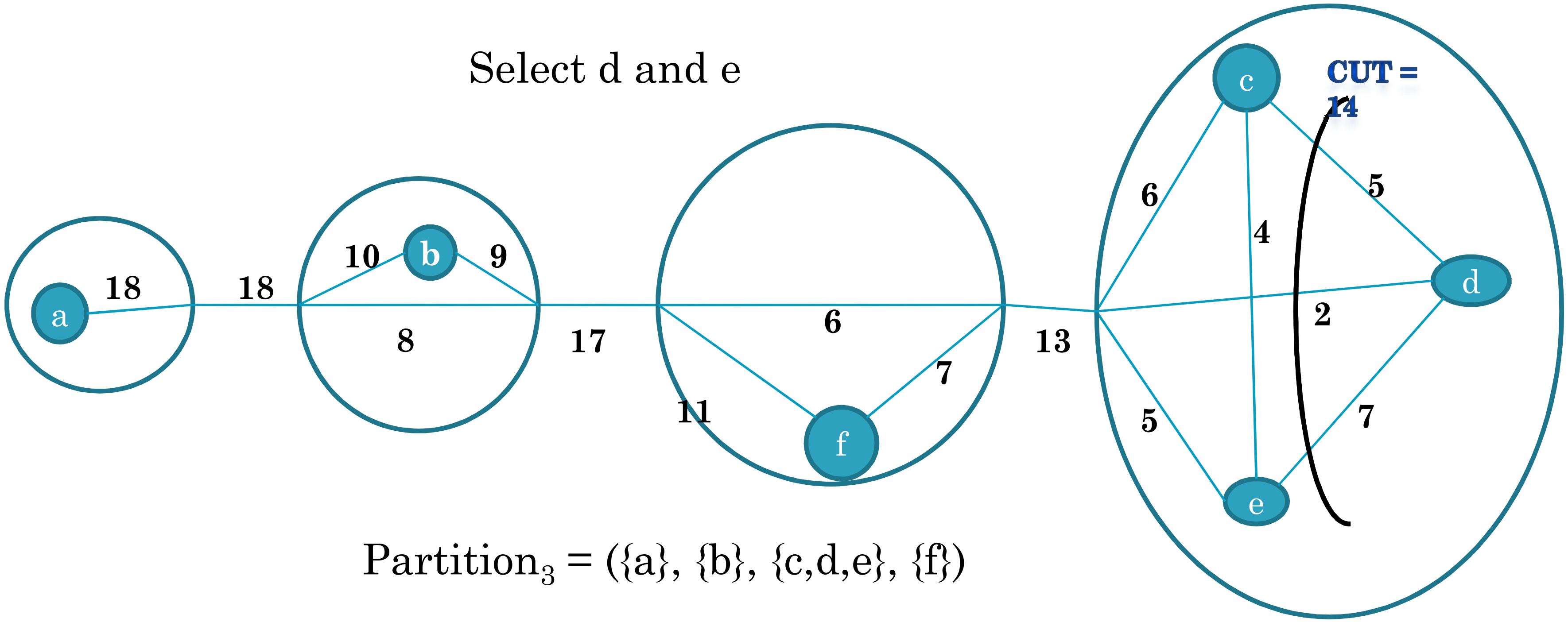
Iteration 3



Partition$_3$ = ({a}, {b}, {c,d,e}, {f})

