Precept Topics
- Dynamic Programming
- Maxflow/Mincut and Ford-Fulkerson

Relevant Material
- Book chapter: 6

A. RECAP: Dynamic Programming and Maxflow/Mincut

Your preceptor will give an overview of the content of this week's lectures.

Feel free to use this space for notes or as scratch paper.

B. EXERCISE: Seam Carving

In the next assignment, you will be required to find (horizontal and) *vertical seams* of images. A $W \times H$ -pixel image defines a $W \times H$ *energy matrix* (which we assume is given; you'll be given a formula to compute energies in your assignment).

A *vertical seam* is a path of pixels connected from the top row to the bottom row , where a pixel at column x and row y is connected to bottom left/bottom/bottom right pixels (i.e., at positions $(x-1, y+1)$, $(x, y+1)$ and $(x+1, y+1)$), if they exist. *Note that these row/column indices, which are standard in image processing, are the reverse of the standard in math.*

The *seam energy* is the sum of the energies of the pixels in the seam, and a *minimum energy vertical seam* is, well, the vertical seam with the minimum energy.

(15,10,16)	(31,15,19)	(15,10,3)
(5,18,0)	(80,18,0)	(120,100,80)
(35,20,12)	(36,17,13)	(15,10,3)
(5,1,13)	(13,1,16)	(120,110,40)

RGB Values of the 3x4 Image

32	72	45
123	163	75
32	75	41
156	161	9

Energy Values

a) Consider the 3x4 image and corresponding energy values above. Find the minimum energy vertical seam and compute its energy.

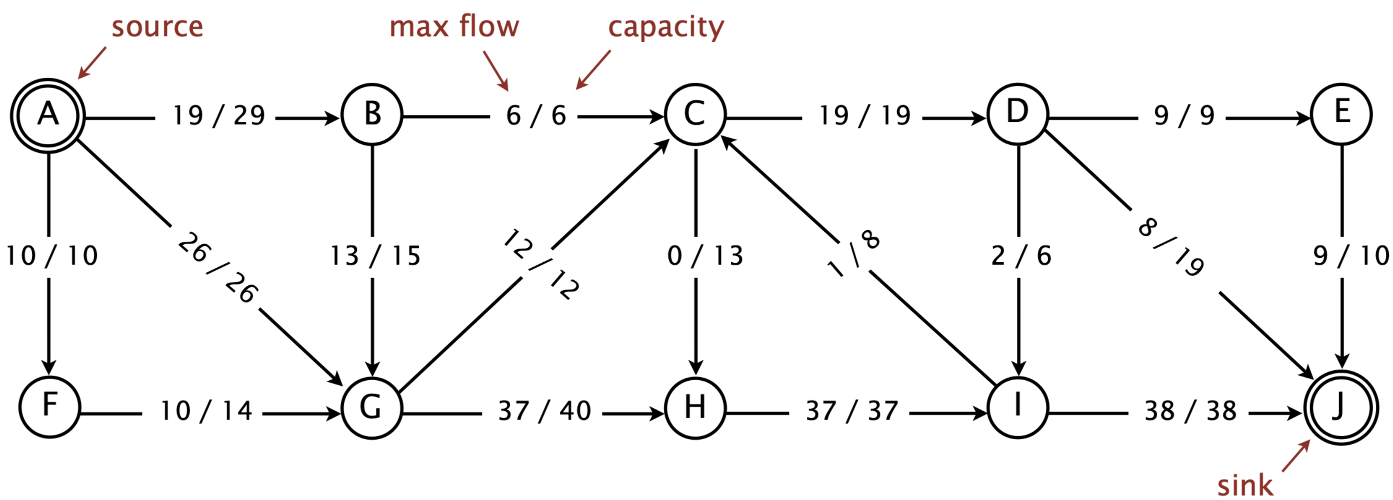
b) In order to find the minimum energy vertical seam, you will have to find the shortest path from any pixel in the top row to any pixel in the bottom row.

