1. Computability, Complexity and Algorithms

Bottleneck edges in a flow network:

Consider a flow network on a directed graph $G = (V, E)$ with capacities $c_e > 0$ for $e \in E$. An edge $e \in E$ is called a *bottleneck edge* if increasing the capacity c_e increases the size of the maximum flow.

Given a flow network $G = (V, E)$ and a maximum flow f^*, give an algorithm to identify *all* bottleneck edges. Do as fast in $O(\cdot)$ as possible. Justify correctness of your algorithm. You can assume basic operations (comparison, addition, subtraction, multiplication, and division) on two numbers take constant time.

Solution: Here is the general algorithm for finding all of the bottleneck edges in the flow network G.

We start with a maximum flow f^* for the flow network G. Consider an edge $\overrightarrow{vw}$ in the flow network G. Increasing the capacity of $\overrightarrow{vw}$ results in an increase in maximum flow value if and only if there exists a path from s to v and a path from w to t in G^{f^*}. This is because if there exists these two paths then more flow can be sent from s to v, then along the edge $\overrightarrow{vw}$, and finally from w to t.

Therefore, our algorithm for finding bottleneck edges is as follows:

1. Find a maximum flow f^* on G.
2. Run Explore/DFS from s in G^{f^*}. Let S be the set of vertices reachable from s in G^{f^*}.
3. Run Explore/DFS from t in the reverse graph of G^{f^*}. Let T be the set of vertices reachable from t in the reverse graph of G^{f^*}; note the set T are those vertices which can reach t in G^{f^*}.
4. For each $\overrightarrow{vw} \in E(G)$, output $\overrightarrow{vw}$ as a bottleneck edge if $v \in S$ and $w \in T$.

Since steps 2, 3, and 4 take $O(|V|+|E|)$ time, then since we are given a max flow f^* the running time is linear time.

Note that this algorithm looks for a path $s \to v$ and $w \to t$. What if these two paths share one or more edges? Then, the joined path will have one or more cycles. So, we can drop that cycle (or cycles) and get a shorter path from $s \to t$, but will this path still go through (v, w)? If one of the cycles contains edge $e = (v, w)$, then we have an augmenting path in G^{f^*} not using e, which would mean f^* is not a max flow. Hence, e cannot be in any of the cycles, so our algorithm works.

2. Analysis of Algorithms

All-pairs shortest paths (APSP) and Min-Sum Products. Suppose W is the adjacency matrix for G a simple undirected graph with no self-loops and no negative edge weights, and W^* is the reachability matrix ($w^*_{ij} = 1$ if there exists a path from i to j).

- Suppose operations are boolean (addition is OR, multiplication is AND). Suppose

$$W = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

Then show that

$$W^* = \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} (A \vee BD^*C)^* & EBD^* \\ D^*CE & D^* \vee GBD^* \end{bmatrix}$$

Observe that F, G use E in their definition, etc., so the calculations have to be done in the correct order. *Hint:* Consider G as partitioned into two subcomponents $V = V_1 \uplus V_2$.

- Now suppose W_{ij} is the weight of the edge (i, j). Moreover, now assume that matrix products are min-sum products (that is, addition is replaced by min and product by sum), and $A \vee B$ is the element-wise minimum of matrices A and B. If W^*_{ij} now denotes the shortest-path distance from i to j, show that W^* is computed by the same relation as in the previous part. You may be brief, 2-3 sentences suffices if your previous answer was thorough.

- Using this idea, show that

$$\text{APSP}(n) \leq 2\text{APSP}(n/2) + 6\text{MSP}(n/2) + O(n^2), \tag{1}$$

where APSP(n) denotes the worst-case running time of computing APSP on an n-vertex input graph, and MSP(n) denotes the worst-case running time of computing the min-sum product of two $n \times n$ matrices. Assume that arithmetic operations can be carried out in constant time.

In turn, show that $\text{APSP}(n) = \tilde{O}(\text{MSP}(n) + n^2)$. *Hint:* We know that MSP is superlinear, even superquadratic, in its runtime, simply since it needs to read its two input matrices.

Solution:

- $E = A \vee BD^*C$ can be read as "take a single step in V_1, or a step from V_1 to V_2, a walk through V_2, and then a step from V_2 back to V_1", which are all the ways to move from some vertex in V_1 to another using at most one edge in V_1 and at most 2 edges between V_1 and V_2. Taking the transitive closure of this gives all possible ways of moving between two vertices in V_1: take some number of steps in V_1, followed by a step to V_2 and a walk in V_2, followed by a step back to V_1, and so on.

