CSCI 570 Fall 2025

Homework 8

Due: March 28

Note: In Questions 3, 4, and 5, we assume that the Ford-Fulkerson algorithm is provided. You do not need to rewrite the Ford-Fulkerson algorithm.

Question 1

Consider a flow network with source 0, sink 6, and the following edges:

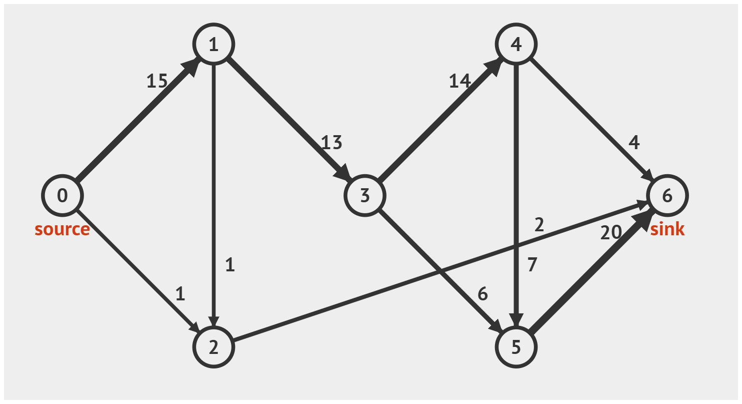


Figure 1: (*The capacity of the edge from 2 to 6 is two.*)

- Draw the **first, second, and final** residual graphs G_f using the Edmonds-Karp implementation of Ford-Fulkerson algorithm corresponding to the max flow problem.
- Determine the max-flow value.
- Provide **all** of the min-cuts.

Solution 1

Source: Main link or link

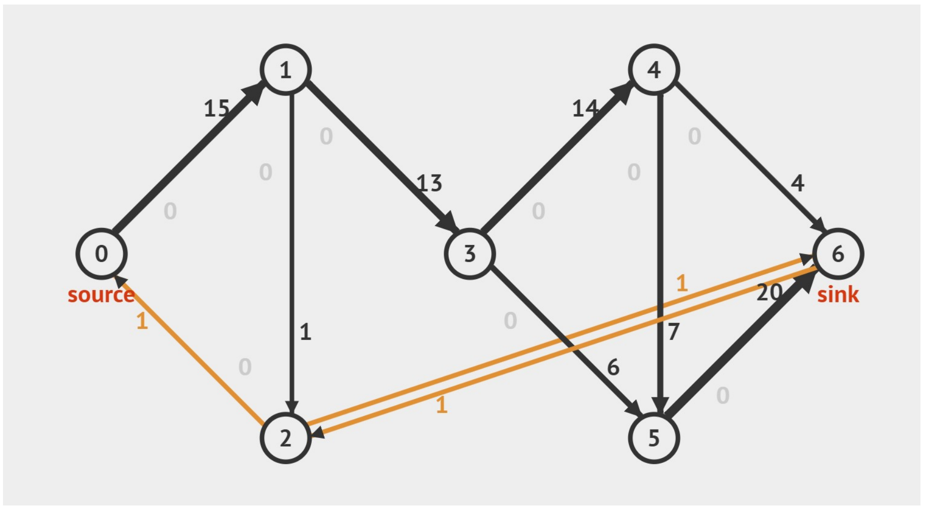


Figure 2: First step's residual graph

Maximum flow:

Maximum flow = 15.

There are 2 min-cuts:

$\{s, 1\}, \{2, 3, 4, 5, t\}$ and $\{s, 1, 2\}, \{3, 4, 5, t\}$

Rubric 1

(21 pts)

- **15** Each of three residual graph gets 5 points and each edge is 0.5 point.
- **2 pts:** Correct final maximum flow answer.
- **4 pts:** 2 points for each of the correct min-cuts

