Assignment 2.
Due November 21

Submit latex-generated pdf on gradescope.

Exercise 1. *(A min-max equality* **[10 marks]***) Consider a graph $G = (V, E)$ with connected components of size at least 2. An edge covering is a subset $F \subseteq E$ such that each vertex is incident to some edge in F (note that E is an edge covering under the initial assumption). Let $\rho(G)$ be the minimum size of an edge covering in G and $\tau(G)$ be the maximum size of a matching in G.*

(1) *Prove that $|V| = \rho(G) + \tau(G)$.*
(2) *Conclude that $\rho(G) = max_{U \subseteq V} \frac{|U| + odd(G[U])}{2}$.*

Proof. **PART 1**

Let k be the number of vertices covered by a max matching size (hence $\frac{k}{2}$ is the max matching size). Note that k is well-defined since each edge in a matching covers exactly 2 vertices.

Claim: a min edge covering must use $\frac{k}{2} + |V| - k$ edges.

I first show a min edge covering cannot use fewer edges than in claim. Suppose a min edge covering, C, has fewer than $\frac{k}{2} + |V| - k$ edges. Note that by minimality of covering, there are no edges $e = \{u, v\}$ in C where u and v are both covered by other edges in C, since otherwise we can remove e to obtain a smaller edge covering. So for any edge e in C, e shares at most one end point with another edge in C. Denote the type of edges that share no end points with another edge as C_1 (call this type of edges "type 1"), and one end point as C_2 ("type 2"). The total number of nodes covered by C is $2 \times C_1 + C_2$ which must equal to $|V|$. Note that $\frac{k}{2} \geq |C_1|$, since $\frac{k}{2}$ is the size of the max matching. So

$$2 \times \frac{k}{2} + |C_2| \leq 2 \times |C_1| + |C_2| = |V|$$

$$\implies |C_2| \leq |V| - k.$$

Thus $|C| = |C_1| + |C_2| \leq \frac{k}{2} + |V| - k$ as desired.

Now we construct an edge covering with $\frac{k}{2} + |V| - k$ edges. First take a max matching M. Any edge in the max matching will be type 1 by definition of matching. Then for any uncovered node (hence unsaturated in matching), add any incident edge to the vertex. Any such edge is type 2. Otherwise, we can add such an edge to M to make a larger matching. Since these edges are type 2, their addition covers exactly 1 extra node each. So we need $|V| - k$ of them to cover all of $|V|$. So this covering has $\frac{k}{2} + |V| - k$ edges and we have shown the claim.

Finally,

$$\rho(G) + \tau(G) = \frac{k}{2} + \frac{k}{2} + |V| - k = |V|$$

as required.

PART 2

$$\rho(G) + \tau(G) = \rho(G) + \max_{S \subseteq V} \frac{|V| + |S| - o(G - S)}{2} = |V|$$
$$\iff \rho(G) = \max_{S \subseteq V} \frac{|V| - |S| + o(G - S)}{2} = \max_{U \subseteq V} \frac{|U| + o(G[U])}{2}$$

as required. The last equality is because $|V| - |S|$ is the size fo the complement of $|S|$, and $G - S$ is the graph induced by the complement of S.

□

Exercise 2. [10 marks] *Let δ be the minimum degree of a node in a graph G. Show that there is a pair of nodes s, t between which we can find δ edge-disjoint paths.*

Proof. If the $\delta = 0$, then there's certainly no paths between the 0 degree node and any other node and we are done. If G is a singleton there's also 0 paths.

Let $\delta > 0$ and consider a Gomory-Hu tree, T for G by setting capacities to 1. We know from class a Gomory-Hu tree always exists. Set any node to be the root. Then T must have a non-root leaf node (degree 1), v. Let u denote the node connected to v in T.

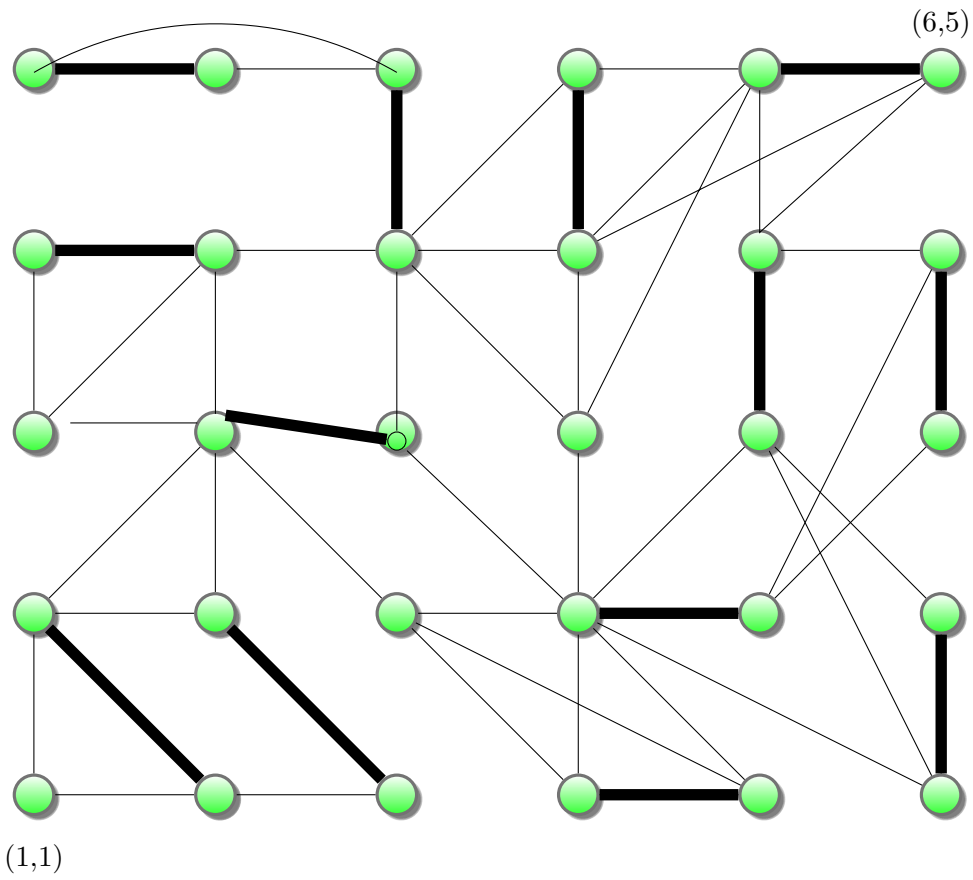
Note that by property of cut trees, $\lambda_{uv} \geq \delta$ since v has degree at least δ. This means that any cut between u and v in G has capacity at least δ.