For $F = EBD^*$, we note that this component of the matrix is asking about reachability from $v_1 \in V_1$ to $v_2 \in V_2$. Any such path starts in V_1, takes a reachability walk through V_1 (meaning it might go through V_2, but ends up back in V_1), and we argued that E contains this reachability. After a reachability walk through V_1, to make it to $v_2 \in V_2$, an edge from V_1 to V_2 must be taken, all of which are included in B. Then, once in V_2, one needs to move from the node in which the walk entered V_2 to v_2 using a walk through V_2, given to us by D^*. Note one need not go back to V_1, since the only reason to do so would be to get different reachability into V_2, but our initial reachability walk through V_1 means that path was already available.

For $G = D^*CE$, to walk from V_2 to V_1, one can take a walk through V_2 (using D^*), then take a single step into V_1 (using C), then take a reachability walk through V_1 (using E).

For $F = D^* \vee GBD^*$, to walk between two vertices in V_2, one can take a walk that stays within V_2 (D^*), or can take a reachability walk to V_1 (G), followed by an edge from V_1 to V_2, followed by a walk entirely within V_2. Because one takes a reachability walk to V_1, one need not revisit V_1 multiple times to change reachability.

- The "or" operation now takes the minimum, meaning that if each subcomponent of an "or" refers to a path of a given length available from i to j, the "or" refers to the smaller length. The previous argument said that all reachability paths are considered, so it remains to show that min-sum product computes the length of a particular path. So, taking the min-sum product of two vectors (one holding adjacency for i, the other for j) finds the two edges both adjacent to the same intermediate vertex which has minimum sum of weights and goes from i to j. Thus, a product finds the minimum-weight 1 or 2-hop path (since a 1 hop-path could be followed by a zero-cost self-loop). Thus, this product preserves reachability and keeps track of the cheapest current path between vertices.

- If we have W^*, we've computed all pairs shortest paths. The equivalence we showed in the previous two parts means it suffices to compute $(Y)^*$ for two submatrices, which we can choose to have size $n/2$, plus computing 6 min-sum products where the matrices are also of size $n/2$. The ors, of which there are a constant number, take $O(n^2)$ time to compute.

For the second part, by the recursion in the first part, we know that we will need $O(\log n)$ levels of recursive calls, and the amount of work at level i of the recursive calls, not including their recursive calls, will be $2^i \left(6 \cdot \mathrm{MSP}(n/2^i)\right) + O\left(2^i \cdot (n/2^i)^2\right)$.

Summing up over $i \in O(\log n)$, we have

$$\sum_{1<i<c\cdot\log n} 2^i \mathrm{MSP}(n/2^i) + (n^2/2^i) \le \mathrm{MSP}(n) \cdot c \cdot \log(n) + c \cdot n^2 \log n$$

since MSP is superlinear in n.

3. Theory of Linear Inequalities

Let $P = \{x \in \mathbb{R}^n \mid Ax \le b\} \subseteq [0,1]^n$ be a polytope with 0/1 vertices. It is well known that the diameter of any 0/1 polytope is at most n. Here we consider a stronger notion of diameter where the sequence of vertices has to be non-decreasing in value with respect to a given objective $c \in \mathbb{Z}^n$: For any two vertices $x, y \in P$ with $cy = \max_{z \in P} cz$ find the shortest path of *adjacent* vertices $x_1, \ldots, x_l$ with $x = x_1$ and $y = x_l$ so that $cx = cx_1 \leq \cdots \leq cx_l = cy$. The *monotone diameter* for an objective c is the maximum length over all such vertex pairs.

Prove that the monotone diameter is at most $O(n \log C)$, where $C = \max_i |c_i|$ (6 points). Can you also show that in this case the monotone diameter is at most n irrespective of the objective c? (4 points)

Solution. The first part follows from using geometric scaling with an augmentation oracle. The augmentation oracle allows that we move between adjacent vertices only. The geometric scaling algorithm generates a sequence of points (adjacent vertices) $x_1, \ldots, x_l$ with $cx \leq cx_1 \leq \cdots \leq cx_l \leq cy$ and moreover geometric scaling optimizes any linear integral objective over 0/1 polytopes with at most $n \log C$ augmentation calls.

For the second part observe that we can fix all coordinates where x and y coincide. Moreover, we can then flip coordinates, so that without loss of generality we can assume that $x = 0$ and $y = 1$. We can now show that the monotone diameter is at most n by induction. Let x^{t-1} be the current vertex. We claim that there exists an adjacent vertex x^t with $x_i^t = 1$ for some $i \in [n]$ so that $cx^t \geq cx^{t-1}$. Suppose not, then consider the cone C spanned by the directions $d_1, \ldots, d_k$ arising from moving to adjacent vertices and observe that $cd_j < 0$ for all $j \in [k]$. Then $P \subseteq x^{t-1} + C$ and in particular $y - x^{t-1} = \sum_j \alpha_j d_j$ for some $\alpha_j \geq 0$ for $j \in [k]$ and thus $cy = cx^{t-1} + \sum_j \alpha_j cd_j < cx^{t-1}$, which is a contradiction. Therefore such a vertex x^t with some coordinate $i \in [n]$, so that $x_i^t = 1$ exists. We can fix the coordinate i to 1 and recurse. This can happen at most n times before we reach the vertex y.

