CSCE 411
Max Flow and Bipartite Matching

---

1. In the *stable roommates* problem, you are given as input a set of $2n$ students (where n is some integer) and a preference list for each student ranking all $2n-1$ of the other students. The goal is to find a way to arrange the students into roommates such there is no unstable pair. As in the stable matching problem, we define an unstable pair as a pair that prefer each other over their current partners.

   Our proof that Gale-Shapley always finds a stable matching proves as a side effect that the stable matching problem always has a solution. Prove that the stable roommates problem does *not* always have a valid solution.

   Solution: Suppose that we have four students, A, B, C, and D with the following preference lists:

   $$A : B, C, D$$
   $$B : C, A, D$$
   $$C : A, B, D$$
   $$D : A, B, C$$

   We claim that this set of students has no stable solution. With four students there are only 3 possible pairings:

   (a) AB and CD: In this case, B and C form an unstable pair since B prefers C over A and C prefers B over D.

   (b) AC and BD: In this case, A and B form an unstable pair since A prefers B over C and B prefers A over D.

   (c) AD and BC: In this case, A and C form an unstable pair since A prefers C over D and C prefers A over B.

   Since every possible matching has an unstable pair, there is no valid solution to this instance of stable roommates.

2. Managers at Bureaucracy Inc. are concerned that the company isn't getting anything done. They've decided that the natural response is to form a collection of committees aimed at addressing this issue. You are given:

   - The number of workers x, and the number of committees y.
   - An array of positive integers $A[1, \ldots, x]$ where $A[i]$ is the maximum number of committees that the i'th worker can handle serving on.
   - A 2D array of booleans $B[1, \ldots, x][1, \ldots, y]$ where $B[i][j]$ is true if the i'th worker can serve on the j'th committee.
   - An array of positive integers $C[1, \ldots, y]$ where $C[i]$ is the number of members that the i'th committee needs

   Your goal is to determine whether there exists an assignment of workers to committees such that each worker is assigned only to committees that they are able to serve on, each committee has as many members as it needs, and no worker is serving on more committees than they can handle. Give an efficient algorithm for this problem.

Solution: We solve this problem using a reduction to maximum flow. We create a flow network $(G = (V, E), c, s, t)$ as follows:

- $V := V_{Work} \cup V_{Com} \cup \{s, t\}$, where:
  - $V_{Work} := w_1, \ldots, w_x$ is a set of vertices representing workers,
  - $V_{Com} := c_1, \ldots, c_y$ is a set of vertices representing committees,
  - s and t are two additional vertices not part of V_{Work} or V_{Com}.
- E contains the following edges
  - For each worker vertex w_i, we have an edge (s, w_i) with capacity $A[i]$.
  - For each pair of a worker vertex w_i and a committee vertex c_i where $B[i][j] =$True, we have an edge (w_i, c_j) with capacity 1.
  - For each committee vertex c_i we have an edges (c_i, t) of capacity $C[i]$.

Our algorithm then computes an (integer) maximum flow in this network and returns true if the maximum flow value is $\sum_{i=1}^{y} C[i]$ and false otherwise.

To show the correctness of our algorithm, we need to show that the value of a maximum flow in our network is $\sum_{i=1}^{y} C[i]$ if and only if it is possible to serve the customers in a way that satisfies all of the constraints. There are two directions to show.

First we show that if it is possible to satisfy all of the constraints, then our network will have a max-flow of value $\sum_{i=1}^{y} C[i]$. Consider some valid assignment of workers to committees. We create a flow f where:

- For each $1 \leq i \leq x$, we send a number of units of flow along (s, w_i) equal to the number of committees the i'th worker is assigned to (since our committee assignment satisfies the constraints we know this worker is on at most $A[i]$ committees).
- For each edge (w_i, c_j) we send 1 unit of flow along it if the i'th worker is in the j'th committee.
- For each edge (c_i, t) we send $C[i]$ units of flow (This satisfies flow conservation since we know a valid assignment must assign $C[i]$ workers to the i'th committee).

We note that the value of this flow is $\sum_{i=1}^{y} C[i]$, since that is the amount of flow we are sending into t.

Now we need to show that if our network has an integer flow of value x, then it is possible to find a valid committee assignment. Let f be an integer maximum flow of value $\sum_{i=1}^{y} C[i]$. Since the total capacity of the incoming edges of t is only $\sum_{i=1}^{y} C[i]$, f must fully saturate all of these edges. Therefore for $1 \leq j \leq n$, the vertex c_j must have incoming flow from $C[j]$ different worker vertices (since each incoming edge has capacity 1). We assign the workers corresponding to those vertices to the j'th committee. This assignment cannot assign any worker to more committees than they can handle, since f can send at most $A[i]$ units of flow into each worker vertex w_i.

