CSE 202 Homework #3

1. **Problem 1: Graph cohesiveness**

   (1) Denote $G = (V, E)$, we label each vertex in V as $v_1, v_2, \cdots, v_n$ and each edge in E as $e_1, e_2, \cdots, e_m$. The endpoints of the edge e_i are v_{l_i} and v_{r_i}. A rational number α is given. We assume that α is non-negative, otherwise it is trivial that every set has cohesiveness greater than α. We also assume that $m \geq 1$, otherwise the cohesiveness is always 0.

   High-level description

   We construct a network with vertices:

   - The source s and sink t.
   - m vertices $b_1, b_2, \cdots, b_m$ which represent $e_1, e_2, \cdots, e_m$, respectively.
   - n vertices $c_1, c_2, \cdots, c_n$ which represent $v_1, v_2, \cdots, v_n$, respectively.

   The flow graph contains directed edges:

   - From s to each b_i $(1 \leq i \leq m)$ with capacity 1.
   - From each b_i to both c_{l_i} and c_{r_i} with capacity ∞.
   - From each c_i $(1 \leq i \leq n)$ to t with capacity α.

   Denote the maximum flow of this network as f. We claim that there exists a set S with cohesiveness greater than α if and only if f is less than m.

   Note that α is a rational number, i.e., $\alpha = \dfrac{\alpha_0}{\alpha_1}$ where both α_0 and α_1 are non-negative integers. In practice, we can multiply all the capacities by α_1 to make them be integers. As a result, we will check whether the maximum flow f is less than $m \times \alpha_1$ instead of m.

   For the correctness proof below, we will use the original capacities, but it is noteworthy that we must provide a network with all integer capacities.

   Proof of correctness

   Claim 1. *The maximum flow is finite and no more than m.*

   Proof. The capacity of all the outgoing edges of the source vertex s is m. Thus the value of the maximum flow should not exceed m. □

Claim 2. *If the maximum flow f is less than m, we can construct a subset with cohesiveness greater than α.*

Proof. Denote a minimum s-t cut of the network as (P, Q) where $s \in P$ and $t \in Q$. As the type b and c vertices are divided into P and Q, we denote $P = \{s\} \cup B_P \cup C_P$ and $Q = \{t\} \cup B_Q \cup C_Q$, where $B_P \cup B_Q = \{b_1, \cdots, b_m\}$ and $C_P \cup C_Q = \{c_1, \cdots, c_n\}$. B_P could not be empty, otherwise as all the type b vertices are in Q, the minimum cut is at least m. Since the capacity of the minimum cut is finite, we cannot have any $b_i \in P$ but either $c_{l_i} \in Q$ or $c_{r_i} \in Q$, because the edge $b_i \to c_{l_i}$ and $b_i \to c_{r_i}$ have infinite capacity. Then for $b_i \in P$, we must have both $c_{l_i} \in P$ and $c_{r_i} \in P$.

On the other hand, if both $c_{l_i} \in P$ and $c_{r_i} \in P$, we must have $b_i \in P$. If $b_i \in Q$, we can move b_i to P and decrease the capacity of the cut by 1, which contradicts our assumption of a minimum cut.

Thus, if we take C_P as the subset, all the edges that have both endpoints in C_P appear in B_P. As the minimum cut is made up of:

- From s to each $b_i \in Q$ with capacity 1, where the sum is $m - e(C_P)$.
- From each $c_i \in P$ to t with capacity α, where the sum is $\alpha \times |C_P|$.

the total capacity is $m - e(C_P) + \alpha \times |C_P|$. It is less than m:

$$\begin{aligned} & m - e(C_P) + \alpha \times |C_P| < m \\ \Leftrightarrow\ & e(C_P) - \alpha \times |C_P| > 0 \\ \Leftrightarrow\ & \frac{e(C_P)}{|C_P|} > \alpha \end{aligned}$$

which indicates the subset C_P is a set with cohesiveness greater than α. □

Claim 3. *If the maximum flow f is m, there is no subset with cohesiveness greater than α.*

Proof. We can prove by contradiction. If there is a subset C'_P with cohesiveness greater than α, denote the set of edges that have both endpoints in C'_P as B'_P, we have $\frac{|B'_P|}{|C'_P|} > \alpha$ We can construct an s-t cut (P', Q') where:

