Ling Li, ling@cs.caltech.edu

14.1 Construct the level graph

Let's modify BFS slightly to construct the level graph $L_G = (V', E')$ of $G = (V, E, c)$ out of a source $s \in V$. We will use a queue Q and an array l containing levels of vertices.

Initialize. $Q = [s]$; Allocate n units for l, where $n = |V|$; $l(s) = 0$ and $l(u) = -1$ for all other vertices; $V' = \{s\}$, $E' = \emptyset$.

Advance. $u := \text{pop}(Q)$. $\forall (u, v) \in E$: if $l(v) < 0$, $l(v) := l(u) + 1$, $\text{push}(Q, v)$, $V' = V' \cup \{v\}$, $E' = E' \cup \{(u, v)\}$. Repeat **Advance** until Q is empty.

The time for **Initialize** is $O(n)$. In **Advance**, any vertex requires at most 1 pair of push/pop operation, and other operations are performed at most once for every edge. Thus the time is $O(m + n)$. The total running time is $O(m + n)$.

Ling Li, ling@cs.caltech.edu

14.2 Binary capacities

Apply Dinic's algorithm to G. I want to show that if all capacities of G are 0 or 1, the time complexity is $O(mn)$. Below the bold font refers to operations in Dinic's algorithm.

When all capacities are 0 or 1, all edges in a path flow are saturated thus will all be deleted in **Augment**. Thus in one phase, each edge of G will be operated by **Advance**, **Retreat** and **Augment** at most once (totally at most 3 operations). And in **Retreat**, a vertex is deleted with its edges, so no more than $O(m)$ time is needed. Together with the $O(m)$ time needed for **Initialize**, the total time for one phase is still $O(m)$. Because there are at most n phases, the total time is $O(mn)$.

Ling Li, ling@cs.caltech.edu

14.3 König-Egerváry theorem

For any matching of size k and any cover with size k' in A, we have $k \leq k'$. This is because any pair $\{i_\ell, j_\ell\}$ in the matching needs a row or column to cover, and those rows and columns must be distinct since all i_ℓ are distinct and all j_ℓ are distinct. Thus $k_{\max} \leq k'_{\min}$, where $k_{\max}$ is the maximum size of a matching and $k'_{\min}$ is the least size of a cover.

Consider A as the adjacency matrix of a bipartite graph $G = (U, V, E)$, with $U = \{u_i\}$, $V = \{v_j\}$, and a directed edge $(u_i, v_j) \in E$ iff $A_{i,j} = 1$. Thus the vertices in set U correspond to rows in A and the vertices in set V correspond to rows in B. Add a new source vertex s and a new sink vertex t, connect s to every vertex in U, and connect every vertex in V to t. Assign every edge capacity 1. Thus we get a capacitated graph G'.

Any integral flow f in G' corresponds to a matching in A, if we take every edge (u_i, v_j) used by f as a pair $\{i, j\}$ in the matching. Conversely, any matching in A corresponds to an integral flow in G'. Therefore, $k_{\max} = |f_{\max}| = c(B_1, B_2)$, where $f_{\max}$ is a max flow and B_1, B_2 a min cut in G'.

Without loss of generality, assume $s \in B_1$ and $t \in B_2$. Let $U_i = B_i \cap U$ and $V_i = B_i \cap V$. Then $B_1 = \{s\} \cup U_1 \cup V_1$ and $B_2 = \{t\} \cup U_2 \cup V_2$. We can construct a cover in A from B_1, B_2 as follows. Let

$$R = \{\text{row } i : u_i \in U_1 \text{ and } \exists v \in V_2 \text{ so that } c(u_i, v) = 1, \text{ or } u_i \in U_2\}$$

and $C = \{\text{column } j : v_j \in V_1\}$. Since for any pair $\{i, j\}$ for which $A_{i,j} = 1$, we have $c(u_i, v_j) = 1$. If $v_j \in V_1$, then column $j \in C$; If $v_j \in V_2$, then row $i \in R$. Thus $R \cup C$ is a cover in A. The size of this cover is