4. Combinatorial Optimization

Let $\mathcal{M} = (U, \mathcal{I})$ be a matroid and $w : U \to \mathbb{R}$ be a weight function.

1. Given any two bases B and B', show that there exists a sequence of bases $B_0, B_1, \ldots, B_k$ with the following properties.
   (a) $B_0 = B$ and $B_k = B'$.
   (b) $B_i \subseteq B \cup B'$ for each $0 \leq i \leq k$.
   (c) $|B_i \Delta B_{i+1}| = 2$ for each $0 \leq i \leq k-1$.
2. Suppose B' is a maximum weight basis under weight function w. Show that we can additionally ensure that $w(B_{i+1}) \geq w(B_i)$ for each $0 \leq i \leq k-1$.

Solution.

1. We construct the sequence inductively satisfying properties (b) and (c). Additionally, we ensure that $|B' \setminus B_{i+1}| < |B' \setminus B_i|$ which will ensure that the sequence ends with $B_k = B'$ for some integer k. We initialize with $i = 0$ and $B_i = B$. Consider any $i \geq 0$ such that $B_i \neq B'$. Let $x \in B_i \setminus B'$. From basis exchange property (see Theorem 39.6 in Schrijver), there exists $y \in B' \setminus B_i$ such that $B_i \cup \{y\} \setminus \{x\} \in \mathcal{I}$. Let $B_{i+1} = B_i \cup \{y\} \setminus \{x\}$.

2. We now show how to ensure that the exchange done to construct B_{i+1} in 1. always increases the weight. From the strong base exchange property (Corollary 39.12a), there exists a bijection $\pi : B_i \setminus B' \to B' \setminus B_i$ such that for all $x \in B_i \setminus B'$ we have $B_i \cup \{\pi(x)\} \setminus \{x\} \in \mathcal{I}$. Since, B' is the maximum weight basis, $w(B_i) \leq w(B')$. Thus there exists an $x \in B_i \setminus B'$ such that $w(x) \leq w(\pi(x))$. Defining $B_{i+1} = B_i \cup \{\pi(x)\} \setminus \{x\}$ gives us the desired sequence.

5. Graph Theory

Let G be a 2-connected graph and let $s \in V(G)$. Prove that G has two spanning trees T_1, T_2 such that for every vertex $v \in V(G)$ the two paths between v and s in T_1 and T_2 are internally disjoint.

Solution: Let t be a neighbor of s. We first show that the vertices of G can be numbered $v_1, v_2, \ldots, v_n$ in such a way that $v_1 = s$, $v_n = t$ and for all $i = 2, 3, \ldots, n$ the vertex v_i has a neighbor in $\{v_1, v_2, \ldots, v_{i-1}\}$ and the vertex v_{i-1} has a neighbor in $\{v_i, v_{i+1}, \ldots, v_n\}$. To that end we proceed by induction on the number of edges. If G is a cycle, then listing the vertices in the order of appearance on the cycle, starting from s and ending in t, is as desired. Thus we may assume that G is not a cycle, and hence by the ear-decomposition theorem it is of the form $G = H \cup P$, where H is a 2-connected proper subgraph of G containing s and t, and P is a path with both ends in H and otherwise disjoint from H. By the induction hypothesis the vertices of H have a required numbering $u_1, u_2, \ldots, u_k$. Let u_i, u_j be the ends of P, where $i < j$, and let $u_i, w_1, w_2, \ldots, w_l, u_j$ be the vertices of P in order. Then $u_1, u_2, \ldots, u_i, w_1, w_2, \ldots, w_l, u_{i+1}, u_{i+2}, \ldots, u_k$ is a desired ordering of the vertices of G.

Now given the order of the vertices as in the previous paragraph we select, for every $i = 2, 3, \ldots, n$, a neighbor $f(v_i)$ of v_i in $\{v_1, v_2, \ldots, v_{i-1}\}$ and a neighbor $g(v_{i-1})$ of v_{i-1} in $\{v_i, v_{i+1}, \ldots, v_n\}$. We now define T_1 to consist of all edges with ends v and $f(v)$ for all $v \in V(G) - \{s\}$ and we define T_2 to consist of the edge st and all edges with one end v and the other end $g(v)$ for all $v \in V(G) - \{s, t\}$. Then T_1 and T_2 are as desired.