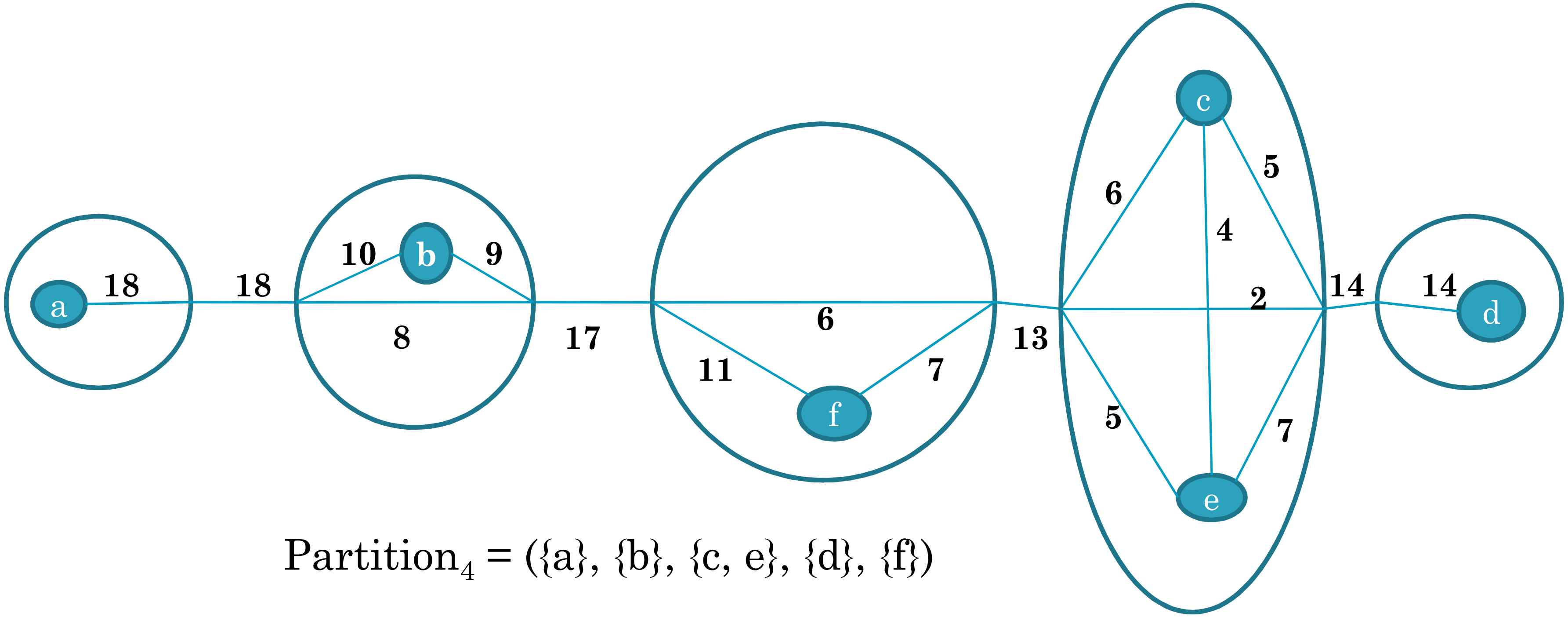
ITERATION 4

Select d and e



Partition$_3$ = ({a}, {b}, {c,d,e}, {f})

