CSCI 570 Homework 9
Name: Abhishek Soundalgekar
USC ID: 2089011000

Question 1

Given a weighted, directed graph G = (V, E), determine whether there exist k paths such that each path starts at a unique vertex in the set A = {a1, a2, . . . , ak} ⊂ V, and ends at a unique vertex in the set B = {b1, b2, . . . , bk} ⊂ V, with no two paths sharing any vertices or edges. (Note that B ∩ A = ∅.)

Solution 1

Flow Network Construction

1. **Vertex Splitting:**
   For every vertex v ∈ V , create two vertices: v in and v out. Add a directed edge from v in to v out with capacity 1. This transformation ensures that each original vertex can only be used once in any path, since the capacity restricts the flow through v to a single unit.
2. **Edge Transformation:**
   For every original directed edge (u,v) ∈ E, add a corresponding edge from u in to v in with capacity 1. This models the connectivity of the original graph while respecting the vertex capacity limits established by the splitting.
3. **Super-Source and Super-Sink:**
   Super-Source s: Add a new vertex s. For each vertex ai ∈ A, add an edge from s to ai,in with capacity 1. This ensures that the flow starts uniquely at each ai.
   Super-Sink t: Add a new vertex t. For each vertex bj ∈ B, add an edge from bj,out to t with capacity 1. This ensures that the flow ends uniquely at each bj.

There exist kk vertex- and edge-disjoint paths from AA to BB in GG if and only if the maximum flow in the constructed network is kk.

Proof of Correctness

Forward Direction

- **Assumption:** There exist k vertex- and edge-disjoint paths in G, each starting at a unique vertex in A and ending at a unique vertex in B.
- **Flow Construction:**
  Send 1 unit of flow along each of these k disjoint paths in the transformed network.

    - **Vertex Constraint:** Since every vertex v is split into v in and v out with an edge of capacity 1, no two paths can pass through the same vertex because doing so would exceed the capacity on the edge (v in, v out).
    - **Edge Constraint:** Similarly, each original edge has been transformed into an edge with capacity 1, ensuring that no two paths share the same edge.
- **Conclusion:**
  Each path contributes 1 unit of flow, yielding a total flow of k. The capacities on edges incident to s and t (each being 1 for the corresponding ai and bj) guarantee that the total flow cannot exceed k. Therefore, the maximum flow is exactly k.

Backward Direction

- **Assumption:** The maximum flow in the constructed network is k.
- **Flow Decomposition:**
  By the flow decomposition theorem, the maximum flow can be decomposed into k unit flows, each constituting a flow from s to t.
- **Path Properties:**
    - Each unit flow must traverse the edge (v in, v out) for any vertex v it uses. Since these edges have capacity 1, no two unit flows can use the same vertex.
    - Similarly, the transformed edges corresponding to the original edges in G each have capacity 1, ensuring that no two flows share an edge.
- **Conclusion:**
  The k unit flows correspond directly to kk vertex- and edge-disjoint paths from vertices in AA (via s to ai, in) to vertices in B (via bj,out to t). This completes the proof that a maximum flow of k implies the existence of k vertex- and edge-disjoint paths.

Flow Network Construction:

Vertex Splitting: Each vertex v is split into v in and v out with an edge $(v\,in, v\,out)$ of capacity 1.

Edge Transformation: For every $(u,v) \in E$, add an edge (u out, v in) with capacity 1.

Super-Source and Super-Sink: Introduce a super-source s with edges (s, a i,in) of capacity 1 for each $ai \in A$, and a super-sink t with edges (b j,out, t) of capacity 1 for each $bj \in B$.

Proof of Correctness:

Forward Direction: If k disjoint paths exist, sending 1 unit of flow along each results in a flow of k without exceeding any edge capacities.

Backward Direction: If the maximum flow is k, then by flow decomposition, the k unit flows correspond to k vertex- and edge-disjoint paths in the original graph.

Question 2
We have n traders, each possessing Wi Swiss Francs (for i = 1, . . . , n). They wish to convert their Francs into m different currencies c1, . . . , cm.