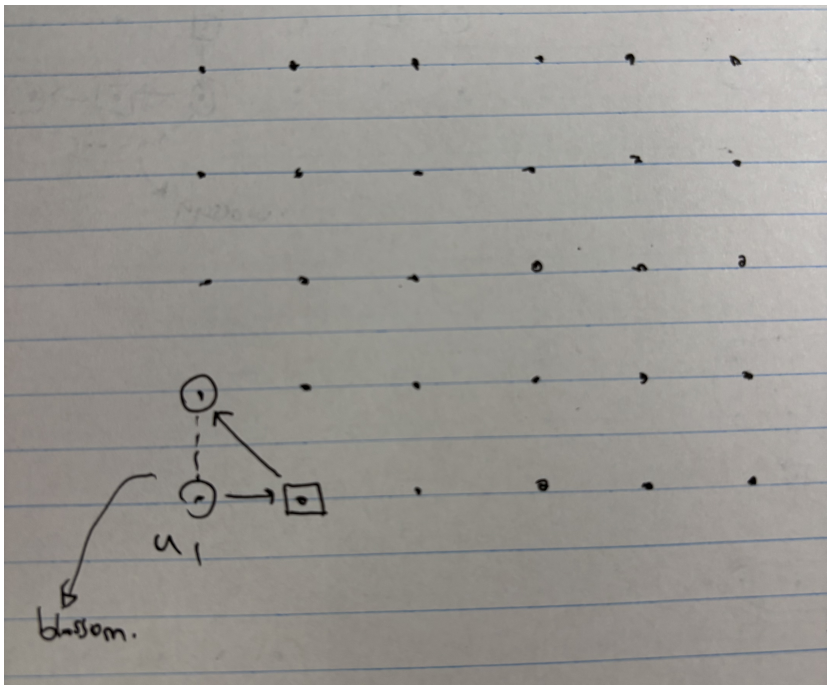
This is equivalent to saying there are at least δ edge-disjoint paths from u to v. To see this, consider the max flow between u and v in G. Since the min cut has size at least δ, we know the max flow is at least δ. Note any two units of routed flow cannot share an edge on the path, since each edge was set to have capacity 1. So indeed any two units of routed flow are routed on edge-disjoint paths. So we must have at least δ disjoint paths between u and v and we are done.

□

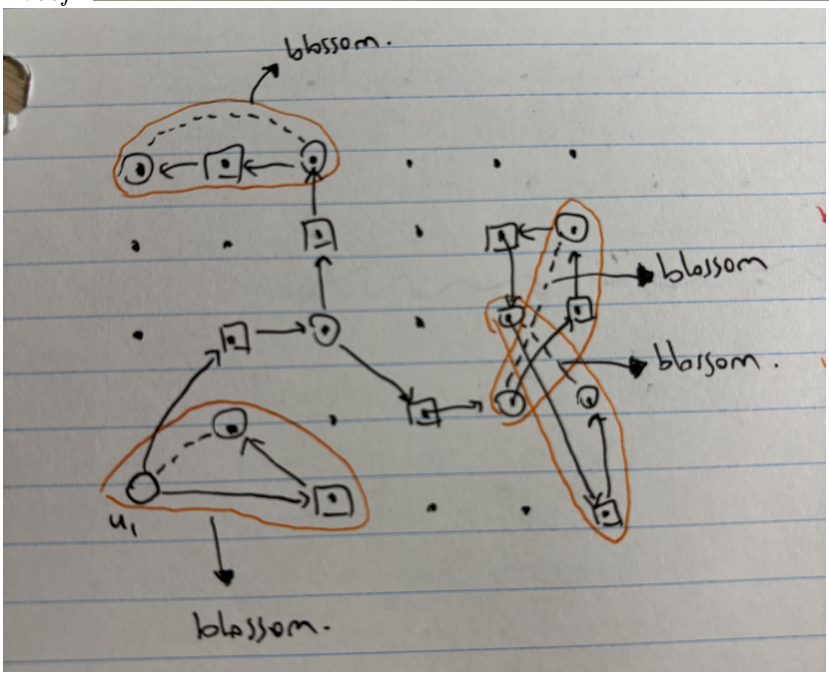
Exercise 3. *(Maximum Cardinality Matching* **[10 marks]***) Run the Blossom Algorithm to completion on the following graph with an initial matching indicated in bold. You do not need to draw a new graph for each time you extend your forest. But you should indicate which edges are in your forest and the direction used when an edge was added. You should also indicate the nested family of blossoms you shrink.*