$$|R| + |C| \leq |U_2| + |V_1| + \sum_{u \in U_1, v \in V_2} c(u, v).$$

From

$$\begin{aligned} c(B_1, B_2) &= \sum_{u \in B_1, v \in B_2} c(u, v) \\ &= \sum_{u \in U_2} c(s, u) + \sum_{v \in V_1} c(v, t) + \sum_{u \in U_1, v \in V_2} c(u, v) \\ &= |U_2| + |V_1| + \sum_{u \in U_1, v \in V_2} c(u, v), \end{aligned}$$

and $k'_{\min}$ is the least size of the cover, we observe

$$k'_{\min} \leq c(B_1, B_2) = k_{\max}.$$

So finally we get $k_{\max} = k'_{\min}$.

(In fact, we can construct the min cut B_1, B_2 by let B_1 contain all the vertices that can be visited from s in the residual graph of $f_{\max}$ in G'. Then $\sum_{u \in U_1, v \in V_2} c(u, v)$ must be 0 and thus $c(B_1, B_2) = |R| + |C|$. This can be proved by contradiction. If there is an edge (u, v) with $u \in U_1$, $v \in V_2$, and $c(u, v) = 1$, then (u, v) is also in the residual graph, otherwise there is a flow through (u, v) and then $u \notin B_1$. Thus v can also be reached from s, which contradicts with $v \in B_2$.)

14.4 Max flow with a sequence of augmenting paths

Let f be a max flow in G. By Lemma 17.4 in Kozen's book, f can be expressed as a sum of k ($k \le |E|$) path flows in G (denoted by p_i, $i = 1, 2, \ldots, k$) and a flow in G of value 0. For $i = 0, 1, \ldots, k$, define flow

$$f_i = \sum_{j=1}^{i} p_j.$$

Then f_0 is a null flow, and $|f_k| = |f|$, f_k is a max flow in G.

By the construction of p_i (in the proof of Lemma 17.4) and $|f| \ge 0$, the value of p_i is always positive. Together with $f_i = f_{i-1} + p_i$ is still a flow in G, p_i is an augmenting path associated with flow f_{i-1} for $1 \le i \le k$.

Thus, starting from the null flow f_0, by adding augmenting path p_i for $i = 1, 2, \ldots, k$, we can eventually get a max flow f_k in G.

Note that for *any* max flow in G, we may not be able to write it as the sum of a sequence of augmenting paths, since a zero flow is not necessarily a null flow. For example, $V = \{s, x, y, z, t\}$ and $c(s,x) = c(y,t) = 1$, $c(x,y) = c(y,z) = c(z,x) = 2$. Thus the flow f with $f(s,x) = f(y,t) = f(y,z) = f(z,x) = 1$, $f(x,y) = 2$ is a max flow in G. However, it can not be found by a sequence of augmenting paths. The one that can be found by the above procedures is f with $f(s,x) = f(x,y) = f(y,t) = 1$.

14.5 Edge connectivity

In homework 13.2 (s,t-connectivity problem), by assigning a unit capacity to the graph, we knew that there are exactly k edge-disjoint paths from s to t, where k is the value of any max flow from s to t in G. Thus if we want to disconnect s and t in G, we have to remove at least k edges. On the other hand, we can find a min cut A, B for source s and sink t. Removing all the edges between A and B disconnects s and t. However, from the Max Flow-Min Cut Theorem, the number of edges between A and B, which is also the value of this min cut, is k. Therefore, to disconnect s and t need and only need removing k edges from G.

