Institut for Matematik og Datalogi
Syddansk Universitet

November 7, 2023
JBJ

DM551/MM851 – Fall 2023 – Weekly Note 9

Second set of exam problems
They will be available both from the homepage and from itslearning by the end of the week.

Stuff covered in week 45

I will cover Cormen 26.3 and start on the notes below on flows. We will spend the second hour on November 9 on the midterm evaluation of the course.

Lectures in week 46
There are two lectures and I will cover the topics below in the given order.

- I cover the notes below on flows
- Cormen sections 5.1-5.4.3 Much of the first two sections is already known to you so I will not cover that
- Kleinberg and Tardos 13.6 and Cormen 11.3.3 and 11.5

Exercises in week 46
There are more exercises listed than you can cover so I will cover the last 4 bullets at the lecture on November 16. You should still try to solve them first yourselves!

- Left over assignments on recurrence realations
- Cormen 26.1-2, 26.1-3, 26.1-6, 26.1-7.
- Cormen 26.2-2, 26.2-3
- 26.2-10. Hint: think about extracting (s,t)-paths one by one and recue the flow in the edges as you go along.
- Cormen 26.2-13. Hint add a suitable value to each capacity
- Cormen 26.3-1, 26.3-5 (note that you can prove this from the integrality thm, as I did at the lecture).
- Suppose you are given a connected undirected graph $G = (V, E)$ with costs on the edges and your task is to give an algorithm which finds a minimum cost set of $E' \subseteq E$ edges whose removal disconnects the graph (that is the graph $G-E'$ is not connected). Explain how to do this in polynomial time (hint: use flows).

1 Notes on flows

Below are some notes on flows to give you a better understanding of some of the things I cover(ed) at the lectures.

1.1 Complexity of the Edmonds-Karp Algorithm

Recall that the Edmonds-Karp algorithm finds a maximum flow by always augmenting the current flow f along shortest (minimum number of arcs) (s,t)-paths in the residual network G_f. To prove that the running time of this algoritm is $O(|V||E|^2)$ we can argue as follows (below I use E to denote the edges (arcs) as in Cormen):

1. We can find the next augmenting path or determine that there is no (s,t)-path in the currect residual network G_f in time $O(|V|+|E|) = O(|E|)$ as we assume that G is connected. By the Max-Flow-Min-Cut theorem, if there is no (s,t)-path in G_f, then f is a maximum flow. It follows that we need to show that the total number of augmenting paths used in the algorithm is $O(|V||E|)$.

2. Let $P_1, P_2, \ldots, P_r$ denote the sequence of augmenting paths that the algorithm finds before termination. Also let $f_0 \equiv 0$ and let $f_1, f_2, \ldots, f_r$ be the flows after each augmentation. That is, f_{i+1} is obtained from f_i by augmenting by $\delta(P_i)$ units along P_i, where $\delta(P_i)$ denotes the minimum residual capacity of an arc on P_i.

   Claim 1: For all $i \in \{1, 2, \ldots, r-1\}$ we have $|E(P_i)| \leq |E(P_{i+1})|$.

   To prove this we use that the augmenting path P_i is found using breath first search (BFS) in $G_{f_{i-1}}$, for $i = 1, 2, \ldots, r$. Suppose that the distance from s to t in $G_{f_{i-1}}$ is k, then the BFS from s defines distance classes $L_0, L_1, \ldots, L_k$ from s where $L_o = \{s\}$ and $t \in L_k$. Let us call an arc from L_a to L_b **forward**,**flat** or **backwards** if, respectively $a = b-1$, $a = b$ or $a > b$. As P_i is a shortest path, every arc (u,v) on P_i is forward. Furthermore every (s,t)-path of length k in $G_{f_{i-1}}$ uses only forward arcs.