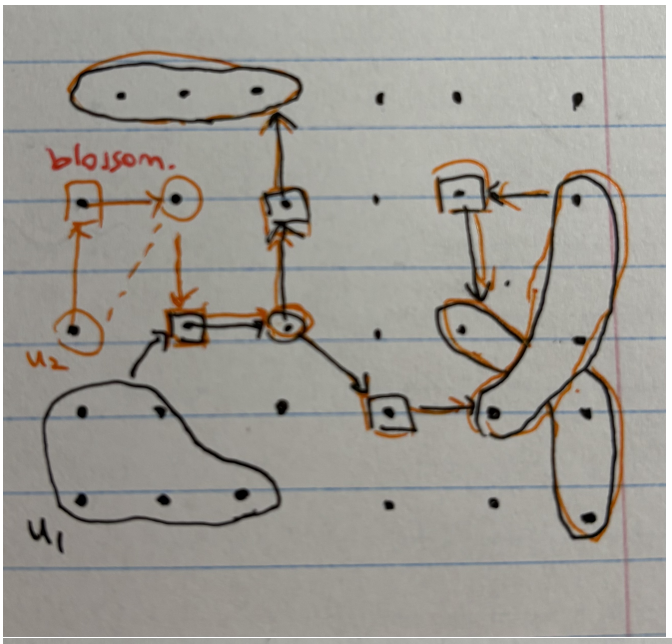
Explain clearly your certificate which verifies that you found a maximum size matching.

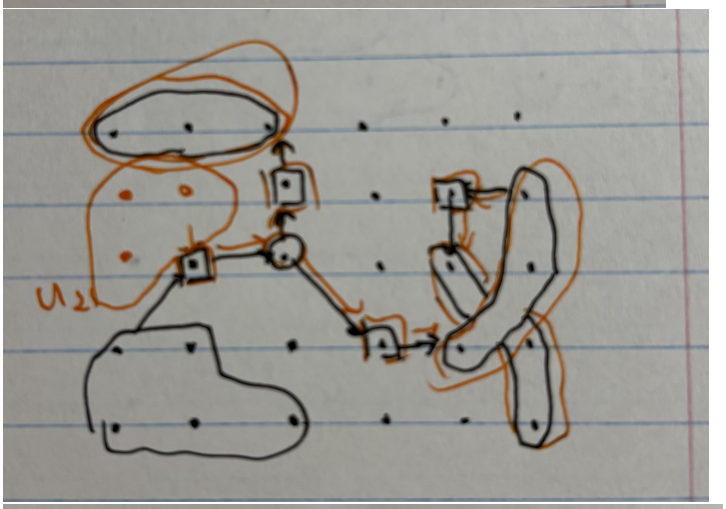


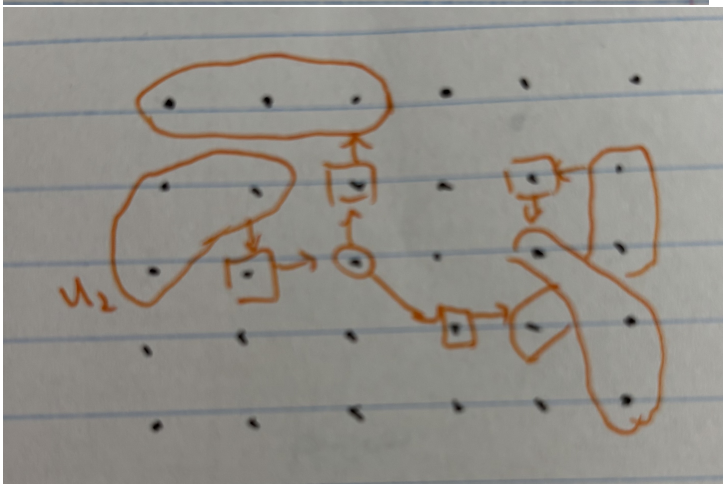


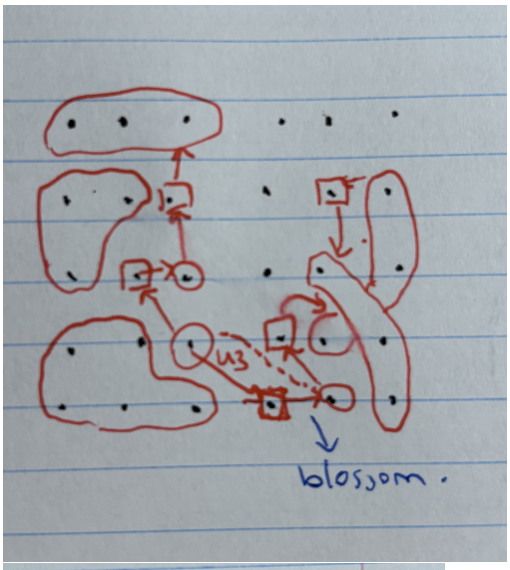
Proof.

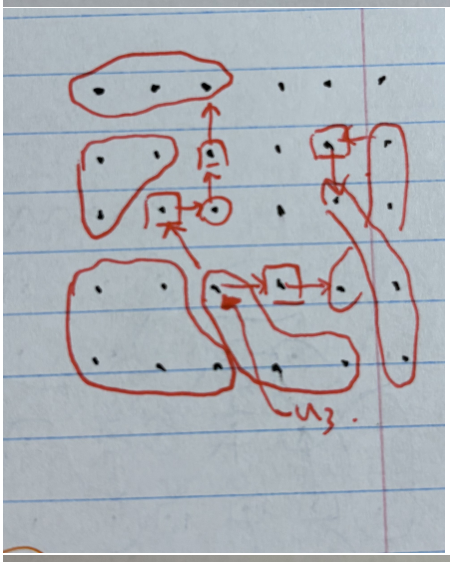


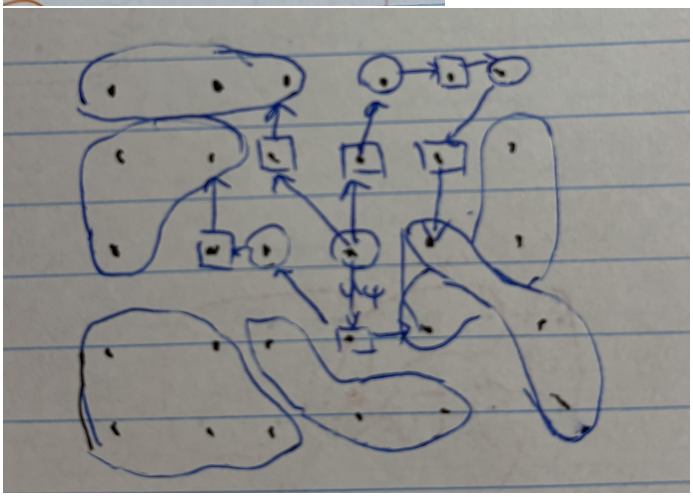

blossom.
u_2
u_1


u_2


u_2







So we have a forest of 4 trees and no augmenting paths. Condensing our blossoms give us