However:
• Each currency cj has a maximum conversion limit Bj , meaning the bank can convert at most Bj Francs into currency cj (for j = 1, . . . , m).
• Each trader i can convert at most Si,j Francs into currency cj .

The question is: P Is it possible to convert all of the traders' Francs, i.e., n i=1 Wi, subject to the given constraints? Design an algorithm to decide feasibility by formulating and solving a maximum flow problem. Clearly describe the construction of the flow network: specify all vertices, edges, and capacities, and explain how each constraint in the original problem is modeled within this network.

Solution 2

Flow Network Construction

1. **Vertices:**

   Source s: Represents the starting point of the flow.
   Trader Vertices ti: For each trader i (where $1 \leq i \leq n1$), create a vertex ti. These vertices represent the traders who initially possess Wi Swiss Francs.
   Currency Vertices cj: For each currency jj (where $1 \leq j \leq m1$), create a vertex cj. These vertices represent the different currencies available for conversion.
   Sink τ: Represents the final destination of the flow (the bank's conversion process).

2. **Edges and Capacities:**

   Edges from the Source to Traders:
   For each trader i, add an edge from the source s to ti with capacity Wi.
   Rationale: This ensures that trader i cannot contribute more than Wi Francs into the network, which mirrors the initial amount they possess.

   Edges from Traders to Currencies:
   For each trader i and each currency j, add an edge from trader ti to currency cj with capacity Si,j.
   Rationale: This edge captures the constraint that trader i can convert at most Si,j Francs into currency cj. It limits the amount of flow that can pass from trader i into each currency channel.

   Edges from Currencies to the Sink:
   For each currency j, add an edge from currency vertex cj to the sink τ with capacity Bj.
   Rationale: This ensures that the bank can convert at most Bj Francs into currency cj. It enforces the overall limit on how much of each currency can be obtained.

3. **Algorithm:**

   Step 1: Construct the flow network as described above.

   Step 2: Use a maximum flow algorithm (such as Ford–Fulkerson, Edmonds-Karp) to compute the maximum flow from the source s to the sink τ.

   Step 3: Check if the maximum flow value | f | equals ∑ i=1 to n (Wi), which is the total amount of Swiss Francs held by all traders.
   If yes: The bank can convert all of the traders' Francs subject to the given constraints.
   If no: It is not possible to convert all of the Francs under the current constraints.

Vertices:
The construction includes all required vertices: s (source), ti for each trader, cj for each currency, and τ (sink). This correctly represents each entity in the problem.

Edges and Capacities:

(s, ti) with capacity Wi: Ensures each trader only contributes what they possess.

(ti, cj) with capacity Si,j: Enforces each trader's individual limit on converting into a specific currency.

(cj,τ) with capacity Bj: Respects the bank's conversion limit for each currency.

Algorithm Justification:

The maximum flow in this network represents the maximum total amount of Francs that can be converted given the constraints. By verifying whether the maximum flow equals ∑ i=1 to n (Wi), we are directly checking if every Franc can be accounted for through some valid path (conversion) that respects the individual trader limits and the bank's currency limits.

In-Short

Vertices:

s: source vertex.
ti: one vertex per trader i (total n traders).
cj: one vertex per currency j (total m currencies).
τ: sink vertex.

Edges and Capacities:

s → ti with capacity Wi.

ti → **cj** with capacity Si,j.
cj → **τ** with capacity Bj.

Algorithm:

Compute the maximum flow from s to τ.
The bank can convert all traders' Francs if and only if the maximum flow equals ∑ i=1 to n (Wi).

Question 3

We have m disjoint time intervals. There are n guards, and each guard is available for some subset of these intervals (i.e., the guard can only be assigned to intervals in which they are available). We need to assign guards so that:

- Each interval is covered by at least one and at most two guards.
- Each guard is deployed to at most Q intervals.
- Each guard is deployed to at least L intervals.

The goal is to decide whether there exists a valid assignment of guards to intervals satisfying all these conditions.

Task: Design an algorithm to decide feasibility by formulating and solving a maximum flow problem. Clearly describe the construction of the flow graph: specify all vertices, edges, capacities, and constraints, and explain how each constraint in the original problem is modeled within this network. Reduce the problem to a standard maximum flow problem without lower bounds by introducing new nodes where necessary.