6. Probabilistic methods

Suppose that we throw m balls into n bins independently and uniformly at random (initially all bins are empty, of course).

(A) Prove that $m^*(n) = n \log n$ is a threshold function for the property 'there exists an empty bin', i.e.,

$$\Pr(\text{there exists an empty bin}) \to \begin{cases} 1 & m \ll n \log n, \\ 0 & m \gg n \log n. \end{cases}$$

(B) Make an educated guess what the threshold function for the property 'there exists a bin with at most one ball' is. Prove the corresponding 0-statement (no proof of the corresponding 1-statement expected).

Hint: *Recall that* $1 - x = e^{-x+O(x^2)}$ *as* $x \to 0$.

Solution: For (A), let X denote the number of empty bins. Writing X_i for the indicator variable for the event that the ith bin is empty, we have $X = \sum_{i \in [n]} X_i$ and thus

$$\mathbb{E}X = \sum_{i \in [n]} \mathbb{E}X_i = n \left(1 - \frac{1}{n}\right)^m.$$

Using $1 - x \le e^{-x}$ it is easy to see that $\mathbb{E}X \to 0$ for $m \gg n \log n$, which proves the 0-statement of (A) [using Markov's inequality or the first moment method].
Turning to the 1-statement, note that for $m \ll n \log n$ the hint $1 - x = e^{-x+O(x^2)}$ gives

$$\mathbb{E}X = ne^{-m/n+o(1)} \to \infty.$$

Furthermore, standard second-moment calculations and the hint similarly give

$$\mathbb{E}X^2 = \sum_{i \in [n]} \mathbb{E}X_i + \sum_{i,j \in [n]: i \neq j} \mathbb{E}X_i X_j = \mathbb{E}X + n(n-1)\left(1 - \frac{2}{n}\right)^m \le \mathbb{E}X + (\mathbb{E}X)^2 \cdot e^{o(1)}.$$

Since $\mathbb{E}X \to \infty$ implies $\mathbb{E}X = o((\mathbb{E}X)^2)$, using $e^{o(1)} = 1 + o(1)$ we infer $\mathbb{E}X^2 \le (1 + o(1))(\mathbb{E}X)^2$, so that

$$\text{var}X = \mathbb{E}X^2 - (\mathbb{E}X)^2 = o((\mathbb{E}X)^2),$$

which implies the 1-statement of (A) [using Chebychev's inequality or the second moment method]

For (B), let Y denote the number of bins with at most one ball. Writing Y_i for the indicator variable for the event that the ith bin contains at most one ball, we have $Y = \sum_{i \in [n]} Y_i$ and thus

$$\mathbb{E}Y = \sum_{i \in [n]} \mathbb{E}Y_i.$$

Distinguishing the cases of one or zero balls in the ith bin, we see that

$$\mathbb{E}Y_i = \left(1 - \frac{1}{n}\right)^m + m \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{m-1} = \left(1 - \frac{1}{n} + \frac{m}{n}\right) \cdot \left(1 - \frac{1}{n}\right)^{m-1}.$$

Hence we obtain

$$\mathbb{E}Y = (n - 1 + m) \cdot \left(1 - \frac{1}{n}\right)^{m-1}.$$

Trying out some possible functions $m = m(n)$, using $1 - x \le e^{-x}$ and the hint it is straightforward to see that $\mathbb{E}Y \to 0$ if $m \gg n \log n$, and $\mathbb{E}Y \to \infty$ if $m \ll n \log n$. This proves the 0-statement, and justifies the educated guess that $m^*(n) = n \log n$ is again the threshold function [as can be verified by calculating the variance/second moment, but this calculation was not expected due to time-constraints], completing (B).

7. Algebra

Suppose p and q are odd primes and $p < q$. Let G be a finite group of order p^3q. Prove that G has a normal Sylow subgroup.

Solution: The number n_q of q-Sylows divides p^3, whence $n_q = 1, p, p^2, p^3$. If $n_q = 1$, then G has a normal q-Sylow. n_q is congruent to 1 mod q, whence $n_q \neq p$ because $p < q$. If $n_q = p^3$, then there are $p^3(q-1)$ distinct elements of order q. This leaves p^3 elements of G which are not of order q. Thus $n_p = 1$, and G has a normal p-Sylow subgroup. So, we may assume that $n_q = p^2$. Therefore $n_q - 1 = (p+1)(p-1)$ is divisible by q. Since $p < q$, q does not divide $p-1$. Therefore q divides $p+1$. It follows that $q = p+1$. This contradicts that q and p are odd primes.