the fantasy property (no two outer nodes in the same tree have an edge between them). We also know from class that there cannot be edges between outer nodes of different trees, otherwise we would have an augmenting path. We also know since we have no augmenting path that all outer nodes are outer nodes in any tree, and inner and inner in any tree.

All of this tells us there are no edges between *any* two outer nodes. So by letting our inner nodes be S ($|S| = 6$), we know that

$$o(G_{shrunk} - S) = 10$$

We also know from class this implies

$$o(G - S) = 10$$

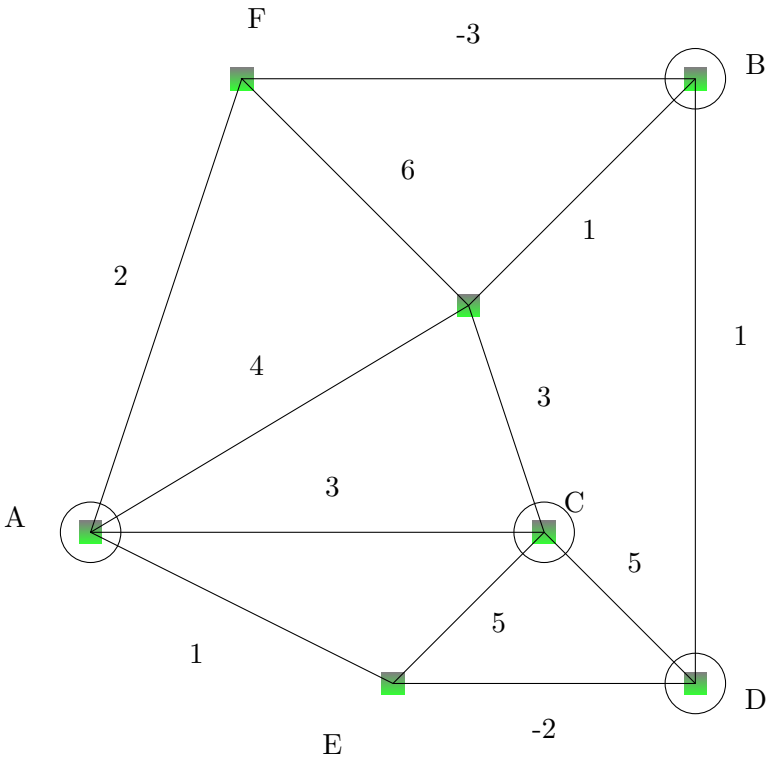
since each outer node contains an odd number of nodes each. Thus, we know by Tutte-Berge that any max matching M is such that

$$|M| \leq \frac{|V| + |S| - o(G - S)}{2} = \frac{30 + 6 - 10}{2} = 13.$$

The current matching we have has size 13 (count the edges). So we know it's a maximum matching. □

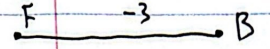
Exercise 4. *(T-joins.* **[10 marks]***)*

Find a minimum cost T-join for the following graph with costs shown and T-nodes circled, i.e., $T = \{A, B, C, D\}$. Show all your work.



Proof. We follow the algorithm on page 22 of the Tjoin.pdf lecture notes

(N)



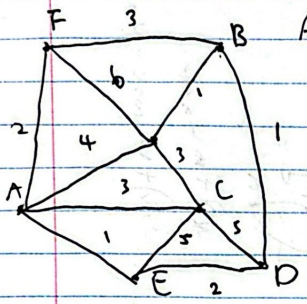
$T = \{B, A, C, D\}$.

$T' = \{B, F, E, D\}$

$N = \{\overline{FB}, \overline{ED}\}$.

$T \triangle T' = \{A, C, F, E\}$.

$G_{[2]} \rightarrow$



AC shortest: A → C (length 3)

AF shortest: A → F (length 2)

AE shortest: A → E (length 1)

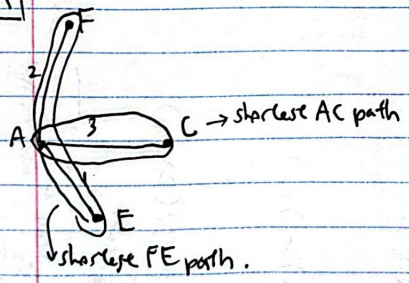
CF shortest: C → A → F (length 5)

CE shortest: C → A → E (length 4)

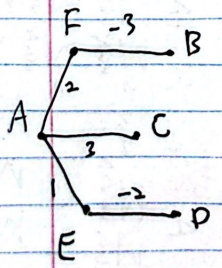
EF shortest: E → A → F (length 3).

• min cost perfect matching has two edges. Any perfect matching has cost 6.
Choose $\overline{AC}$ and $\overline{FE}$.

min cost.
$T \triangle T'$-join $= J$.