Answer 3]

Flow Network Construction with Lower Bounds

We must assign guards to m intervals so that each interval gets covered by at least one and at most two guards, and each guard is assigned to between L and Q intervals. We model this as a circulation problem with lower bounds on some edges. Later, we transform it into a standard maximum flow problem.

1. Vertices

Source and Sink for the Circulation:
s: A source vertex.
t: A sink vertex.

Guard Nodes:
For each guard i (where $1 \leq i \leq n1$), create a vertex gi.

Interval Nodes:
For each interval j (where $1 \leq j \leq m1$), create a vertex Ij.

2. Edges and Capacity/Lower Bound Assignments

1. **Edges from the Source s to Each Guard gi:**
   - **Edge:** (s,gi)
   - **Lower Bound:** L
     Rationale: Ensures that each guard is deployed to at least L intervals.

- **Capacity:** Q
  Rationale: Guarantees that each guard is deployed to at most Q intervals.

2. **Edges from Guard gi to Interval Ij:**
   - **Edge:** (gi,Ij) is added **if and only if** guard ii is available during interval j.
   - **Lower Bound:** 0
     Rationale: There is no minimum requirement on a particular assignment.
   - **Capacity:** 1
     Rationale: A guard can cover a given interval at most once.

3. **Edges from Each Interval Ij to the Sink t:**
   - **Edge:** (Ij,t)
   - **Lower Bound:** 1
     Rationale: Guarantees that each interval is covered by **at least one** guard.
   - **Capacity:** 2
     Rationale: Ensures that each interval is covered by **at most two** guards.

4. **Edge from t to s:**
   - **Edge:** (t,s)(t, s)
   - **Lower Bound:** 0
   - **Capacity:** ∞
     Rationale: This edge "closes the loop" and allows us to convert the circulation into a flow circulation problem (ensuring flow conservation over the network).

3. Modeling the Constraints

- **Guard Assignment Constraints:**
  The edge (s, gi) with lower bound L and capacity Q ensures that guard i must be assigned to at least L and at most Q intervals.

- **Interval Coverage Constraints:**
  The edge (Ij, t) with a lower bound of 1 and a capacity of 2 guarantees that each interval j receives at least one guard and no more than two guards.

- **Availability Constraint:**
  Edges (gi, Ij) are only present if guard i is available during interval j; each such edge has a capacity of 1, meaning a guard can cover an interval at most once.

Reducing to a Standard Maximum Flow Problem

The above circulation formulation involves edges with lower bounds. We now describe the standard reduction process:

1. **Initial Flow Assignment:**
   For each edge e = (u, v) with lower bound l(e) and capacity c(e), we push l(e) units of flow through e. This defines a base flow f0(e)=l(e).

2. **Adjust Residual Capacities:**
   For every edge e=(u,v), define a new capacity:

$c'(e)=c(e)-l(e)$

This removes the lower bound requirement, as the new edge now has a capacity equal to the extra flow that can be sent above the pre-assigned l(e).

3. **Adjust Node Imbalances:**
   For every vertex v, compute its imbalance b(v) as:

   $b(v) = \sum \{e : \text{into } v\}\ l(e) - \sum \{e : \text{out of } v\}\ l(e)$

   Then update the demand:

   $d'(v)=d(v)-b(v)$

   In network:

   For the source s:
   s has n outgoing edges (one to each guard) each with lower bound L, so:

   $b(s) = 0 - nL = -nL \Rightarrow d'(s) = nL.$

   For each guard gi:
   gi has an incoming edge (s,gi) with lower bound L and outgoing edges with lower bound 0:

   $b(gi) = L- 0 = L \Rightarrow d'(gi) = -L.$

   For each interval Ij:
   Ij has an outgoing edge (Ij,t) with lower bound 1 and incoming edges with lower bound 0:

   $b(Ij) = 0 -1 = -1 \Rightarrow d'(Ij) = 1.$

   For the sink t:
   t has m incoming edges (one from each interval) each with lower bound 1:

   $b(t) = m - 0 = m \Rightarrow d'(t) = -m.$

4. **Introducing a Super-Source and Super-Sink:**

   Super-Source S and Super-Sink T:
   We now add two new nodes to balance the imbalances:

   For every vertex v with $d'(v) < 0$, add an edge (S, v) with capacity $|d'(v)|$.