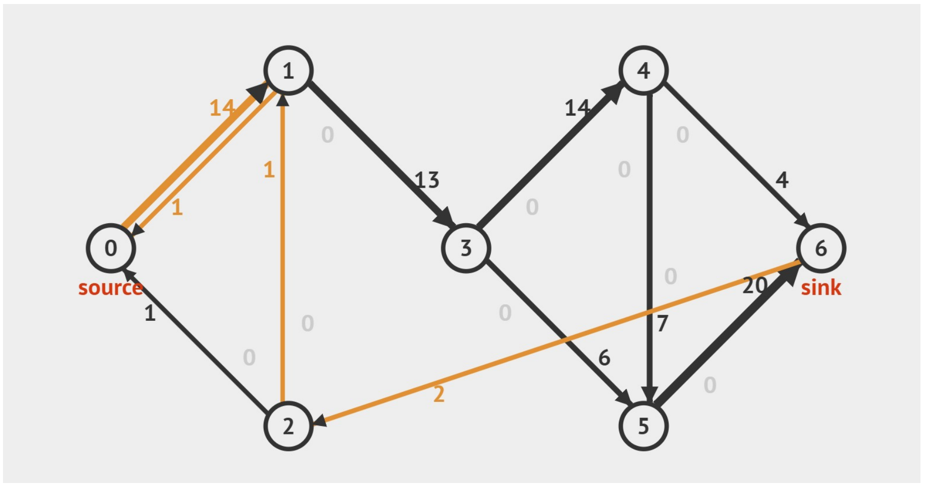


Figure 3: Second step's residual graph

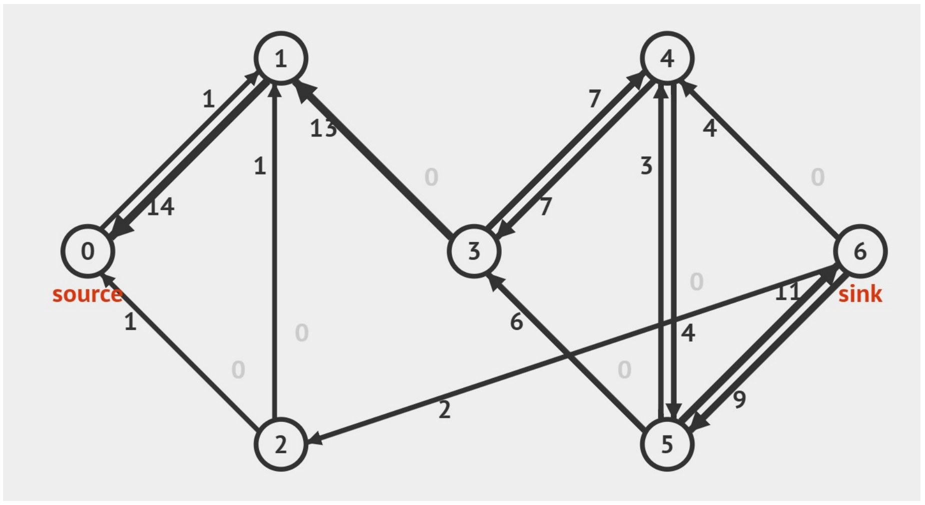
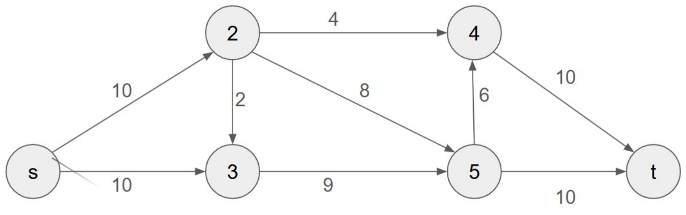


Figure 4: Final step's residual graph

Question 2

Consider a flow network with source s, sink t, and the following edges:



- Draw the **first, second, and final** residual graph G_f using the scaled version of Ford-Fulkerson algorithm corresponding to the max flow problem.
  - Give the value of Δ and G_f after first iteration.
  - Give the value of Δ and G_f after second iteration.
  - Give the final G_f.
- Find the max-flow value.
- Find **all** of the min-cuts.