   Now consider which new arcs the new residual network G_{f_i} may contain. The only new arcs that can appear when going from $G_{f_{i-1}}$ to G_{f_i} are arcs that are opposite of those on P_i and, by the remark above, each of these correspond to arc which are backwards with respect to $L_0, L_1, \ldots, L_k$. Thus in G_{f_i} the distance from s to t is at least k as every path which uses at least one arc which is flat (backwards) with respect to $L_0, L_1, \ldots, L_k$ will have length at least $k+1$ $(k+2)$. This shows that the distance from s to t in G_{f_i} is at least k so $|E(P_i)| \leq |E(P_{i+1})|$, for $i = 1, \ldots, r-1$. □

   Claim 2: There are at most $|E|$ paths among $P_1, P_2, \ldots, P_r$ which have the same lenght.

This follows from the fact that every time we augment along a shortest path P_i at least one arc (u, v) which is forward wrt. the current distance classes $L_0, L_1, \ldots, L_k$ will not be present in the next residual network, namely those arcs which have residual capacity $\delta(P_i)$. So after at most $|E|$ augmentations there will be no forward arc wrt. $L_0, L_1, \ldots, L_k$ and hence no (s,t)-path of length k in the current residual network.

Now we can complete the proof of the complexity. The possible lengths of augmenting paths are $1, 2, \ldots, |V| - 1$ so Claim 2 implies that the total number of augmenting paths is at most $|V||E|$, implying that the Edmonds-Karp algorithm runs in time $O(|V||E|^2)$. □

1.2 The integrality theorem for flows

Recall that the integrality theorem for flows say that if we are given a flow network $G = (V, E)$, a capacity function $c : E \to \mathcal{Z}_0$ and distinct vertices s, t of V, then there exists a maximum flow f^* such that $f^*(u, v)$ is a non-negative integer for every arc $(u, v) \in E$. This follows easily from the way the Ford Fulkerson (or the Edmonds-Karp) algorithm works by induction over the number of augmenting paths we use before we reach a maximum flow.

To see that this simple theorem can be quite usefull let us prove the following claim (which you are asked to prove in a different way in Exercise 26.3-5). A graph $G = (V, E)$ is d-regular if every vertex $v \in V$ is incident to exactly d edges. A matching is **perfect** if it is incident to all vertices of the graph.

Theorem Every d-regular bipartite graph $G = (X, Y, E)$ has a perfect matching.

Proof:

As usual X, Y denote the two vertex sets of the bipartition. We can count the number of edges in E by summing the degrees of vertices in X or in Y, so $|X| = |Y|$ holds as $|E| = d|X| = d|Y|$.

Now, as in Section 26.3 of Cormen, consider the flow network N_G that we can build from G by

- Orienting every edge $xy \in E$ as the arc (x, y) from X to Y and give capacity 1 to these.
- Adding a new vertex s and an arc (s, x) of capacity 1 for each $x \in X$.
- Adding a new vertex t and an arc (y, t) of capacity 1 for each $y \in Y$.

Now let f be the flow that has value 1 one every arc incident to s or t and $\frac{1}{d}$ on every other arc. It is easy to check that this is an (s,t)-flow that respects all capacities. The value of f is $|X|$ $(= |Y|)$ so f is a maximum flow (the cut $S = \{s\}, T = V \setminus \{s\}$ shows this). By the integrality theorem there exist a flow f' such that $|f'| = |f|$ where $f'(u,v)$ is an integer for all arcs (u,v). Now by Corollary 26.11 in Cormen, we obtain a matching of size $|X|$ (and hence it is perfect) by setting $M = \{xy \in E | f'(x,y) = 1\}$. □

1.3 Flows with general balances

In this section we consider a flow in a network $N = (V, A, c)$ is a function f on the arcs so that $0 \leq f(u,v) \leq c(u,v)$ holds for all arcs $(u,v) \in A$.
Recall that the **balance vector** of a flow f is the function

$$b_f(v) = \sum_{w \in V} f(v,w) - \sum_{w \in V} f(w,v)$$

For every flow f we have $\sum_{v \in V} b_f(v) = 0$ as every arc (u,v) has two contributions to the sum, namely $+f(u,v)$ for the vertex u (the arc goes out of u) and $-f(u,v)$ for the vertex v (the arc goes into v).