JOIN



□

Exercise 5. [10 marks] *Consider a set V of nodes that wish to communicate. Each pair i, j wants to share d_{ij} units of demand. We want a tree on V to carry this communication so as to minimize the cost*

$$\sum_{i,j} d_{ij} \cdot dist_T(i,j)$$

where $dist_T(i,j)$ is the length of the ij path in T. Show that the Gomory-Hu tree for the weights d_{ij} solves this problem.

Proof. We first develop some intuition. Notice that for any tree, the objective function is actually the sum of the edge weights of the cut tree formed by the tree on the demand graph G. To see this, we use an "edge formulation" instead of the current "distance formulation" of the objective value.

For a fixed tree T, instead of pairs of nodes, we sum over all edges in T. For each unique path between pairs of nodes i, j, denoted as P_{ij}^T, if $e \in P_{ij}^T$ we incur d_{ij}. These two are equivalent because we see that the demand / cost d_{ij} is incurred on exactly $|P_{ij}^T| = dist_T(i,j)$ edges. This means

$$\sum_{ij} d_{ij} dist_T(i,j) = \sum_{e \in T} \sum_{i,j: e \in P_{ij}^T} d_{ij} = \sum_{e \in T} \delta_T(e)$$

Now, consider a cut tree T_{cut} formed by the edges of T. Recall G is the graph of demand edges. So for each edge, the weight of each edge in T_{cut} is the sum of demands crossing the edge, which is exactly our edge formulation expression. Namely $w_{T_{cut}}(e) = \sum_{i,j: e \in P_{ij}^T} d_{ij}$. So the objective value of any tree is indeed the sum of edge weights of its corresponding cut-tree. Hence, our problem is equivalent to finding a cut tree with the smallest total edge weight. I now show that a Gomory-Hu tree satisfies this criteria.

Let T be any tree and T_{cut} the cut tree formed by edges of T. Also let T_G a Gomory Hu tree. Let $w_T(e)$ denote the weight of edge e on a cut tree T.

Main Claim: There exists a bijective map $f : E(T_G) \to E(T_{cut})$ such that $w_{T_{cut}}(f(e)) \geq w_{T_G}(e)$

If such a map exists, then gom-hu trees indeed have the minimum sum of edge weight of any cut-tree.

Claim 1: $\forall e \in E(T_G)$**,** $\exists e' \in E(T_{cut})$ *s.t.* $w_{T_{cut}}(e') \geq w_{T_G}(e)$

Proof. Let $e = \{s, t\} \in E(T_G)$. Then $w_{T_G}(e)$ is the value of a min cut between s, t in G. Consider the path P between s and t in T_{cut}. Removing any edge in this path forms an s, t-cut. So any of edge weight of P will be greater or equal to $w_{T_G}(e)$ (the value of the min s, t-cut). □

We define the $cover(e)$ function to be the set of all these edges.

Definition 1 (Cover)**.** *For any $e = \{s, t\} \in T_G$, define $cover(e)$ to be the set of edges on the unique path between s, t in T_{cut}. Some properties are that*

(1) *e connects the endpoints of the T_{cut} path in T_G*
(2) *$cover(e)$ is always non-empty*
(3) *No two edges $e_1, e_2 \in T_G, e_1 \neq e_2$ have $cover(e_1) = cover(e_2)$. Namely, cover is injective.*

Intuitively, *cover* maps each Gomory-Hu edge to T_{cut} edges that are guaranteed to have greater weight than $w_{T_G}(e)$.

Claim 2. For any $C \subseteq E(T_{cut})$, the number of edges in $e \in T_G$ such that $cover(e) \subseteq C$ cannot be greater than $|C|$.

Proof. Let C be a counter example with size $k \in \mathbf{N}$ but $l \geq k+1$ edges e such that $cover(e) \subseteq C$. Note such $e \in T_G$ connects some two vertices in C. So in T_G, we have l edges that share the $k+1$ nodes. This must produce a cycle in T_G, which is a contradiction. □

For a function h and subset edges $C \in E(T_{cut})$, define the set $E_C^h = \{e \in T_G : h(e) \subseteq C\}$.

Proposition 1. *Let $h : E(T_G) \to 2^{E(T_{cut})}$ be a function with the following properties:*

(1) $\forall e \in E(T_G), h(e) \neq \emptyset$
(2) *For edges $C \subseteq E(T_{cut})$, $|E_C^h| \leq |C|$.*

then for every fixed $C \subseteq E(T_{cut})$, there is a injective function $g : E_C^h \to C$ such that $\forall e \in E_C^h$, $g(e) \in h(e)$.

Proof. Proceed with induction on size of C. For the base case, let $C = \{e'\}$. We know $|E_C^h| \leq 1$ by property 2 of h. If $|E_C^h| = 0$ we don't need a map and we are done. Otherwise $E_C^h = \{e\}$ for some $e \in T_G$. Map $g(e) = e'$. By definition of E_C^h, e is an edge such that $h(e) \subseteq C = \{e'\}$. Further, $h(e) = \{e'\}$ since it is non-empty by property 1. So indeed $g(e) = e' \in h(e)$.

Next, let the C have size $k+1$. Let l be a leaf edge in a component C (note $C \subseteq T_{cut}$ so each component is a subtree). We know $|E_C^h| \leq |C| = k+1$ by property 2. We potentially select an edge e^* by the following procedure:

(1) If there is an edge $e \in E_C^h$ such that $h(e) = \{l\}$, then select e.
(2) Otherwise, select an arbitrary edge $e \in E_C^h$ such that $l \in h(e)$.
(3) Otherwise $\forall e \in E_C^h$ we have $l \notin h(e)$. Select nothing.

In the third case, we have $E_C^h = E_{C-l}^h$ since $\forall e \in E_C^h$, we have $l \notin h(e)$, which implies $h(e) \subseteq P - l$. Thus, we can use induction on the shorter path $P - l$ (with length k) with the same function h. This gives us a function g from E_{C-l}^h to $C - l$ (namely $E_C^h \to C - l$ since $E_{C-l}^h = E_C^h$) such that $g(e) \in h(e)$. So g satisfies our proposition for $k+1$.