Solution 2

Source: link

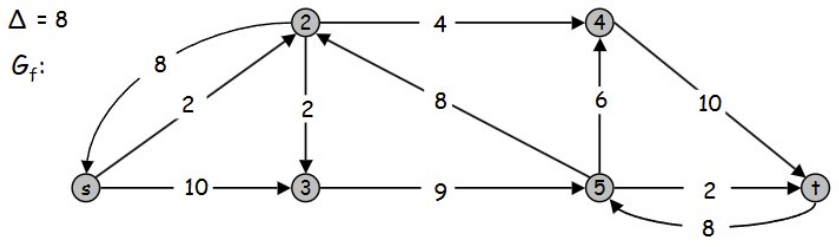


Figure 5: First step's residual graph

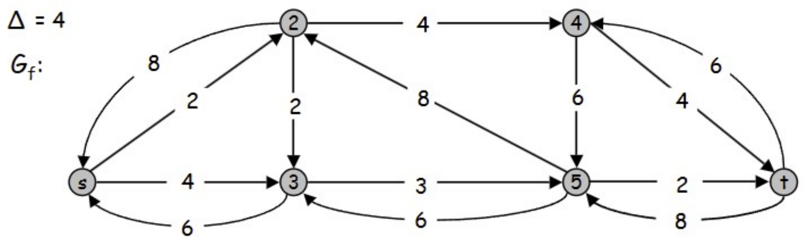


Figure 6: Second step's residual graph

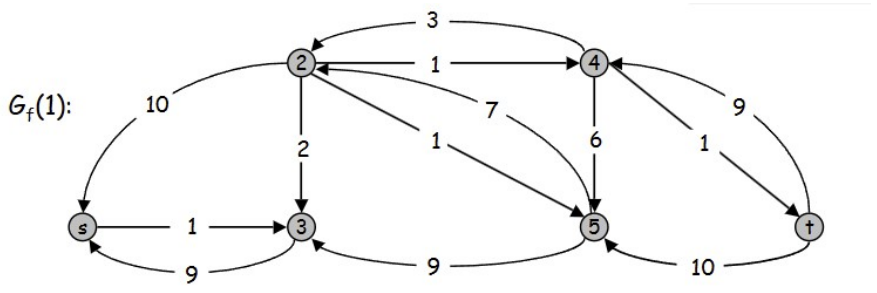


Figure 7: Final step's residual graph

Maximum flow:

Maximum flow = 19.

There is only 1 min cut:

$\{s, 3\}, \{2, 4, 5, t\}$

Rubric 1

(20 pts)

- **13.5 pts:** Each of the three residual graph gets 4.5 points and each edge is 0.5 point.
  Please note that there are several valid execution paths for this algorithm, which means the residual graph is not unique. Full credit will be given for any correct answer.
- **4.5 pts:** Correct final maximum flow answer.
- **2 pts:** Correct min-cut

Question 3

Part (a): Assignment Feasibility

You are given:

- n entities $e_1, e_2, \dots, e_n$.
- k groups $g_1, g_2, \dots, g_k$.
- For each entity e_j, a subset $p_j \subseteq \{g_1, g_2, \dots, g_k\}$ with $|p_j| \geq m$, where m is a positive integer.
- For each group g_i, a capacity q_i (maximum number of entities it can contain).

You want to determine if it is possible to assign each entity e_j to **at least** m groups from its subset p_j, such that no group g_i contains more than q_i entities.

Task: Design an algorithm to check if such an assignment exists and **prove** its correctness.

(Hint: The proof will have two directions. 1)Forward Direction: If feasible, then [statement]. 2) Backward Direction: If [statement] then feasible.)

Part (b): Selection with Constraints

Assume a feasible solution to part (a) exists, and each entity e_j is assigned to **exactly** m groups from p_j. Select exactly one entity as a "representative" for each group g_i from the entities assigned to it. No entity can be a representative for more than r groups, where $r < m$. Determine if such a selection of representatives is possible.

Task: Design an algorithm to check if this selection exists and prove its correctness.

(Hint: Use the solution from part (a) as a starting point.)

(Hint: The prrof will have two directions. 1)Forward Direction: If feasible, then [statement]. 2) Backward Direction: If [statement] then feasible.)

Solution 3

Part (a): Assignment Feasibility

Algorithm

We can solve the problem by constructing a network flow graph and then running Ford–Fulkerson algorithm to get the max flow. The setup of the graph is described below.

1. Construct a bipartite graph with n entity vertices $e_1, e_2, \ldots, e_n$ and k group vertices $g_1, g_2, \ldots, g_k$. (2 Points)
2. Add an edge from e_j to g_i for each $g_i \in p_j$ with capacity 1. (2 Points)
3. Add a source vertex s and a sink vertex t. (2 Points)
4. Connect s to each e_j with capacity m. (2 Points)
5. Connect each g_i to t with capacity q_i. (2 Points)
6. Run Ford-Fulkerson from s to t. If the max flow is $n \cdot m$, the assignment is feasible; otherwise, it is not. (2 Points)

Claim

The assignment is feasible if and only if the maximum flow is $n \cdot m$.

