# 1 (More) Linear Programming

## 1.1 (More) Shortest Paths

**Exercise.** *Suppose we have a graph $G = (V, E)$, and want to find the shortest path between some two vertices $S$ and $T$. Write a linear programming formulation of this problem (you may assume that there are no negative cycles).*

# 2 Max flow / min cut

1. **Maximum flow is equal to minimum cut**:
   (a) Maximum flow $\leq$ minimum cut is clear.
   (b) Maximum flow $\geq$ minimum cut comes from the augmenting paths algorithm: when there are no more augmenting paths to be found, all the vertices reachable from $S$ in the residual network form a cut. All edges crossing this cut must have zero residual capacity, which means the flow is at least the sum of their original capacities, which in turn is at least the flow of the minimum cut.

   In particular, if $e$ is an edge in the original graph with capacity $c(e) = c$ (and with $\bar{e}$ not in the graph), and we ship flow $f \leq c$ across $e$, then $e$ has residual capacity $c - f$. The reverse edge $\bar{e}$ then has residual capacity $0 - (-f) = f$.
2. **Ford-Fulkerson algorithm** To find the maximum flow of a graph with integer capacities, we repeatedly use DFS to find an **augmenting path** from start to end node in the residual network (note that you can go backwards across edges with negative residual capacity), and then add that path to the flows we have found. We stop when DFS is no longer able to find such a path.
   (a) **Residual flow**: To form the residual network, we consider all edges $e = (u, v)$ in the graph as well as the reverse edges $\bar{e} = (v, u)$.
      i. Any edge that is not part of the original graph is given capacity 0.

ii. If $f(e)$ denotes the flow going across $e$, we set $f(\bar{e}) = -f(e)$.

iii. The **residual capacity** of an edge is $c(e) - f(e)$.

(b) Correctness:

This algorithm might be a little confusing. Some clarifying points:

i. We think of the residual network as of a networking without any flow whose capacities can however change during the course of the algorithm

ii. after each new augmenting path has been added, we have a flow on our original network, and we change the residual capacities accordingly.

Here's the argument for maximum flow $\geq$ minimum cut, which also establishes correctness. The set of reachable nodes after the algorithm terminates defines a cut of the graph. In particular, it must be true that min cut $\leq$ cut found = flow found $\leq$ max flow, hence min cut = max flow, and in fact all inequalities are equalities, and the flow found is a maximum flow.

I find the following part to be the trickiest in this argument: why is the flow found equal to the cut found? The reason we can make this argument is because we have the residual network. Suppose we have a source $s$ and a target $t$. Consider the cut determined by the algorithm: let the nodes reachable from $s$ form the set $S$, and the nodes unreachable from $s$ form the set $T$. We know that all edges crossing from $S$ to $T$ are at full capacity: otherwise we would've been able to reach something in $T$ from $S$. But we also know that there is no flow coming from $T$ to $S$, because then the residual edge going in the opposite direction would be able to carry some flow, again contradicting the un-reachability of $T$. So, the total flow flowing into $T$ is equal to the size of the cut between $S$ and $T$; and on the other hand, by conservation of flow, it is equal to the total flow coming into $T$.

(c) The runtime is $O((m+n)(\text{max flow}))$.

(d) **Non-integer variant** If we use BFS instead of DFS above, this method also works for non-integral capacities and runs in time $O(nm^2)$.

3. **Matchings**: The maximum size matching in a bipartite graph can be solved by taking the integer maximum flow from one side to the other.

4. **Karger's algorithm** (We didn't cover this in class, but it's pretty cool.) To try to find a minimum cut on a graph where all edges have weight 1, randomly contract edges until there are only two vertices left, and then output the weight of the cut which separates the two vertices.

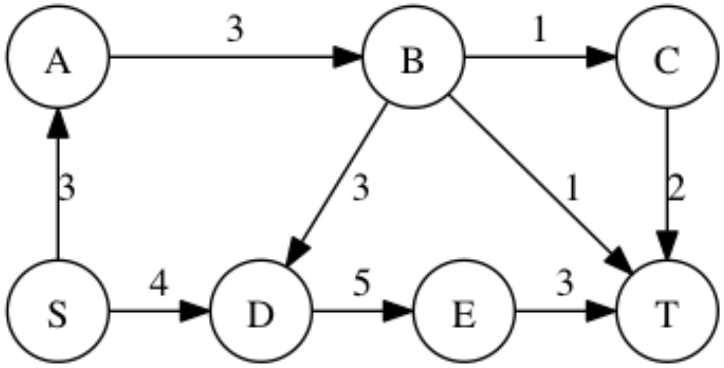
(a) To *contract* an edge $(u, v)$ means replacing $(u, v)$ by a new node $w$, and then replacing $u$ or $v$ with $w$ in all edges of the graph where they occur.

(b) With probability at least $1/\binom{n}{2}$, Karger's algorithm will output a particular minimum cut.

**Exercise.** *Perform the augmenting paths algorithm on this example, showing the residual graph at each step:*

## 2.1 Power Grid

**Exercise.** *Suppose you are the owner of a power plant with some customers, you have a power grid in place where each power line (edge) has some maximum energy it can carry, and each customer has a*




*maximum amount of energy they require. What is the most energy that you can dispense with the current configuration so that no customer gets more than they request?*

**Exercise.** *Now, suppose that each of the transformers along the way (e.g. each vertex of the graph) also has a maximum flow which can pass through it. What is the most energy that you can dispense with the current configuration so that no customer gets more than they request?*

**Exercise.** *Upon revisiting your power allocation algorithm, you realize that customers will probably be happier if they receive excess energy than if they receive insufficient energy. Given the same configuration as above, but this time where each customer has a minimum amount of energy that s/he requires, how could you determine whether it is possible to meet all the customer needs?*

## 2.2 Class Planning

**Exercise.** *Suppose you have N students and M classes. Each student signs up to take 4 classes, and each class has some maximum total enrollment. We want to assign each student to some subset of the classes they have chosen so that we maximize the total enrollment across all of the classes, and no class exceeds its maximum enrollment. How should we compute such a configuration?*

## 2.3 Minimum Edge Flows

**Exercise.** *Suppose that each edge, in addition to having a maximum capacity, also has a minimum capacity. Reduce the problem of determining whether such a flow exists to a max flow problem.*

Now, use this to solve the following problem from Salil's quals:

**Exercise.** *Suppose we have some m by n matrix of real numbers, such that the sum of every column and every row is an integer. Prove that there exists a way to round every entry up or down, such that all the row and column sums are preserved.*