Now let $b : V \to \mathbf{R}$ be an arbitrary function on the vertices of a digraph D so that $\sum_{v \in V} b(v) = 0$. We want to find out whether there exists a flow f in $N = (V, A, c)$ so that $b_f(v) = b_v$ for every $v \in V$. Such a flow is called **feasible** with respect to b, c. To see that a feasible flow may not exist consider the network consisting of only one arc (u,v) with $c(u,v) = 1$ and set $b(u) = 2$, $b(v) = -2$. A lot of important problems (including the problem of deciding whether a bipartite graph has a perfect matching) can be formulated as a feasible flow problem. So we need an algorithm that can find a feasible flow whenever one exists. We will use $N = (V, A, c, b)$ to denote a directed graph with a capacity function c on the arcs and a prescribed balance vector b. Given such a network we can form a new network $N' = (V \cup \{s,t\}, A', c', s, t)$ by adding vertices and arcs to N as follows:

- Partition the vertices of V into three sets V_+, V_-, V_0 where $V_+ = \{v | b(v) > 0\}$, $V_0 = \{v | b(v) = 0\}$ and $V_- = \{v | b(v) < 0\}$.
- Add two new vertices s, t
- Add an arc from s to every vertex $v \in V_+$ and give such an arc (s,v) capacity $b(v)$.
- Add an arc from every vertex $u \in V_-$ to t and give such an arc (u,t) capacity $-b(u)$.

Theorem The network $N = (V, A, c, b)$ has a feasible flow if and only the maximum (s,t)-flow in the network N' has value $\sum_{v \in V_+} b(v)$.

Proof: Suppose first that f is a feasible flow in N, so we have $b_f(v) = b(v)$ for every $v \in V$. Then we obtain an (s,t)-flow f' in N' by setting $f'(u,v) = f(u,v)$ for every arc $(u,v) \in A$, setting $f'(s,v) = b(v)$ for every arc (s,v) that we added above and setting $f'(u,t) = -b(u)$

for every arc (u, t) that we added above. It is easy to check that we have

$$b_{f'}(v) = \begin{cases} 0 & \text{if } v \notin \{s, t\} \\ \sum_{v \in V_+} b(v) & \text{if } v = s \\ \sum_{u \in V_-} b(v) = -\sum_{v \in V_+} b(v) & \text{if } v = t \end{cases} \tag{1}$$

Since $\sum_{v \in V} b(v) = 0$, this shows that f' is an (s, t)-flow of value $\sum_{v \in V_+} b(v)$.
To prove the converse direction it suffices to observe that if f' is a flow in N' which satisfies 1, then the flow f in N that we obtain by letting $f(u, v) = f'(u, v)$ for every arc $(u, v) \in A$ is a feasible flow in N. $\diamond$.

Clearly the cut $S = \{s\}, T = V \cup \{t\}$ is an (s, t)-cut in N' and its capacity is $\sum_{v \in V_+} b(v)$. Hence it follows from the result above that $N = (V, A, c, b)$ has a feasible flow if and only if the value of a maximum (s, t)-flow in N' is $\sum_{v \in V_+} b(v)$. So we can solve the problem of finding a feasible flow by constructing the new network N' and then running the Edmonds-Karp algorithm to find a maximum flow f^* in N'. If $|f^*| = \sum_{v \in V_+} b(v)$, then we obtain a feasible flow in N just by deleting s, t and all arcs incident to these and otherwise there is no feasible flow in N.

1.4 Orienting a graph to get a digraph with prescribed out-degrees

Let $G = (V, E)$ be an undirected graph. An **orientation** of G is any digraph $D = (V, A)$ that we can obtain by giving each edge $uv \in E$ one of the two possible orientations $u \rightarrow v$ or $v \rightarrow u$ (so A will contain the arc (u, v) in the first case and the arc (v, u) in the latter). The **out-degree**, $d^+_D(u)$, of a vertex u in a digraph $D = (V, A)$ is the number of arcs going out of u, that is, $|\{v|(u, v) \in A\}|$. It follows from this that $\sum_{u \in V} d^+_D(u) = |A|$ since every arc in A goes out of exactly one vertex.