Proof of Correctness

- **Forward Direction: If feasible, then max flow = $n \cdot m$:**
  - Suppose a feasible assignment exists: each e_j is assigned to at least m groups from p_j, and each g_i has at most q_i entities.
  - By flow conservation, the flow into each e_j from s is at least m. Since edges $s \to e_j$ have capacity m and the assignment is feasible, these edges are saturated (flow $= m$).
  - This flow of m from each e_j is distributed to groups $g_i \in p_j$ via edges of capacity 1, totaling $n \cdot m$ across all entities.
  - Each g_i sends flow to t within capacity q_i, and the total flow to t is $n \cdot m$.
  - Thus, the max flow is $n \cdot m$.
- **Backward Direction: If max flow = $n \cdot m$, then feasible:**
  - If max flow is $n \cdot m$, each $s \to e_j$ edge carries flow m (since total flow is $n \cdot m$ and there are n such edges).
  - This flow splits across edges $e_j \to g_i$ (capacity 1), assigning e_j to exactly m groups from p_j, satisfying the minimum requirement.
  - Flow from each g_i to t is at most q_i, respecting group capacities.
  - Thus, following the flow defines a feasible assignment.

Part (b): Selection with Constraints

Algorithm

We can solve the problem by starting from the solution from part (a). We will modify the constructed network flow graph slightly and then will run Ford–Fulkerson algorithm to get the max flow. If the max flow is equal to the number of classes k, then a selection exists otherwise no such selection exists. The setup of the graph is described below.

1. Use the same nodes as in part (a): s, t, e_j, g_i. (2 Points)
2. s to each g_i: capacity 1 (one representative per group). (2 Points)
3. g_i to e_j if e_j is assigned to g_i: capacity 1. (2 Points)
4. e_j to t: capacity r. (2 Points)
5. Run Ford-Fulkerson from s to t. If the max flow is k, a selection exists; otherwise, it does not. (2 Points)

Claim

The selection is feasible if and only if the maximum flow is k.

Proof of Correctness

- **Forward Direction: If feasible, then max flow = k:**
  - Suppose a feasible selection exists: each g_i has one representative from its assigned entities, and no e_j represents more than r groups.
  - Flow from s to each g_i is 1 (capacity 1), and by conservation, each g_i sends flow 1 to one e_j assigned to it.
  - Each e_j sends flow to t within capacity r, and the total flow is k (one per group).
  - Thus, the max flow is k.
- **Backward Direction: If max flow = k, then feasible:**
  - If max flow is k, each $s \to g_i$ edge carries flow 1 (total flow $= k$ for k groups).
  - Each g_i sends flow 1 to one e_j (capacity 1), selecting e_j as its representative.
  - Flow from each e_j to t is at most r, so no entity represents more than r groups.
  - Thus, following the flow defines a feasible selection.

Rubric 3

Rubric Part A (18 pts)

- **12 pts:** Correct construction of the network flow graph (a detailed rubric is provided in the solution).
- **3 pts:** Proof of correctness, Forward Direction
- **3 pts:** Proof of correctness, Backward Direction

Rubric Part B (16 pts)

- **10 pts:** Correct construction of the network flow graph (a detailed rubric is provided in the solution).
- **3 pts:** Proof of correctness, Forward Direction
- **3 pts:** Proof of correctness, Backward Direction

Question 4

Given a flow network $G = (V, E)$ with source s, sink t, and unit-capacity edges ($u_e = 1$ for all $e \in E$), and an integer k, find a set $F \subseteq E$, $|F| = k$, to minimize the max s-to-t flow in $G' = (V, E - F)$.

Task: Give an algorithm to solve this problem. Prove the correctness.

Solution 4

Algorithm

1. Compute the max s-to-t flow g in G, and identify a min-cut (S, T) with edge set $C = \{(u, v) \in E \mid u \in S, v \in T\}$, $|C| = g$.
2. **If** $g \leq k$**:**
   - Set $F \leftarrow C$. If $k > g$, add $k - g$ arbitrary edges from $E - C$ to F.
   - We have in $G' = (V, E - F)$, the maximum flow is 0.
3. **If** $g > k$**:**
   - Set F as any k edges from C.
   - In $G' = (V, E - F)$, the maximum flow is equal to $g - k$.
4. Return F.

