We left off last time by showing how bipartite matching can be reduced to maximum flow. We will now consider an extension of the bipartite matching problem called the *survey design* problem.

Suppose a company has kept logs of what products each customer has bought, and they are interested in sending surveys to the customers to ask them questions regarding their products. Each customer should receive at most one question per product they bought.

We don't want to ask any one customer too many questions, so let c_i denote the maximum number of questions we ask customer i. We also don't want to ask too many questions regarding the same product, so let p_j denote the maximum number of questions we ask about product j.

Subject to these constraints, we would like to ask as many questions as possible to get the most information we can regarding the products.

Again we want to construct a flow network based off of our input.

Notice that the graph relating customers and products is again a bipartite graph. The problem is similar in flavor to the bipartite matching problem, however this time a customer may be asked about multiple products.

How can we modify the construction for bipartite matching to handle the survey design problem?

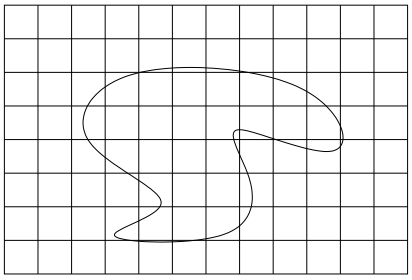
Last time we explained the equivalence between computing a maximum flow and a minimum cut in a flow network, and showed how the bipartite matching problem can be reduced to maximum flow.

For some applications which can be reduced to maximum flow, it is more intuitive to think of the problem as a cut problem rather than a flow problem.

One such example is the **image segmentation** problem in which we are given an image as input and we want to develop an algorithm which can identify an object in the image.

Intuitively, we want the algorithm to "cut" an object out of the image. We will show that we can reduce the image segmentation to the min cut problem, which can be solved using the Ford-Fulkerson method.

In the image segmentation problem, we are given a "grid graph" G where the vertices correspond to pixels in an image and the edges correspond to "neighboring" pixels.



For each pixel i, we have a likelihood a_i that it belongs in the object and a likelihood b_i that it belongs in the background. Both are arbitrary nonnegative numbers. We hope to assign the likelihoods such that $a_i > b_i$ if the pixel is more likely to be in the object than the background, and $b_i > a_i$ if it is more likely to be in the background.

In practice, assigning these likelihoods is a difficult problem in and of itself, and simply taking all of the pixels i for which $a_i > b_i$ could lead to a solution that visually looks quite poor.

Intuitively, if many of a pixel i's neighbors are being assigned to the background, then i probably should too.

To try to influence this behavior, for each pair of neighbors (i, j) we assign a separation penalty $p_{ij} \geq 0$ for placing one of i or j in the object and the other in the background.

Given the likelihoods and separation penalties, the image segmentation problem is to compute a partition of the vertices in G into two sets A (the object) and B (the background) so as to maximize the following objective function:

$$q(A, B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, |A \cap \{i,j\}| = 1} p_{ij}$$

We will show that we can solve this problem via a reduction to min cut.

One main issue we must overcome is that the min cut problem is a minimization problem, but the image segmentation problem is a maximization problem.

Let $Q = \sum_i (a_i + b_i)$. Note that $\sum_{i \in A} a_i + \sum_{j \in B} b_j$ is the same as $Q - \sum_{i \in A} b_i - \sum_{j \in B} a_j$.

Therefore we can write our objective function $q(A, B)$ as so:

$$q(A, B) = Q - \sum_{i \in A} b_i - \sum_{j \in B} a_j - \sum_{(i,j) \in E, |A \cap \{i,j\}| = 1} p_{ij}$$

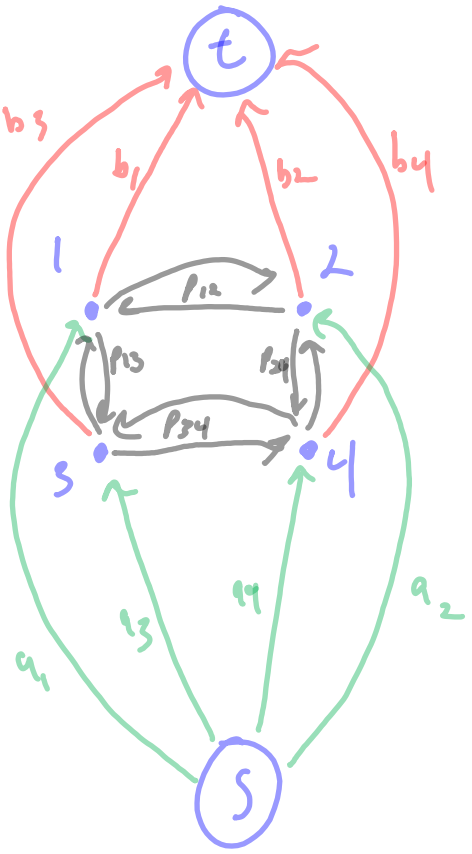
Note Q is a constant (independent of the choice of A and B), so in order to maximize $q(A, B)$ we need to minimize the following:

$$q'(A, B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in E, |A \cap \{i,j\}| = 1} p_{ij}$$

Now we have a minimization problem that can fit into the framework of the minimum cut problem.

Idea: Create a flow network G' such that any object/background partition A and B corresponds to an $s - t$ cut in G' such that the capacity of the cut is equal to $q'(A, B)$ (and vice versa). If we can do this, then we can find the optimal A and B using the Ford-Fulkerson method.

Example with 4 vertices:



S-T cut:

view S as being object pixels and T as background pixels