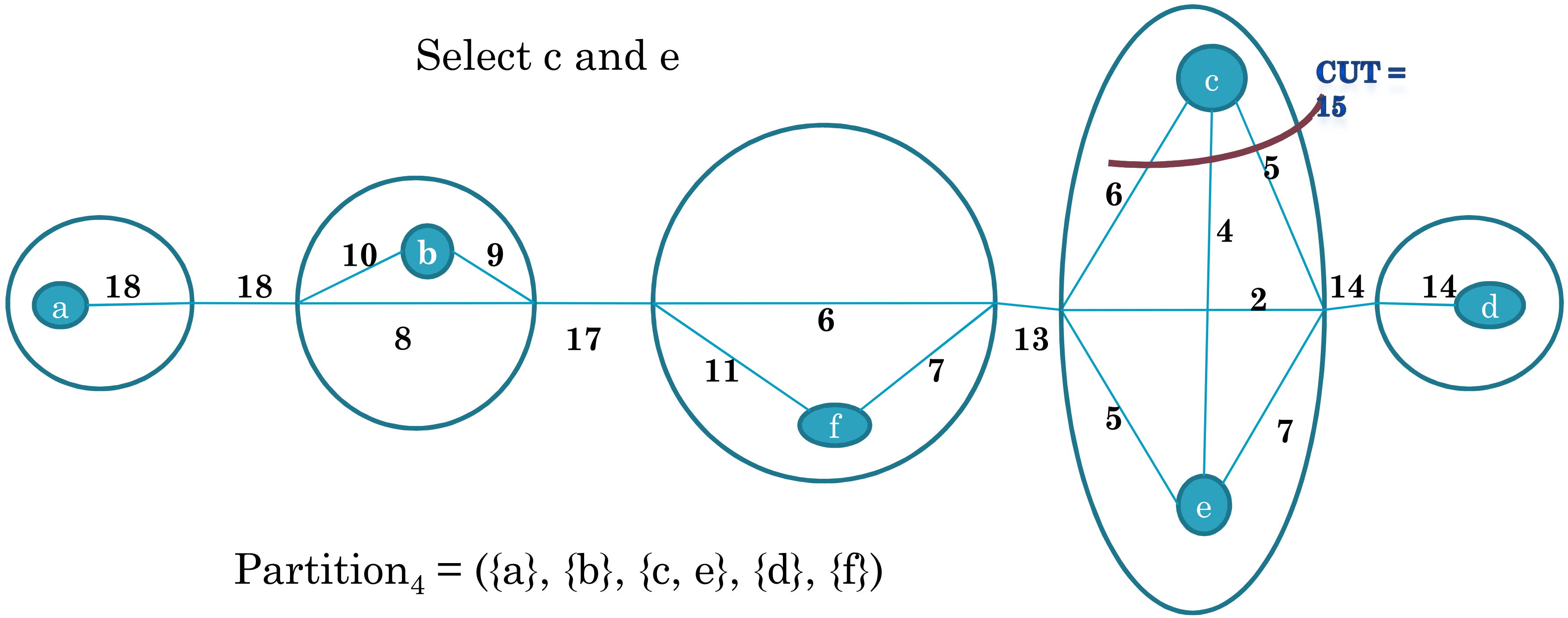
ITERATION 4



Partition$_4$ = ({a}, {b}, {c, e}, {d}, {f})