In the first two cases, we have chosen some edge e^* such that $l \in h(e^*)$. Define new function h' with:

(1) $\forall D \subseteq E(T_{cut}), \forall e \in E_D^h$ such that $e \neq e^*$ and $l \in h(e)$, define $h'(e) = h(e) - l$.
(2) otherwise, $h'(e) = h(e)$

I want to show that h' is a function that satisfies property (1) and (2) described in Proposition 1.

Property (1) By construction h' only ever deleted l from any $h(e)$. So suppose for some $x \in T_G$, we have $h'(x) = \emptyset$. Then since $h(x) \neq \emptyset$ by property (1), we must have $h(x) = \{l\}$. Note then that $x \neq e^*$ since $h'(e^*) = h(e^*)$ by construction (whereas $h(x) \neq h'(x)$). And since x wasn't chosen as e^* and $h(x) = \{l\}$, e^* must be another edge such that $h(e^*) = \{l\}$. So

$$\{x, e^*\} \subseteq E_{\{l\}}^h \implies |E_{\{l\}}^h| \geq 2$$

which is a contradiction to property (2) since $\{l\}$ has length 1 so we should have $|E^h_{\{l\}}| \leq 1$. This shows property 1.

Claim 4: If $l \in D$, then $E^{h'}_D = E^h_D$. If $l \notin D$, then $E^h_D \subseteq E^{h'}_D$.

Proof. For $e \in E^{h'}_D$, we have $h'(e) \subseteq D$. We either have $h(e) = h'(e) \subseteq D$ or $h(e) = h'(e) + l \subseteq D$. For E^h_D, we have $h(e) \subseteq D$. We either have $h'(e) = h(e) \subseteq D$ or $h'(e) = h(e) - l \subseteq D$. □

(INCOMPLETE)

Claim 5: For any $D_1, D_2 \subseteq E(T_{cut})$, such that $D_1 \cap D_2 = \emptyset$ and $|\bigcup_{e \in D_1 \cup D_2} h(e)| < |D_1 \cap D_2|$, at most one of the $D \in \{D_1, D_2\}$ has $|E^h_D| = |D|$ (equality instead of $<$).

Proof. Suppose $|E^h_{D_1}| = |D_1|$ and $|E^h_{D_2}| = |D_2|$. Then $E^h_{D_1 \cap D_2} = \{e \in T_G : h(e) \subseteq \bigcup_{e \in D_1 \cup D_2} h(e)\} = D_1 + D_2$. But □

Property (2) Let $D \subseteq E(T_{cut})$ By **Claim 4** if $l \in D$, then $|E^{h'}_D| = |E^h_D| \leq |D|$ and we are done. So suppose $l \notin D$. We note that $E^{h'}_{D+l} = E^h_{D+l}$ by **Claim 4**. So

$$|E^{h'}_{D+l}| = |E^h_{D+l}| \leq |D| + 1.$$

by property (2) of function h. Suppose $E^h_{D+l} = |D| + 1$ and $\forall e \in E^{h'}_{D+l}, e \subseteq D$. Then $E^{h'}_{D+l} = E^{h'}_D$ so $|E^{h'}_{D+l}| = |E^{h'}_D| = |D| + 1$. Recall h' only ever removed l from any $h(e)$, since we only have $h'(e) = h(e)$ or $h'(e) = h(e) - l$. We also know $|E^h_D| \leq |D|$. So there is a $e \in T_G$ such that $e \in E^{h'}_D, e \notin E^h_D$, meaning $h'(e) = h(e) - l$.

This edge $e \notin E^h_D$ but certainly $e \in E^h_{D+l}$ since $h(e) = h'(e) + l \subseteq D + l$. By **Claim 4**, this means this edge is in $e \in E^{h'}_{D+l}$.

Now, consider $E^{h'}_{D+C}$. We know $l \in h'(e^*) = h(e^*)$ by choice of e^* so $h(e^*) \subseteq C$ but $h(e^*) \nsubseteq D$.

Claim 3: $E^{h'}_{C-l} = E^h_C - e^*$.

Proof. Let $e \in E^{h'}_{C-l}$, then $h'(e) \subseteq C - l$. Note that $l \in h'(e^*)$ so $e \neq e^*$. Now $h(e)$ is either $h'(e) + l \subseteq C$ or $h'(e) \subseteq C - l \subset C$. So $e \in E^h_C$, but $e \neq e^*$. Let $e \in E^h_C - e^*$, so $h(e) \subseteq C$. Then if $h(e)$ contains l, the construction of $h'(e)$ removes it. Otherwise, $h(e)$ does not contain l and $h'(e) = h(e)$. In either case, $h'(e) \subseteq C - l$ so $e \in E^{h'}_{C-l}$. □

So h' is a function that satisfies properties in proposition 1. So I invoke induction with h' and $C - l$ to get a function $g : E^{h'}_{C-l} \to C - l$. Namely, $g : E^h_C - e^* \to C - l$ (by **Claim 3**) such that $g(e) \in h'(e) \subseteq h(e)$. Finally, add $g(e^*) = l$, then g is a mapping from $g : E^h_C \to C$. Also note $g(e^*) = l \subseteq h(e^*)$ by choice of e^*.

□

Proof Main Claim. Note that *cover* is a function that satisfies Proposition 1, since each edge in T_G maps to a non-empty path in T_{cut}. Proposition 1 property 2 is true by **Claim 2**. Choose C to be $E(T_{cut})$. Then we know there exists an injective function $g : E^{cover}_{E(T_{cut})} =$

$E(T_G) \to E(T_{cut})$ such that $\forall e \in E^{cover}_{E(T_{cut})} = T_G, g(e) \in cover(e)$. Since $|E(T_G)| = |E(T_{cut})|$, this mapping is bijective. So we have shown the main claim.