Draw the implicit graph represented by the energies matrix. (That is, draw the graph whose paths correspond to seams and whose lengths correspond to the energies of the seams.) Show all the edges and edge weights.

c) What is the order of growth of the algorithm that finds a minimum energy vertical seam on the graph represented by a $W \times H$ image (as a function of W and H), if it uses Dijkstra to find shortest paths? What if it uses Bellman-Ford instead? (Assume edges are relaxed from the bottom to the top.) Explain how you can compute vertical seams faster than with either shortest-path algorithm above.

C. EXERCISE: Maximum Flow

Consider the following flow network with a maximum flow f^*.



a) What is the value of f^*? Compute the *capacity* of the cut $\{A, B, C\}$ and the net *flow* across this cut.

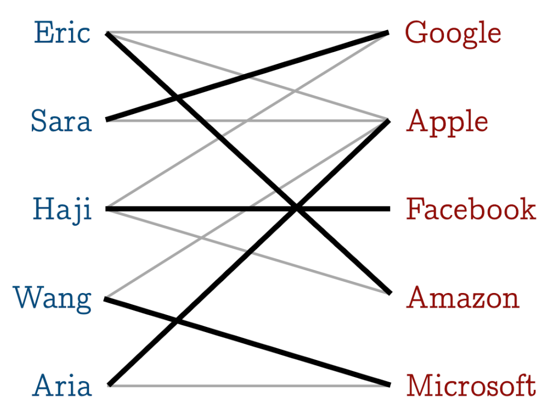
b) Find a minimum cut in the network. Which vertices are on the source side of the cut?

c) What is the capacity of the minimum cut? What is the net flow across the minimum cut?

D. EXERCISE: Bipartite Matching

The graph shown next has a set X of job applicants on the left and the set Y of companies with job openings on the right. An edge (x_i, y_j) is present if company y_j sent applicant x_i a job offer. Design an algorithm for finding which job offers should be accepted so that every company hires one applicant and every applicant is hired by one company.

As shown in the example below, this problem can be modeled as a *bipartite graph*, where edges represent job offers. Accepted offers are marked in bold.

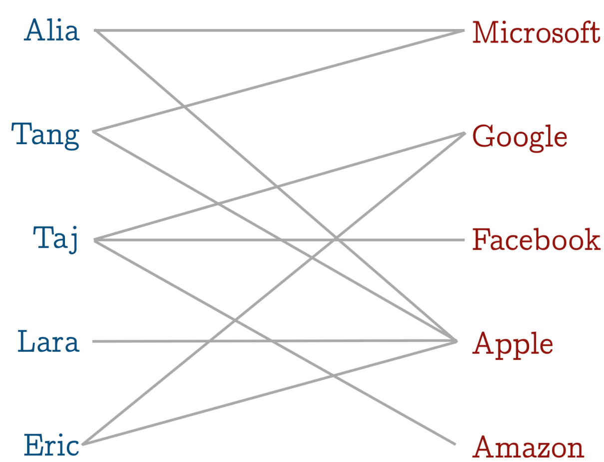


Perfect Matching

a) Model the above as a **max-flow** problem by converting the bipartite graph into a flow network. Show your work by drawing the flow network corresponding to the "Perfect Matching" example above.

Make sure to mark the `source` and `sink` vertices and the edge `capacities`, and describe how a max-flow in the network corresponds to a matching in the bipartite graph.

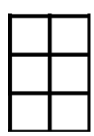
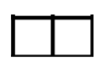
b) Use the Ford-Fulkerson algorithm to find the maximum matching for the following (different) instance of the problem, and draw the max-flow it finds.



c) What is the order of growth of the running time of the algorithm as a function of the number of edges E and the number of vertices V in the bipartite graph?

d) Find a **minimum cut** in the max-flow network you have computed in part b).

<u>**Optional**</u> **problem:** What is the number of distinct ways an $n \times 2$ grid can be tiled by 1×2 and 2×1 tiles? (As a warm-up, consider the $n = 3$ problem below.)

Grid ($n = 3$):  Tiles:  and