- $P' = s \cup B'_P \cup C'_P$
- $Q' = t \cup (\{b_1, \cdots, b_m\} \backslash B'_P) \cup (\{c_1, \cdots, c_n\} \backslash C'_P)$

and the total capacity of this cut is $m - |B'_p| + \alpha \times |C'_p|$:

$$m - |B'_p| + \alpha \times |C'_p| < m - \alpha \times |C'_p| + \alpha \times |C'_p| = m$$

which is less than m. Thus it contradicts the assumption that the maximum flow is m. □

Time complexity

The number of vertices in the network is $2 + m + n = O(n^2)$. By using the push-relabel algorithm to find the maximum flow, we could solve this problem in $O((n^2)^3) = O(n^6)$ time.

(2) We can use a binary search together with the algorithm described in (1) to find the set with maximum cohesiveness. We first put all $\frac{i}{j}$ into a list *cand*, where $0 \leq i \leq m$ and $1 \leq j \leq n$, indicating all the possible values of α.

We then sort *cand* in ascending order and do a binary search over it, ending with a maximum value α' indicating that there exists a set with cohesiveness greater than α'. We claim that the maximum cohesiveness is just the next greater element of α' in *cand*, and the specific set can be constructed by the method described in Claim 2.

We can prove the correctness by contradiction. If the maximum cohesiveness is larger than the next greater element of α' (denoted as α'_{+1}), our binary search will end in finding a value no smaller than α'_{+1}, which contradicts the finding α'.

As α has $(m+1)n = O(n^3)$ possible values, the binary search takes $O(\log n^3) = O(\log n)$ time. During each step of the binary search, we find the maximum flow of the corresponding network, thus the total time complexity is $O(\log n) \times O(n^6) = O(n^6 \log n)$.

2. **Problem 2: Remote Sensor**

High-level description

We construct a network with vertices:

- The source s and sink t.
- m vertices $b_1, b_2, \cdots, b_m$ which represent the base stations $B_1, B_2, \cdots, B_m$, respectively.
- n vertices $c_1, c_2, \cdots, c_n$ which represent the remote sensors $s_1, s_2, \cdots, s_n$, respectively.

The flow graph contains directed edges:

- From s to each b_i $(1 \leq i \leq m)$ with capacity C.
- From each b_i to each c_j with capacity ∞, as long as the Euclidean distance between the base station B_i and the remote sensor s_j is no more than 2 km.
- From each c_i $(1 \leq i \leq n)$ to t with capacity r_i.

Denote the maximum flow of this network as f. If f equals the sum of all r_i entries, we claim that there exists a solution which meets all the constraints. The solution can be constructed by the following:

- For each base station B_i and each remote sensor s_j, if the edge $b_i \to c_j$ exists and the flow (denoted as $f_{i,j}$) is greater than 0, we assign $f_{i,j}$ bandwidth on base station B_i to remote sensor s_j.

Otherwise, there is no valid solution and we would say "No solution exists".

Proof of correctness

Claim 4. *Denote the sum of all r_i entries as R, the maximum flow is finite and no more than R.*

Proof. The capacity of all the incoming edges of the sink vertex t is just R, thus proves the claim. □

Claim 5. *If the maximum flow f equals R, our algorithm will find a solution which meets all the constraints.*

Proof. As there is no edge from b_i to c_j when the Euclidean distance between base station B_i and the remote sensor s_j is more than 2 km, we will never assign any bandwidth in this case.

For every base station B_i, the only incoming edge of the corresponding vertex b_i is $s \to b_i$ with capacity C, thus the total flows of the outgoing edges of b_i should be no more than C, which means the bandwidth assigned on the base station B_i is no more than C.

For all the remote sensors, we construct an s-t cut (P, Q) where $Q = \{t\}$ and $P =$ all other vertices in the network. The capacity of this cut is R, which shows (P, Q) is a minimum cut, thus each edge from P to Q must have its flow equals capacity.

For every remote sensor s_i, the only outgoing edge of the corresponding vertex c_i is $c_i \to t$ with capacity r_i, as this edge is from P to Q, the flow must be exactly r_i. Thus the total flows of the incoming edges of b_i should also be r_i, which means the bandwidth assigned to the remote sensor s_i is exactly r_i (and it definitely satisfies $\geq r_i$).