□

Exercise 6. [15 marks] *Let $G = (V, E)$ be a graph with edge weights $w_{uv} > 0$ which can be viewed as how "similar" u, v are. For subsets $A, B \subseteq V$ define $w(A, B)$ to be the sum of w_e for all edges with one endpoint in A and the other in B.*

Create an auxiliary graph G' obtained by adding a new node s adjacent to every node of V; these new edges have weight $\alpha \geq 0$. Let T be a Gomory-Hu tree for the resulting graph with capacities $w(e)$. Deleting s from T yields a partition (corresponding to subtrees of $T - s$) of V: $V = C_1 \cup C_2 \ldots \cup C_{n_\alpha}$. We show that these clusters *have some nice properties in terms of keeping similar nodes together.*

(1) *For each cluster C_i and $v \in C_i$ (other than its root node), v has more weight to nodes inside C_i, than outside:*

$$\sum_{u \in V \setminus C_i} w_{uv} \leq \sum_{u \in C_i} w_{uv}$$

Proof. Let C_i be a cluster and $v \in C_i$ such that $v \neq v_i$ (the root of cluster i) as per the question. Note that the v_i, s edge defines a minimal v_i, s-cut $\delta(C_i, V - C_i)$ with weight $w(C_i, V - C_i + \{s\})$. Now, consider also the cut formed by moving v out of C_i, which defines the v_i, s-cut $\delta(C_i - v, V - C_i + \{s\} + v)$ with weight $w(C_i - v, V + \{s\} - C_i) + w(C_i - v, v)$.

Since $v \neq v_i$, this new cut is still an v_i, s cut. So

$$\begin{aligned}
&\Rightarrow w(C_i, V + \{s\} - C_i) \leq w(C_i - v, V + \{s\} - C_i) + w(C_i - v, v) \\
\Rightarrow w(v, V + \{s\} - C_i) + w(C_i - v, V + \{s\} - C_i) &\leq w(C_i - v, V + \{s\} - C_i) + w(C_i - v, v) \\
&\Rightarrow w(v, V + \{s\} - C_i) \leq w(C_i - v, v) \\
&\Rightarrow \alpha + \sum_{u \in V - C_i} w_{uv} \leq \sum_{u \in C_i} w_{uv} \\
&\Rightarrow \sum_{u \in V - C_i} w_{uv} \leq \sum_{u \in C_i} w_{uv}
\end{aligned}$$

as desired. □

(2) *Each cluster is internally dense. Define the* expansion *of a set S as follows. If $|S| = 1$ it is ∞. Otherwise it is:*

$$\min_{\emptyset \neq Q \subsetneq S} \frac{w(Q, S \setminus Q)}{\min\{|Q|, |S \setminus Q|\}}.$$

Show that each cluster has expansion at least α.

Proof. Let C_i be a cluster, and $Q \subsetneq C_i, v_i$ the root of C_i. Assume WLOG that $|Q| \leq |C_i - Q|$, since otherwise we can relabel. Suppose first $v_i \in C_i - Q$.

We know that $\delta(C_i, V + \{s\} - C_i)$ is a minimum v_i, s-cut. Now, consider the cut $\delta(C_i - Q, V + \{s\} - C_i + Q)$, with weight $w(C_i - Q, \{s\}) + w(C_i - Q, V - C_i) +$

$w(C_i - Q, Q)$. Note this a v_i, s-cut since $v_i \in C_i - Q$. So

$$\begin{aligned}
w(C_i, V + \{s\} - C_i) &\leq w(C_i - Q, V + \{s\} - C_i) + w(C_i - Q, Q) \\
\Rightarrow w(Q, V + \{s\} - C_i) &\leq w(Q, C_i - Q) \\
\Rightarrow w(Q, \{s\}) + w(Q, V - C_i) &\leq w(Q, C_i - Q) \\
\Rightarrow w(Q, s) &\leq w(Q, C_i - Q) \rightarrow \text{(since edge weights are positive)} \\
\Rightarrow \alpha|Q| &\leq w(Q, C_i - Q) \\
\Rightarrow \alpha &\leq \frac{w(Q, C_i - Q)}{|Q|} = \frac{w(Q, C_i - Q)}{\min\{|Q|, |C_i - Q|\}} \rightarrow \text{(since } |Q| \leq |C_i - Q|)
\end{aligned}$$

Now let $v_i \in Q$. Then $\delta(Q, V - Q + \{s\})$ is a v_i, s-cut. So similarly, using the min v_i, s-cut $\delta(C_i, V + \{s\} - C_i)$, we have:

$$\begin{aligned}
w(C_i, V + \{s\} - C_i) &\leq w(Q, V - Q + \{s\}) \\
\Rightarrow w(C_i, V - C_i) + |C_i|\alpha &\leq w(Q, V - Q) + |Q|\alpha \\
\Rightarrow w(C_i, V - C_i) + |C_i|\alpha &\leq w(Q, V - C_i) + w(Q, C_i - Q) + |Q|\alpha \\
\Rightarrow w(C_i - Q, V - C_i) + |C_i|\alpha &\leq w(Q, C_i - Q) + |Q|\alpha \\
\Rightarrow w(C_i - Q, V - C_i) + (|C_i - Q|)\alpha &\leq w(Q, C_i - Q) \rightarrow (Q \subseteq C_i \text{ so } |C_i| - |Q| = |C_i - Q|) \\
\Rightarrow (|C_i - Q|)\alpha &\leq w(Q, C_i - Q) \rightarrow \text{(since edge weights are positive)} \\
\Rightarrow \alpha &\leq \frac{w(Q, C_i - Q)}{|C_i - Q|} \leq \frac{w(Q, C_i - Q)}{\min\{|Q|, |C_i - Q|\}}
\end{aligned}$$

since $|Q| \leq |C_i - Q|$, as desired. □

(3) *Moreover, show the boundary of each cluster is sparse:* $\frac{w(C_i, V \setminus C_i)}{|V \setminus C_i|} \leq \alpha$.