Finally, we analyze the runtime of our algorithm. The input size n for our problem is $\Theta(xy)$. Our graph has $O(x + y)$ vertices and $O(xy)$ edges, so we can find an (integer) max-flow in it in time $O((xy)^{1+o(1)}) = O(n^{1+o(1)})$.

Note: We could directly solve this problem using bipartite matching (instead of max flow) if we made $A[i]$ vertices for the i'th employee and $C[j]$ vertices for the j'th committee and looked for a perfect matching in the resulting graph. This approach could yield a graph with xy vertices and $(xy)^2$ edges, however, so the runtime would be worse.

3. Give a polynomial time algorithm to determine if an instance of the stable matching problem has a

unique solution (i.e. if only one possible matching is stable).

Solution: We just run Gale-Shapley twice, once with the hospitals making offers and once with the doctors making offers. If both runs of the algorithm yield the same matching we report that this is the unique stable matching, otherwise, we report that there is not a unique solution.

Correctness: If the two runs of Gale-Shapley produce different outputs then have found two different stable matchings so there is no unique solution. If the two runs of Gale-Shapley produce the same output, we claim that this output is the only stable matching. To see this, we use the fact from lecture that Gale-Shapley yields a hosptial-optimal matching (i.e. an optimal matching for the side making offers).

Let M be the matching output by both runs of Gale-Shapley, and suppose for contradiction that there was another stable matching M'. Let h be some hospital which is matched to a different doctor under M' than under M. Let d be the doctor that h is paired with in M, let d' be the doctor that h is paired with in M', and let h' be the hospital that d is paired with in M'.

We know that when hospitals make offers, Gale-Shapley gives a hospital-optimal matching. Therefore, since h ends up with d in the run of Gale-Shapley where hospitals make offers, d is the doctor that h prefers most out of all doctors that h could be paired with in a stable matching. In particular, h prefers d to d'. Similarly, we know that when doctors make offers, Gale-Shapley gives a doctor-optimal matching. Therefore, since d ends up with h in the run of Gale-Shapley where doctors make offers, h is the doctor that d prefers most out of all doctors that d could be paired with in a stable matching. In particular, d prefers h to h'.

Thus, h and d form an unstable pair in M', because they both prefer each other over their current partners. This contradicts our assumption that M' was a stable matching, so M is the unique stable matching.

Runtime: Gale-Shapley runs in time $\Theta(n^2)$, where n is the number of hospitals/doctors. We perform two runs of Gale-Shapley and then do $\Theta(n)$ additional work to check if the two matchings returned are the same, so the overall runtime is $\Theta(n^2)$.

4. Prove that a graph is bipartite if and only if it has no cycles of odd-length.

Solution: There are two directions to prove.

- If G is bipartite then it has no cycle of odd length.

  Let C be a cycle in a bipartite graph G. Let $x_1, \ldots, x_k$ be the vertices of C (in order). Since G is bipartite, it is possible to assign $x_1, \ldots, x_k$ to sets L and R such that every edge of the cycle goes between L and R. Assume without loss of generality that $x_1 \in L$. Then, we must have $x_2 \in R$, and $x_3 \in L$. Continuing the pattern, we can show by induction that x_i must be in L if i is odd and in R if i is even. Since G is bipartite, the edge $\{x_k, x_1\}$, must be have one endpoint in L and the other in R. Since x_1 is in L, this means x_k is in R. Therefore, by our pattern, k must be even, and thus C is a cycle of even length. Since C was an arbitrary cycle in G, we conclude that G has no odd-length cycles.

- If G has no odd cycle, then it is bipartite. We prove this by giving a valid bipartition of V. We will assume without loss of generality that G is connected. If this is not the case, we can separately find a bipartition (L_C, R_C) for each component C of G and then define $L := \bigcup_{C\colon C \text{is a component of } G} L_C$ and $R := \bigcup_{C\colon C \text{is a component of } G} R_C$.

  Let $s \in V$ be an arbitrary vertex. Since G is connected, for every vertex $x \in V$, $dist(s, x)$ is some finite value. Let $L := \{x \in V\colon dist(s, x) \text{ is even}\}$ and let $R := \{x \in V\colon dist(s, x) \text{ is odd}\}$. We note that $s \in L$. We need to show that there is no edge $\{u, v\}$ where $dist(s, u)$ and $dist(s, v)$ are either both even or both odd. Suppose for contradiction that there was such an edge. Consider a shortest path tree rooted at s. Let x be the lowest common ancestor of u and v in this tree. Since $dist(s, u) \equiv dist(s, v) \pmod 2$, we have that $dist(x, u) \equiv dist(x, v) \pmod 2$, and thus, the cycle formed by combining $\{u, v\}$ with the $x \to u$ and $v \to x$ paths in T is an odd-length cycle. But since G has no odd cycle, this is a contradiction. We conclude that the edge $\{u, v\}$ cannot exist and so (L, R) is a valid bipartition.