Now let $G = (V, E)$ be an undirected graph and let $o : V \rightarrow \mathbf{Z}_0$ satisfy that $\sum_{u \in V} o(u) = |E|$. We would like to know whether we can orient the edges of G in such a way that we obtain a digraph $D = (V, A)$ with $d^+_D(u) = o(u)$ for all $u \in V$. We say that such an orientation of G is **good**.

Let us see how to formulate this problem as a feasible flow problem in some network N. First let $D = (V, A')$ be the digraph that we obtain from $G = (V, E)$ by first enumerating the vertices of V as $v_1, v_2, \ldots, v_n$, $n = |V|$ and then orienting each edge $v_i v_j \in E$ with $i < j$ as the arc (v_i, v_j). We call D' the **reference orientation** of G. Clearly every orientation of G can be obtained by changing the orientation of 0 or more of the arcs of D', so we are looking for a way to find such a set if it exists. Let us give each arc (v_i, v_j) of A' a capacity of one and then study integer flows (0 or 1 on every arc) in $N' = (V, A', c \equiv 1)$. Suppose we will interpret $f(v_i, v_j) = 1$ is indicating that we should replace the arc (v_i, v_j) by the arc (v_j, v_i). Then, using that we want the resulting out degree of each vertex v_i to be $o(v_i)$ we get that f must satisfy the following:

$$o(v_i) = d^+_{D'}(v_i) - \sum_{(v_i,v_j)\in A'} f(v_i, v_j) + \sum_{(v_k,v_i)\in A'} f(v_k, v_i) = d^+_{D'}(v_i) - b_f(v_i) \qquad (2)$$

From this we see that there exists a good orientation of G with respect to o if and only if there exists a feasible flow in the network $N'' = (V, A', c \equiv 1, b'')$, where $b''(v_i) = d^+_{D'}(v_i) - o(v_i)$. Thus, using the results on how to find feasible flows, we obtain a polynomial algorithm for checking whether a given graph G has a good orientation.

1.5 Finding the edge connectivity of a graph G using flows

Let $G = (V, E)$ be a graph and recall that the **edge-connectivity** of G, denoted $\lambda(G)$ is the minimum number of edges of G whose removal will leave a disconnected graph. So $\lambda(G) > 0$ precisely when G is connected. For any choice of distinct vertices x, y in $G = (V, E)$ we denote by $\lambda(x, y)$ the minimum number of edges whose removal disconnects x from y, that is, $\lambda(x, y)$ is the minimum number of edges across some partition $X, V \setminus X$ where $x \in X, y \in V \setminus X$. This implies that $\lambda(G) = \min\{\lambda(x, y)|x, y \in V\}$. Since every vertex v of G is either in V_1 or in V_2 for every choice of non-empty sets V_1, V_2 with $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$ it follows that if we fix our favorite vertex x, then we will have $\lambda(G) = \min\{\lambda(x, v)|v \in V \setminus \{x\}\}$, so we can determine $\lambda(G)$ by calculating the minimum of the $|V| - 1$ values $\lambda(x, v)$, $v \neq x$.

Let $G = (V, E)$ be given and construct a directed graph $D = \overset{\leftrightarrow}{G} = (V, A)$ by replacing every edge $uv \in E$ by a pair of anti-parallel arcs $(u, v), (v, u)$. From D we can then construct a network $N = (V, A, c \equiv 1)$ by giving each arc capacity 1. Let us fix a vertex $s \in V$ and observe that if V_1, V_2 is a partition of V with $s \in V_i$ and t is some vertex in V_{3-i}, then the capacity of the (s, t)-cut $(S = V_i, T = V_{3-i})$ in N is the same as the number of edges in G that go between V_1 and V_2. Thus the capacity of a minimum (s, t)-cut in N is exactly $\lambda(s, t)$. Thus it follows from the max-flow-min-cut theorem that we can find $\lambda(s, t)$ by finding a maximum (s, t)-flow f in N. Now the remark above implies that we can determine $\lambda(G)$ by $|V| - 1$ maxflow calculations where we keep the source vertex s fixed and let the sink t run through all possible vertices of $V \setminus \{s\}$. In Cormen they do not allow anti-parallel arcs, but they show how one can easily modify the network (by subdividing one arc of each such pair) without changing the problem.