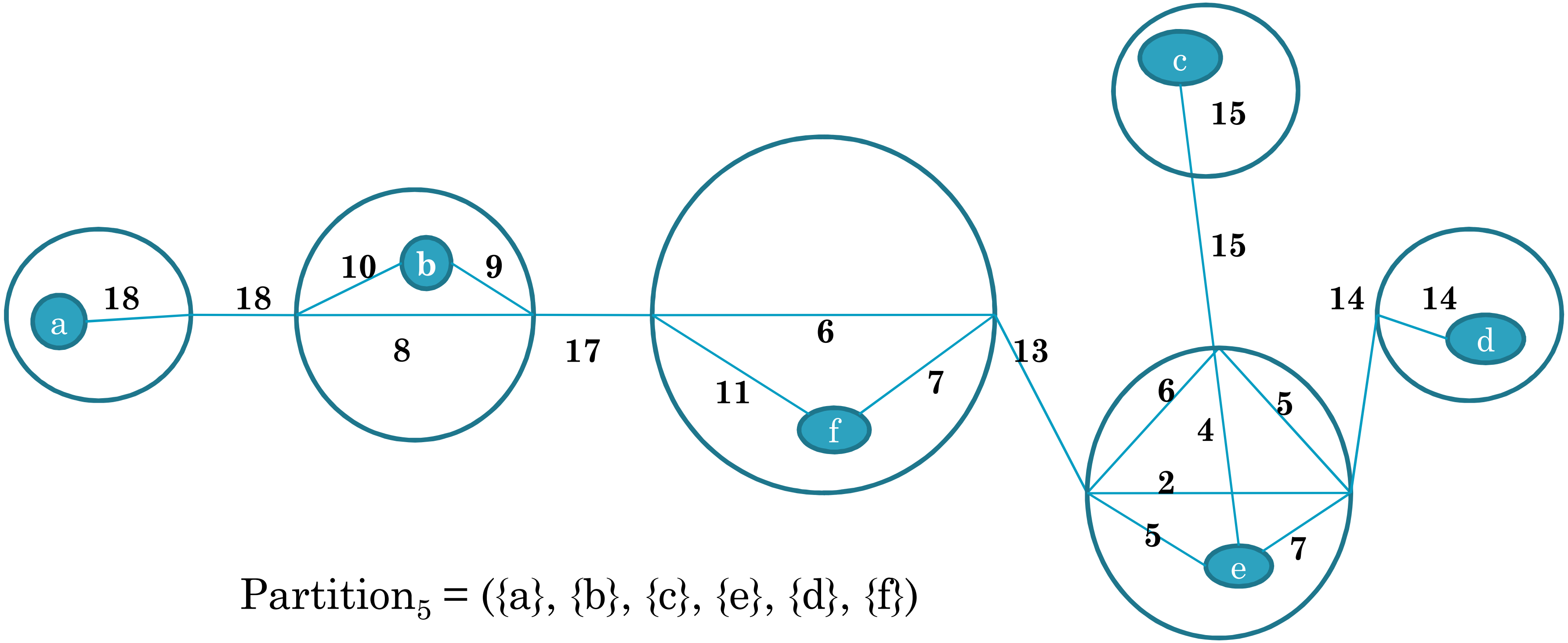
ITERATION 5

Select c and e



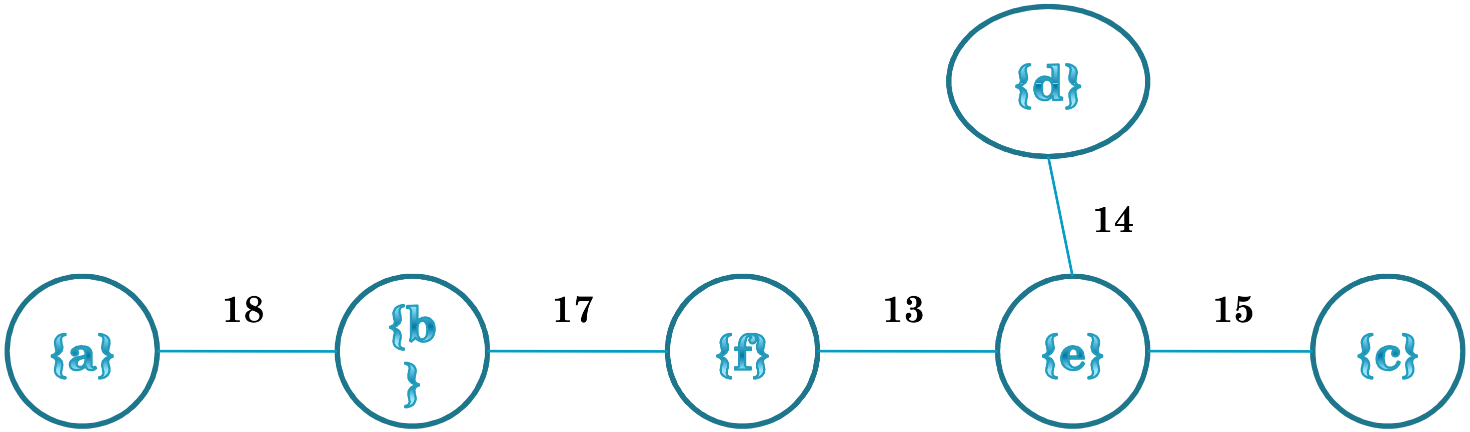
Partition$_4$ = ({a}, {b}, {c, e}, {d}, {f})