Thus the solution found by our algorithm meet all the constraints. □

Lemma 6. *If there is a solution in which exists a remote sensor s_i that the sum of all the bandwidth assigned to it is strictly larger than r_i, we can derive another solution where the sum is exactly r_i.*

Proof. We can do this by decreasing r_i by 1 every step. We find a base station B_j which has assigned its bandwidth to remote sensor s_i. We can always find one because $r_i \geq 0$. We decrease the assigned bandwidth by 1. As the only constraint on B_j is the sum of all bandwidth assigned should be $\leq C$, decreasing bandwidth will never violate the constraint. □

Claim 7. *If the maximum flow f is less than R, there is no valid solution which meets all the constraints.*

Proof. We can prove by contradiction. Assume there is a valid solution, by Lemma 6, we can derive another valid solution where each remote sensor s_i has exactly r_i bandwidth assigned to it. We can then construct a network flow:

- $s \to b_i$: the flow is the total bandwidth assigned on base station B_j.
- $b_i \to c_j$, the flow is the bandwidth assigned on base station B_i to sensor node s_j.
- $c_i \to t$, the flow is the bandwidth assigned to sensor node s_i, which is exactly r_i.

which is valid and its value is R. Thus it contradicts the assumption that the maximum flow is less than R. □

Time complexity

The number of vertices in the network is $2+m+n = O(n)$. By using the push-relabel algorithm to find the maximum flow, we could solve this problem in $O(n^3)$ time.

3. **Problem 3: Scheduling in a medical consulting firm**

(1) **High-level description**

We construct a network with vertices:

- The source s and sink t.
- n vertices $b_1, b_2, \cdots, b_n$ which represent the day $1, 2, \cdots, n$, respectively.
- k vertices $c_1, c_2, \cdots, c_k$ which represent the doctor $1, 2, \cdots, k$ respectively.

The flow graph contains directed edges:

- From s to each b_i $(1 \leq i \leq n)$ with capacity p_i.
- From each b_i to each c_j with capacity 1, as long as the i-th day is in the list L_j.
- From each c_i $(1 \leq i \leq k)$ to t with capacity ∞.

Denote the maximum flow of this network as f. If f equals the sum of all p_i entries, we claim that there exists a solution which satisfies all the properties, i.e., for each day i there are exactly p_i doctors, and for each doctor i he/she only works on a subset of L_i. The solution can be constructed by the following:

- For each return list L'_i, it contains all the j indices that there is an edge from b_j to c_i and the flow of that edge is 1.

Otherwise, there is no valid solution and we would report it.

Proof of correctness

Claim 8. *Denote the sum of all p_i entries as M, the maximum flow is finite and no more than M.*

Proof. The capacity of all the outgoing edges of the source vertex s is just M, thus proves the claim. □

Claim 9. *If the maximum flow f equals M, our algorithm will find a solution which satisfies all properties.*

Proof. For each doctor i, there is an incoming edge from b_j to the corresponding vertex c_i if and only if $j \in L_i$, thus he/she will only work on the days in his/her list.

For all the days, we construct an s-t cut (P, Q) where $P = \{s\}$ and $Q =$ all other vertices in the network. The capacity of this cut is M, which shows (P, Q) is a minimum cut, thus each edge from P to Q must have its flow equals capacity.

For each day i, the only incoming edge of the corresponding vertex b_i is $s \to b_i$ with capacity p_i, as this edge is from P to Q, the flow must be exactly p_i. Thus the total flows of the outgoing edges of b_i should also be p_i, which means exactly p_i doctors work on day i.

Thus our algorithm will find a solution which satisfies all properties. □

Claim 10. *If the maximum flow f is less than M, there is no valid solution which satisfies all properties.*

Proof. We can prove by contradiction. Assume there is a valid solution, we can then construct a network flow:

- $s \to b_i$: the flow is the exactly p_i.
- $b_i \to c_j$, the flow is 1 if doctor j works on day i, otherwise the flow is 0.
- $c_i \to t$, the flow is the number of days that docker j works in total.

which is valid and its value is M. Thus it contradicts the assumption that the maximum flow is less than M. □

Time complexity

The number of vertices in the network is $2 + n + k = O(n + k)$. By using the push-relabel algorithm to find the maximum flow, we could solve this problem in $O((n + k)^3)$ time.