   For every vertex v with $d'(v) > 0$, add an edge (v, T) with capacity $d'(v)$.

   In our network:

   For each guard gi:

Since d′(gi) = -L, add edge (S, gi) with capacity L.

For the sink t:
Since d′(t) = -m, add edge (S, t) with capacity m.

For the source s:
Since d′(s) = nL, add edge (s, T) with capacity nL.

For each interval Ij:
Since d′(Ij) = 1, add edge (Ij, T) with capacity 1.

5. **Solving the Max Flow:**
   Solve the maximum flow problem from S to T on the transformed network. If the maximum flow saturates all edges leaving S (or equivalently, if the total flow equals the sum of all positive imbalances, which in this case is m+nL), then a feasible circulation exists in the original network. This feasible circulation corresponds to a valid assignment of guards to intervals meeting all the original constraints.

In-short

Flow Network Vertices:
s: Original source.
t: Original sink.
gi for each guard i.
Ij for each interval j.

Edges and Their Purposes:

(s, gi) with lower bound L and capacity Q: Enforces that each guard covers at least L and at most Q intervals.

(gi, Ij) with capacity 1 (if guard i is available for interval j): Ensures a guard covers an interval at most once.

(Ij, t) with lower bound 1 and capacity 2: Guarantees each interval is covered by at least one and at most two guards.

(t,s) with capacity ∞: Completes the circulation.

Reduction to Standard Max Flow:

Adjust each edge by pushing the lower bound.

Define residual capacities $c'(e)=c(e)-l(e)$.

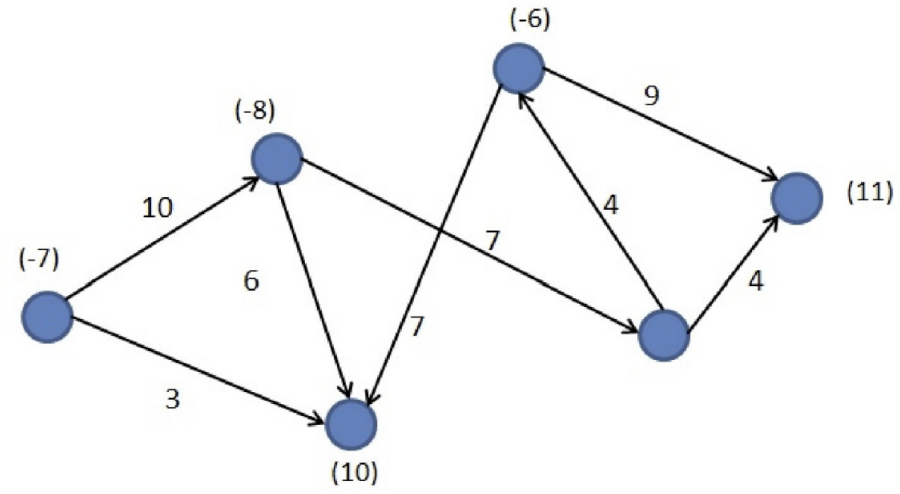
Compute node imbalances b(v) and adjust demands.

Introduce a super-source S and super-sink T with appropriate edges based on the imbalances.

Solve the resulting standard max flow problem from S to T. A full flow (equal to the sum of the positive imbalances, m+nL) indicates that a valid guard assignment exists.