Correctness

- $g \leq k$**:** Removing C ($|C| = g$) disconnects s from t, yielding max flow 0, the minimum possible.
- $g > k$**:** Removing k edges from C leaves $g - k$ edges in the cut, so max flow $\geq g - k$. Since each edge has capacity 1, max flow = number of edge-disjoint paths. Removing k edges reduces this by at most k, and targeting a min-cut achieves exactly k, making $g - k$ minimal (any lower flow implies a smaller original min-cut, a contradiction).

Rubric 4

(6)

- **4 points**: Any correct algorithm is acceptable.
- **2 points**: Proof

Question 5

A dating service receives data **D** from p men and q women. These data determine which pairs of men and women are mutually compatible and which are not. Since the dating service's commission is proportional to the number of dates it arranges, it would like to determine the maximum number of compatible couples that can be formed. (Note that each man or woman is assigned at most one date.)

Task: Design an algorithm to determine the maximum number of mutually compatible dates that can occur simultaneously. Proof is not required.

Solution 5

We can model this problem using a graph:

- Create a bipartite graph with p nodes representing men and q nodes representing women. (2 Points)
- Draw an edge with capacity one from a man to a woman if they are mutually compatible. (2 Points)
- Add a source node s and connect it to each man with an edge of capacity one. (1 Points)
- Add a sink node t and connect each woman to t with an edge of capacity one. (1 Points)

Now, determine the maximum flow using the Ford-Fulkerson algorithm, which is equal to the maximum number of mutually compatible dates possible. (2 Points)

Rubric (8 points)

A detailed rubric is provided in the solution.

Ungraded

A vertex cover of an undirected graph $G = (V, E)$ is a subset $A \subseteq V$ such that every edge $e \in E$ has at least one of its incident vertices in A. Given a bipartite graph G' with vertex partition $V = L \cup R$ ($L \cap R = \emptyset$, edges only between L and R) and a positive integer k, design an algorithm to decide if G' has a vertex cover of size at most k.

Solution

Algorithm

Construct a flow network G'' from G':

(a) Add source s and sink t.

(b) Add edges $s \to u$ (capacity 1) for each $u \in L$.

(c) Add edges $v \to t$ (capacity 1) for each $v \in R$.

(d) For each $(u, v) \in E$ (from $u \in L$ to $v \in R$), add $u \to v$ (capacity ∞).

(e) Compute max s-to-t flow in G'' using Ford-Fulkerson; let its value be m.

(f) If $k < m$, output NO; else output YES.

Running Time

Ford-Fulkerson runs in $O(E \cdot |f|)$, where $|f| \leq |V|$. Total: $O(EV)$, polynomial time.

Correctness

Claim: G' has a vertex cover of size $\leq k$ if and only if the max flow in G'' is $\leq k$.

- **Vertex Cover Size = Min-Cut Capacity:**
  - Let (S, T) be a min-cut in G'', $s \in S$, $t \in T$, capacity m.
  - Define $A := (L - S) \cup (R \cap S)$.
  - Edges $(u, v) \in E$ (infinite capacity) don't cross the cut (if $u \in S$, $v \in S$, since A is a vertex cover).
  - Cut edges are: (s, u) for $u \in L \cap S$, (v, t) for $v \in R \cap T$.
  - Capacity $= |L \cap S| + |R \cap T| = |L - (L - S)| + |R \cap S| = |(L - S) \cup (R \cap S)| = |A|$.
  - Thus, min-cut capacity = size of vertex cover A.
- **Converse: Given Vertex Cover A, Construct Cut:**

  - Define $S := \{s\} \cup (L \cap A) \cup (R - A)$, $T := V \cup \{t\} - S$.
  - For $(u, v) \in E$, if $u \notin A$, then $u \notin S$; if $v \in A$, then $v \in S$, so no infinite-capacity edge crosses.
  - Cut edges: (s, u) for $u \in L \cap A$, (v, t) for $v \in R - A$.
  - Capacity $= |L \cap A| + |R - A| = |A|$.

- **Conclusion:** Min vertex cover size = max flow m. If $k < m$, no vertex cover of size $\leq k$ exists; otherwise, YES.