To disconnect G is equivalent to disconnect a *fixed* vertex s and *some* other vertex t in G. Thus, we can select and fix a vertex s. Then apply the max flow algorithm to G with unit capacities and pair (s,t), for every vertex $t \in V - \{s\}$. For each pair (s,t), we get a max flow with value $k(t)$, which is the minimum number of edges needed to be removed in order to disconnect s and t. Calculate

$$K = \min_{t \in V - \{s\}} k(t).$$

Then it is the edge connectivity of G, i.e., the minimum number of edges that must be removed to disconnect G.

The max flow algorithm runs $|V| - 1$ times. The capacitated G has $2|E|$ edges and $|V|$ vertices, since two directed edges with unit capacities are added for one undirected edge in E. Thus we meet the requirements in the problem.

Ling Li, ling@cs.caltech.edu

14.6 Updating max flow

Lemma: Let $G = (V, E, c)$ and $G^+ = (V, E, c^+)$ be two capacitated graphs. $c^+ = c$ except for edge (u, v), $c^+(u, v) = c(u, v) + 1$. $f_{\max}$ is a max flow in G and $f^+_{\max}$ is a max flow in G^+. Then $|f_{\max}| \le |f^+_{\max}| \le |f_{\max}| + 1$. Thus for integral capacitated graphs, $|f^+_{\max}| = |f_{\max}|$ or $|f^+_{\max}| = |f_{\max}| + 1$.

Proof: Since G and G^+ share the same V and E, a min cut A, B in G is also a cut in G^+. By the construction of c^+, we have $c^+(A, B) \le c(A, B) + 1$. Thus $|f^+_{\max}| \le c^+(A, B) \le c(A, B) + 1 = |f_{\max}| + 1$. Since any flow in G is also a flow in G^+, we also have $|f_{\max}| \le |f^+_{\max}|$. Therefore $|f_{\max}| \le |f^+_{\max}| \le |f_{\max}| + 1$. For integral capacities, this equals $|f^+_{\max}| = |f_{\max}|$ or $|f^+_{\max}| = |f_{\max}| + 1$.

(a) Let $f_{\max}$ be the given max flow in G, and G^+ be the capacitated graph with the capacity of edge (u, v) increased by 1. By the Lemma, the value of a max flow in G^+ is either $|f_{\max}|$ or $|f_{\max}| + 1$. Thus by adding at most 1 augmenting path to $f_{\max}$, we can get a max flow in G^+. The algorithm is

1. Calculate the residual graph $G^+_{f_{\max}}$ associated with $f_{\max}$ in G^+ with time $O(m)$.
2. Find an augmenting path in $G^+_{f_{\max}}$ by BFS with time $O(m)$.
3. If no augmenting path is found, $f_{\max}$ is a max flow in G^+. Otherwise adding the augmenting path to $f_{\max}$ (with time $O(n)$) and the new flow is a max flow in G^+.

Thus, the total time is $O(m + n)$.

(b) Let $f_{\max}$ be the given max flow in G, and G^- be the capacitated graph with the capacity of edge (u, v) decreased by 1. G can be taken as $(G^-)^+$. By the Lemma, the value of a max flow in G^- is either $|f_{\max}|$ or $|f_{\max}| - 1$. As in the proof of the Lemma, we can first find a flow f in G^- with value $|f_{\max}| - 1$ by decreasing $f_{\max}$ along some path. Thus by adding at most 1 augmenting path to f, we can get a max flow in G^-. The algorithm is

1. Find a path p in $f_{\max}$ (that is, an edge (u', v') will be explored iff $f_{\max}(u', v') > 0$) from s to t containing edge (u, v) by BFS, with time $O(m)$.
2. Construct flow f in G^- by decreasing $f_{\max}$ by 1 along p. The time is $O(n)$.
3. Calculate the residual graph G^-_f associated with f in G^- with time $O(m)$.
4. Find an augmenting path in G^-_f by BFS with time $O(m)$.
5. If no augmenting path is found, f is a max flow in G^-. Otherwise adding the augmenting path to f (with time $O(n)$) and the new flow is a max flow in G^-.

Thus, the total time is $O(m + n)$.