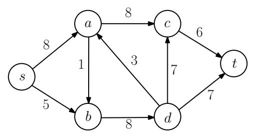
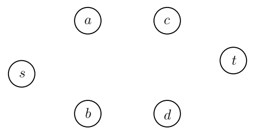
15-451/651 Algorithms, Fall 2017
Recitation #8 Worksheet

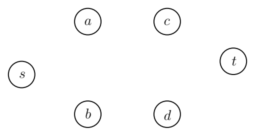
Network Flow

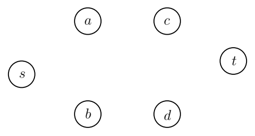
Example of running Ford-Fulkerson: Here is a problem from the midterm in 2013. Consider the graph below.



Run Ford-Fulkerson and show the final residual graph. Also what is the maximum flow?







Solution: Max flow is 12.

Network flow for image processing. Here is an application of network flow to 3-D image processing. To get a 3-D image, you take a stereo camera that produces two pictures, and match the pictures together to produce a single image, where each pixel is labeled with its depth in the image. The problem is that this process can produce noise because of mistakes in matching things up. To fix those mistakes, one approach is to use the fact that in general, most objects have smooth boundaries, which means that most pairs of neighboring pixels should be at the same depth.

We can formalize the problem like this: Given a pixel image I, where each pixel is 0 or 1 (indicating close or far — in actual applications, you have many depth levels, but we will just consider 2 levels, so I is a 0/1 matrix), we want to find the best modification I' of I using the following criteria: we pay \$$a$ for each bit of I whose depth level we flip, and pay \$$b$ for each pair of neighboring pixels in I' at different depths. For instance, if we decide to not change I at all ($I' = I$) then we pay b times the number of neighboring pixels in I that are at different levels. If we decide to just make all depths equal to 0, we pay a times the number of ones in I.

Solve for the best I' by setting this up as a min-cut problem, and using a max-flow algorithm.

Solution: Here's how we do it: create a vertex for each pixel, and also a source and sink. Connect source to all pixels that are initially 0s, and sink to all 1's by edges of capacity a. Connect neighboring pixels by bidirectional edges of capacity b.

Running our favorite max-flow/min-cut algorithm will give us a cut, which can be thought of as a partition of all the pixels. The pixels that are now in the same partition as the source are going to be 0s, and the pixels that are in the same partition as the destination are going to be 1s.

Now, we want to show that the cost of any such cut is the same as the cost of the flips and edges. If that is the case, then the min-cut must be the optimial solution!

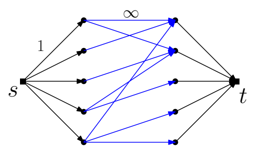
Note that if a pixel was initally a 0, and ends up in the partition with the sink (so is a 1), then the edge from the source to it must have been cut, and so it cost a to flip it. The same is true for flipping a 1 to a 0.

In addition, any neighboring pixels which are in a different partition (and therefore don't match) must have had the edge between them cut, costing b. No other edges would have been cut.

So, we can see that it costs a for every element that was flipped, and b for every pair that was not equal. Therefore, a cut costs the amount specified in the problem, and since we have found the min-cut, we have found the solution with the lowest cost.

Hall's Marriage Theorem. Given a bipartite graph $G = (L, R, E)$, we found a maximum matching by finding a s-t maximum (integer) flow. Here is a slightly different reduction:

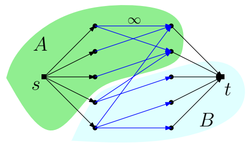
(a) Add a new source s and target t. Add *unit capacity* directed edges from s to vertices in L, and from vertices in R to t. Direct edges in E towards R, make them *infinite capacity*.



Show a correspondence between integral s-t flows in this flow network, and matchings in G. (Hence the maximum matching in G equals the minimum s-t cut in this network.)

(b) For any set $S \subseteq L$, let $N(S) = \{v \in R \mid \exists u \in S, (u, v) \in E\}$ be the "neighbors" of S. Hall's Marriage theorem says: the size of the maximum matching in G equals $|L|$ *if and only if* for each subset $S \subseteq L$, $|N(S)| \geq |S|$. Deduce Hall's theorem from part (a), and the max-flow min-cut theorem.

Solution: Suppose the max-matching has size $M < |L|$. By part (a), so has the s-t min-cut in G. Look at this minimum cut (A, B) with $s \in A, t \in B$; the total capacity of arcs from A to B is M. So none of the infinite capacity edges go from A to B. All the arcs from A to B are either s-to-L edges, or R-to-t edges, each of unit capacity. (Infinite capacity arcs could go from B to A, of course.)



look at $S := A \cap L$, and $T = A \cap R$. We have that $M = |L \setminus A| + |A \cap R| = (|L| - |S|) + |T|$. (Why? Each unit of capacity cut by (A, B) corresponds to a node in $L \setminus A$, or in $L \cap R$, as in the figure above.)

Also $M < |L|$, so $|T| < |S|$. Moreover, all arcs leaving S have infinite capacity, so they must stay within A. This means $N(S) \subseteq (A \cap R) = T$. So $|N(S)| \leq |T| < |S|$. Conversely, if there exists a set S with $|S| > |N(S)|$, an identical argument shows a min s-t cut less than $|L|$. This proves Hall's theorem.