Proof. Let C_i a cluster. Assume $V \neq C_i$ (from Piazza). The edge $\{v_i, s\}$ gives us the fundamental cut $\delta(C_i, V - C_i + \{s\})$. Consider the cut $\delta(\{s\}, V)$. Then this is a $v_i, s-$cut. So

$$\begin{aligned}
w(C_i, V - C_i + \{s\}) &\leq w(\{s\}, V - \{s\}) \\
w(C_i, \{s\}) + w(C_i, V) = \alpha|C_i| + w(C_i, V - C_i) &\leq \alpha|V| \\
w(C_i, V - C_i) &\leq \alpha|V - C| \\
\frac{w(C_i, V - C_i)}{|V - C|} &\leq \alpha
\end{aligned}$$

as desired. □

(4) *When $\alpha = 0$ argue that we have a single cluster. Determine a value α_0 (dependent on $w(e)$'s) for which we obtain n singleton clusters, i.e., each $|C_i| = 1$. Now show that as we increase α the clusterings obtained are slowly refined. In other words, show that if $\alpha' > \alpha$, then each cluster for α' is a subset of some cluster for α.*

Proof. **Single Cluster**

We recall that on a Gomory Hu tree T_G, λ_{ab} is equivalent to the minimum-capacity edge on the path from a to b. Thus if $\alpha = 0$, an edge connected to s cannot appear on the path of any two non-s nodes in a Gomory-Hu tree. This is

because for $a \neq s, b \neq s$, we have $\lambda_{ab} \geq w_{ab} > 0 = \alpha$. This means s must be a leaf node in a Gom-Hu tree. Removing s thus gives one component.

a0 Set $a_0 = \sum_{u,v \in V} w_{uv} + 1$. To see why the gomory-hu tree must be a star, suppose for contradiction, there is a component C_i with at least two nodes. Let x be a node connected to the root of C_i, v_i (in the gom-hu tree). Then the cut $\delta(C_i, V + \{s\} - C_i)$ is a v_i, s-cut with weight

$$w(C_i, V + \{s\} - C_i) \geq w(\{v_i, x\}, \{s\}) = 2\alpha_0$$

But the cut $\delta(\{v_i\}, V + \{s\} - \{v_i\})$ with weight

$$\alpha_0 + w(\{v_i\}, V - \{v_i\}) \leq \alpha_0 + \sum_{u \in V - v_i} w_{v_i u} < 2\alpha_0$$

is a v_i, s-cut with a smaller weight. Which is a contradiction by definition of gomory-hu trees.

Refined Clusters Let $\alpha' > \alpha \geq 0$. Let the gom-hu tree formed with $\alpha = \alpha$ be $T_{G',\alpha}$ and the one formed with $\alpha = \alpha'$ be $T_{G',\alpha'}$. Let C_i' with root v_i' be a component in $T_{G',\alpha'}$. Let C_i be the component in $T_{G',\alpha}$ with root v_i such that $v_i' \in C_i$. I want to show $C_i \subseteq C_i'$.

Both components form v_i', s-cuts. $T_{G',\alpha'}$ tells us:

$$w(C_i', V - C_i' + \{s\}) \leq w(C_i, V - C_i + \{s\})$$
$$w(C_i', V - C_i') + |C_i'|\alpha' \leq w(C_i, V - C_i) + |C_i|\alpha'$$

Now, let $c_1, ..., c_k \in C_i - C_i' - v_i$. And iteratively apply this procedure:

(a) Let C initially be $C_i' - v_i$
(b) Since C_i' is an minimum cut, there is a minimum c_1, s-cut containing C_i'.
(c) Set C to be this new cut. Repeat.

As the last step, C is a minimum c_k, s-cut. So there must be a minimum v_i, s-cut containing C. So we have generated a minimum v_i, s-cut (in G' with $\alpha = \alpha'$) that contains $C_i' \cup C_i$ such that:

$$w(C, V - C) + |C|\alpha' \leq w(C_i, V - C_i) + |C_i|\alpha'$$

Since C_i induces a min v_i, s-cut in G' with $\alpha = \alpha$, we have

$$w(C_i, V - C_i) + |C_i|\alpha \leq w(C, V - C) + |C|\alpha$$

Suppose $w(C_i, V - C_i) > w(C, V - C)$. Then $|C|\alpha' < |C_i|\alpha'$. This is a contradiction because $C_i \subseteq C$ so $|C| \geq |C_i|$. Similarly, if $w(C_i, V - C_i) < w(C, V - C)$. Then $|C|\alpha < |C_i|\alpha$, a contradiction.

So we must have $w(C_i, V - C_i) = w(C, V - C)$. Then our equations become $|C|\alpha' \leq |C_i|\alpha'$ and $|C_i|\alpha \leq |C|\alpha$. So

$$|C_i|\alpha \leq |C|\alpha \leq |C|\alpha' \leq |C_i|\alpha'$$

If $|C_i| < |C|$ then $|C|\alpha' \leq |C_i|\alpha'$ is a contradiction. If $|C_i| > |C|$ then $|C_i|\alpha \leq |C|\alpha$ is a contradiction. So $|C_i| = |C|$.

So indeed $C = C_i \cup C_i' = C_i$. So $C_i \subseteq C_i'$ as required.

THERE IS A MISTAKE HERE BUT I DON'T SEE IT? MY CONCLUSION HERE IS THAT $C_i = C_i'$, WHICH IS INCORRECT.

□