(2) **High-level description**

We construct a network with vertices:

- The source s and sink t.
- n vertices $b_1, b_2, \cdots, b_n$ which represent the day $1, 2, \cdots, n$, respectively.
- k vertices $c_1, c_2, \cdots, c_k$ which represent the doctor $1, 2, \cdots, k$ respectively, the type c vertices indicate that each doctor works on the day in his/her list.
- k vertices $c'_1, c'_2, \cdots, c'_k$ which represent the doctor $1, 2, \cdots, k$ respectively, the type c' vertices indicate that each doctor works on the day not in his/her list.

The flow graph contains directed edges:

- From s to each b_i $(1 \le i \le n)$ with capacity p_i.
- From each b_i to each c_j with capacity 1, as long as the i-th day is in the list L_j.
- From each b_i to each c'_j with capacity 1, as long as the i-th day is not in the list L_j.
- From each c_i $(1 \le i \le k)$ to t with capacity ∞.
- Form each c'_i $(1 \le i \le k)$ to t with capacity c.

Denote the value of the maximum flow of this network as f. If f equals the sum of all p_i entries, we claim that there exists a solution which satisfies all the properties, i.e., for each day i there are exactly p_i doctors, and for each doctor i, he/she only works on a subset of L_i together with at most c days not in L_i. Every return list L'_i is constructed as follows: if there is an edge from b_j to c_i or c'_i and the flow of that edge is 1, we include j to the list L'_i.

Otherwise, there is no valid solution and we would report it.

Proof of correctness

Claim 11. *Denote the sum of all p_i entries as M, the maximum flow is finite and no more than M.*

Proof. The capacity of all the outgoing edges of the source vertex s is just M, thus proves the claim. □

Claim 12. *If the maximum flow f equals M, we can construct a solution which satisfies all properties.*

Proof. For each doctor i, there is an incoming edge from b_j to the corresponding vertex c_i if and only if $j \in L_i$, thus he/she will only work on the days he/she finds acceptable.

For all the days, we construct an s-t cut (P, Q) where $P = \{s\}$ and $Q =$ all other vertices in the network. The capacity of this cut is M, which shows (P, Q) is a minimum cut, thus each edge from P to Q must have its flow equals capacity.

For each day i, the only incoming edge of the corresponding vertex b_i is $s \to b_i$ with capacity p_i, as this edge is from P to Q, the flow must be exactly p_i. Thus the total flows of the outgoing edges of b_i should also be p_i, which means exactly p_i doctors work on day i.

Thus if we construct the solution where doctor i works on the day j as long as there is an edge from b_j to c_i and its flow is 1, it will meet all the constraints. □

Claim 13. *If the maximum flow f is less than M, there is no valid solution which satisfies all properties.*

Proof. We can prove by contradiction. Assume there is a valid solution, we can then allocate flows on the network:

- $s \to b_i$: the flow is the exactly p_i.
- $b_i \to c_j$, the flow is 1 if doctor j works on day i, otherwise the flow is 0.
- $c_i \to t$, the flow is the number of days that docker j works in total.

The above flow is valid and its value is M, which contradicts the assumption that the maximum flow is less than M. □

Time complexity

The number of vertices in the network is $2 + n + k = O(n + k)$. By using the push-relabel algorithm to find the maximum flow, we could solve this problem in $O((n + k)^3)$ time.

4. **Problem 4: Cellular network**

Assume there are n sites labeled $1, 2, \cdots, n$. We also assume that for each benefit $b_{i,j}$, it is valid if and only if $i < j$:

- When $i = j$, the sites are the same thus there will be no traffic revenue.
- When $i > j$, it should be the same as $b_{j,i}$.

High-level description

We construct a network with vertices:

- The source s and sink t.
- $\frac{n(n-1)}{2}$ vertices $r_{1,2}, r_{1,3}, \cdots, r_{1,n}, r_{2,3}, \cdots, r_{2,n}, \cdots, r_{n-1,n}$ which represent $b_{1,2}, b_{1,3}, \cdots, b_{1,n}, b_{2,3}, \cdots, b_{2,n}, \cdots, b_{n-1,n}$, respectively.
- n vertices $s_1, s_2, \cdots, s_n$ which represent the sites $1, 2, \cdots, n$, respectively.