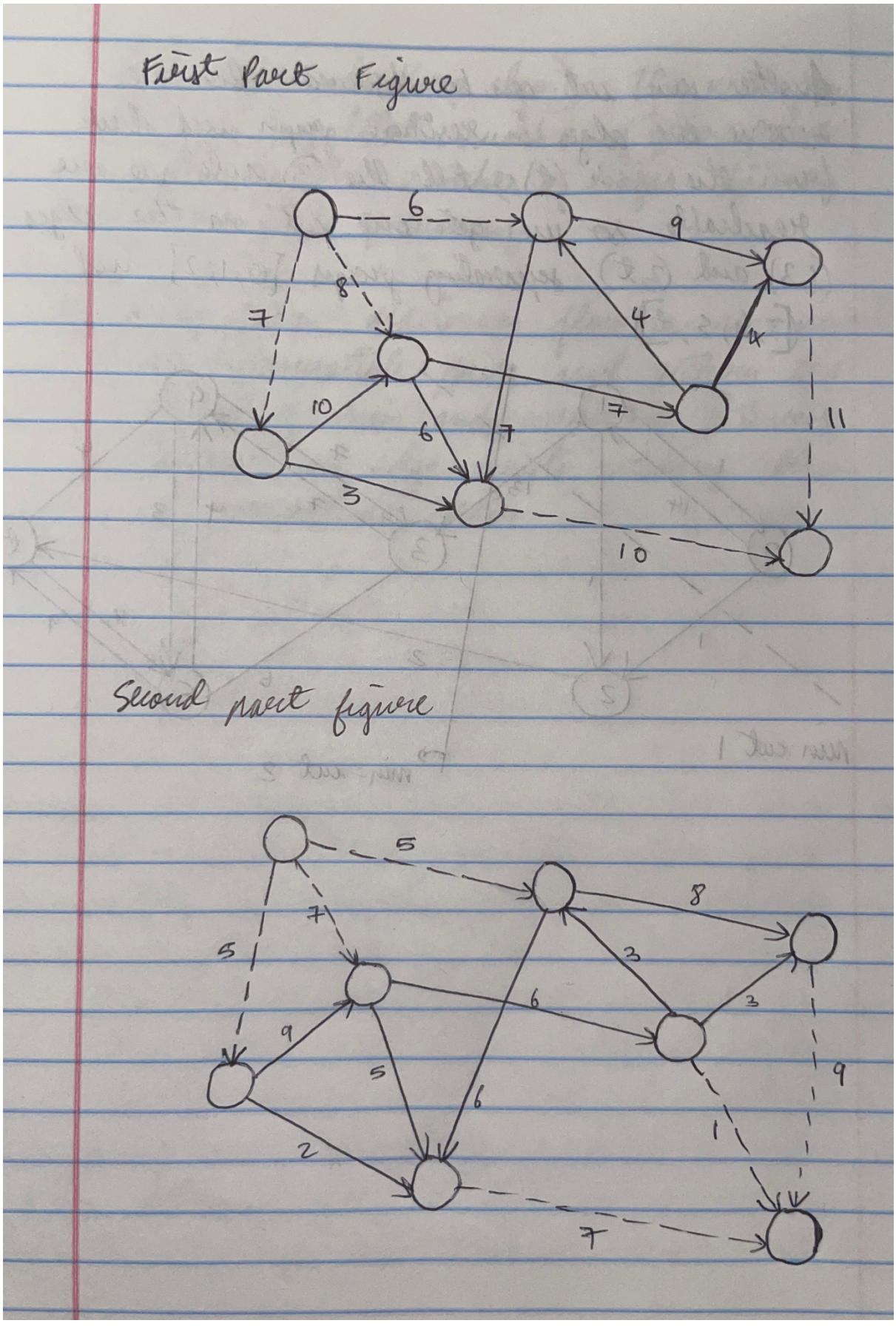
Question 4]
Graph G is an instance of a circulation problem with demands. The edge weights

indicate capacities, and the node weights (shown in parentheses) indicate demands. A negative demand indicates that a node acts as a source.



• Transform this graph into an instance of max-flow problem. Just draw the new graph.
• Now, assume that each edge of G has a constraint of lower bound of 1 unit, i.e., one unit must flow along all edges. Find the new instance of max-flow problem that includes the lower bound constraint. Just draw the new graph.

Solution 4]


First Part Figure
6
9
8
7
4
4
10
7
6
7
11
3
10
Second part figure
5
8
7
5
3
6
3
9
5
6
9
1
2
7