ITERATION 5



$\text{Partition}_5 = (\{a\}, \{b\}, \{c\}, \{e\}, \{d\}, \{f\})$

FINAL GH TREE



Final Gomory-Hu Tree

PROOF OF CORRECTNESS

Basic Lemmas (1)

- Let $f(u,v)$ denote the weight of a minimum **u-v cut** in G.
- For u, v, w $\in$ V, the following inequality holds:

$$f(u,v) \geq min\ \{ f(u,w), f(w,v) \}$$

- Generalization:

  For u, v, w_1, w_2, .., w_r $\in$ V:

$$f(u,v) \geq min\ \{ f(u, w_1), f(w_1, w_2), \ldots, f(w_r, v) \}$$

PROOF #2

Basic Lemmas (2)

- Let (A, A’) be a minimum s-t cut, $s \in A$.
- Choose any two vertices $x,y \in A$.
- Obtain graph G’ by **collapsing** all vertices of A’ to a single vertex $v_{A'}$.
- The weight of an edge $(a, v_{A'})$ is defined to be the sum of the weights of (a,b), where $b \in A'$.
- A minimum x-y cut in G’ defines a minimum x-y cut in G !!
- Thus, condensing A’ to a single node does not affect the value of a minimum cut from x to y.

PROOF #3

PROOF

- **INVARIANT (PROOF #4):**
  - For any edge (S_i, S_j) in T, there are vertices a and b in S_i and S_j respectively such that

1. w' (S_i, S_j) = f(a,b)
2. The cut defined by edge (S_i, S_j) is a minimum a-b cut in G.

- The first property satisfies the first GH condition (equivalent flow tree).
- The second property satisfies the second GH condition (cut property).

Minimum K-Cut Problem

DEFINITION

- Let G= (V,E) an undirected weighted graph.
- A set of edges of E whose removal leaves k connected components is called a k-cut.

- The **MINIMUM k-CUT** problem asks for a minimum weight k-cut.

Algorithm

- **Step 1**

  Compute a GH tree for graph G.

- **Step 2**

  Output the union of the lightest k-1 cuts of the n-1 cuts associated with edges of T in G. Let C be this union.

ANALYSIS

- **Lemma** :

  Let S be the union of cuts in G associated with l edges of T. Then, the removal of S from G leaves a graph with at least l+1 components.

- Hence, the union of k-1 cuts picked from T will form a k-cut in G.
- We will prove that the previous algorithm obtains an approximation ratio of 2 – 2/k.

PROOF #5

Other Interesting Properties of GH Trees (1)

- If the GH tree for a graph G contains all n-1 distinct weights, then G can have only one minimum weight cut!
- We can improve the performance of the GH algorithm by picking vertices for each set which after the min-cut computation will partition the set in equally sized subsets.

Other Interesting Properties of GH Trees (2)

- Let G be a network having an edge e = [i, j] with parametric capacity $c(e) = \lambda$.
- Let GH^{α} be a cut-tree obtained when $c(e) = \alpha$.
- Let $P_{i,j}^{\alpha}$ be the path in GH^{α} between i and j.
- For $\lambda > \alpha$ it is sufficient to compute $|P_{i,j}^{\alpha}|-1$ minimum cuts in G^{λ} in order to obtain a cut-tree GH^{λ} .

For theorists

Implementation

Implementation in C++ (1)

- To solve the undirected max-flow problem, we used linear programming (GNU LP API).
- Faster algorithms could be used!
- Based on the above max-flow algorithm, we implemented an algorithm for the min s-t cut problem (max-flow and reachability in residue graph).

Implementation in C++ (2)

- We implemented the GH algorithm using the above functions, as well as some basic STL classes (e.g. set and map).
- A quite fast method for computing the collapsed graph was used.
- The final GH tree is represented as a collection of weighted edges

IMPLEMENTATION IN C++ (3)

- The current implementation is only console-based.
- A graphical version is on the road. Damn it, you linux library dependencies!!

Thank You For Your Attention !