The flow graph contains directed edges:

- From s to each $r_{i,j}$ $(1 \leq i < j \leq n)$ with capacity $b_{i,j}$.
- From each $r_{i,j}$ to both s_i and s_j with capacity ∞.
- From each s_i $(1 \leq i \leq n)$ to t with capacity c_i.

Denote the maximum flow of this network as f and the sum of all $b_{i,j}$ entries as B. We claim that the maximum sum of benefits less the node costs is $B - f$. The solution can be constructed by the following:

- Denote any minimum s-t cut as (P, Q) where $s \in P$ and $t \in Q$, extract all the type s vertices representing sites in P as the subset.

Proof of correctness

Claim 14. *The maximum flow is finite and no more than B.*

Proof. The capacity of all the outgoing edges of the source vertex s is B, thus proves the claim. □

Claim 15. *If the minimum cut is f, our algorithm will find a solution where the sum of benefits less the node costs is $B - f$.*

Proof. Denote a minimum s-t cut of the network as (P, Q) where $s \in P$ and $t \in Q$. As the type r and s vertices are divided into P and Q, we denote $P = \{s\} \cup R_P \cup S_P$ and $Q = \{t\} \cup R_Q \cup S_Q$, where $R_P \cup R_Q = \{r_{1,2}, \cdots, r_{n-1,n}\}$ and $S_P \cup S_Q = \{s_1, \cdots, s_n\}$.

Since the capacity of the minimum cut is finite, we cannot have any $r_{i,j} \in P$ but either $s_i \in Q$ or $s_j \in Q$, because the edge $r_{i,j} \to s_i$ and $r_{i,j} \to s_j$ have infinite capacity. Then for $r_{i,j} \in P$, we must have both $s_i \in P$ and $s_j \in P$.

On the other hand, if both $s_i \in P$ and $s_j \in P$, we must have $r_{i,j} \in P$. If $r_{i,j} \in Q$, we can move $r_{i,j}$ to P and decrease the capacity of the cut by $b_{i,j}$, which contradicts our assumption of a minimum cut.

If we take S_P as the subset, all the traffic revenue that have both sites in S_P appear in R_P. As the minimum cut is made up of:

- From s to each $r_{i,j} \in Q$ with capacity $b_{i,j}$, where the sum is $B - \sum_{r_{i,j} \in P} b_{i,j} = B - \sum_{s_i, s_j \in P} b_{i,j}$.
- From each $s_i \in P$ to t with capacity c_i, where the sum is $\sum_{s_i \in P} c_i$.

The total capacity is $B - \sum_{s_i, s_j \in P} b_{i,j} + \sum_{s_i \in P} c_i$. It equals f:

$$\begin{aligned} & B - \sum_{s_i, s_j \in P} b_{i,j} + \sum_{s_i \in P} c_i = f \\ \Leftrightarrow \quad & \sum_{s_i, s_j \in P} b_{i,j} - \sum_{s_i \in P} c_i = B - f \end{aligned}$$

Thus if we choose S_P as the subset, the sum of benefits less the node costs is $B - f$. □

Claim 16. *If the minimum cut is f, there is no subset of sites where the sum of benefits less the node costs is strictly more than $B - f$.*

Proof. We can prove by contradiction. If there exists a subset of sites S'_P and its corresponding traffic revenue R'_P, we have $\sum_{s_i, s_j \in S'_P} b_{i,j} - \sum_{s_i \in S'_P} c_i > B - f$.

We can construct an s-t cut (P', Q') where:

- $P' = s \cup R'_P \cup S'_P$
- $Q' = t \cup (\{r_{1,2}, \cdots, r_{n-1,n}\} \backslash R'_P) \cup (\{s_1, \cdots, s_n\} \backslash S'_P)$

and the total capacity of this cut is:

$$B - \sum_{s_i, s_j \in S'_P} b_{i,j} + \sum_{s_i \in S'_P} c_i < B - (B - f) = f$$

which is less than f. Thus it contradicts the assumption that the maximum flow is f. □

Time complexity

The number of vertices in the network is $2 + \frac{n(n-1)}{2} + n = O(n^2)$. By using the push-relabel algorithm to find the maximum flow, we could solve this problem in $O((n^2)^3) = O(n^6)$ time.