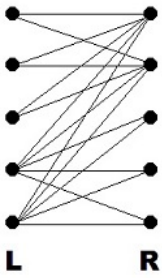
CSCI 270 Lecture 21: Alternate flow formulations

Bipartite Matching

We are given an undirected bipartite graph $G = (L \cup R, E)$.



$M \subseteq E$ is a matching if each node appears at most once in M. Find the max-size matching.

- What is the largest-sized matching you can find?
- Why can't there be a larger matching?

A perfect matching includes every node. This instance cannot have a perfect matching.

We could try to come up with our own algorithm to solve this problem, but that sounds like too much work. Instead we're going to be **lazy**, and use an existing algorithm. Namely, we'll use Ford-Fulkerson.

To use Ford-Fulkerson, we have to transform the current graph into a network flow graph. In addition, it must be done in such a way that the solution to the network flow graph helps us find the max-sized matching.

- What components are missing in this graph, which are needed in a Network Flow graph?
- How should we transform our graph into a Network Flow graph?
- How do we extract the answer to Bipartite Matching once we've solved Network Flow?

So, by reducing Bipartite Matching to Network Flow in polynomial time, we have proven that Bipartite Matching is also solvable in polynomial time!

Now that we've reduced Bipartite Matching to Network Flow, Bipartite Matching is a valid problem to reduce to as well. You could reduce some new problem $C \leq_p$ Bipartite Matching to show C is poly-time solvable.

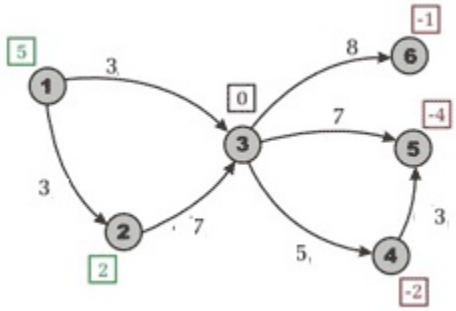
Circulations

We are given a directed graph G, with edge capacities $c(e)$, for all $c \in E$.

Each node has an integer **demand** $d(v)$. If $d(v) < 0$, then we say that this node has a **supply**.

A circulation is a function f which satisfies:

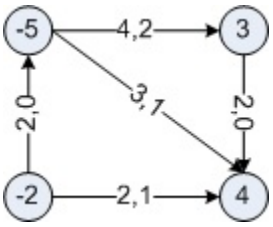
1. $0 \leq f(e) \leq c(e)$, for all e
2. $\sum_{(x,v)} f(x,v) - \sum_{(v,y)} f(v,y) = d(v)$



- Is there a circulation that satisfies these constraints?
- What components are missing in this graph, which would be needed in Network Flow?
- What components are in this graph, which must be removed in Network Flow?
- How should we transform our graph into a Network Flow graph?
- How can we extract the answer for Circulations, once we've solved Network Flow?

Circulations with Lower Bounds

Suppose edges have lower bounds $l(e)$. The new capacity rule is: $l(e) \leq f(e) \leq c(e)$



- Is there a circulation that satisfies these constraints?
- What problem should we reduce to? (Hint: NOT Network Flow!)

- What components are in this graph, which must be removed?
- How should we transform our graph?
- How can we extract the answer for Circulations with Lower Bounds?

You may reduce to Network Flow, Bipartite Matching, or Circulations with or without lower bounds. You may reduce to other problems in P as well, but this is a unit on Network Flow, so it will be most fruitful to focus on those problems.

Important: don't mix up your problems. For example: Network-Flow with lower bounds is NOT a problem we have shown to be in P. You must do Circulations with lower bounds, or Network Flow without lower bounds.

Survey Design

There are n customers $\{c_1, ..., c_n\}$ and m products $P = \{p_1, ..., p_m\}$. Customer c_i owns a subset of the products $S_i \subseteq P$.

We want to design a survey which gets feedback on all of our products: we must ask at least qp_i questions (to get meaningful feedback) and no more than qp_i' questions (because the questions would become redundant) about product p_i, and we can only ask questions to customers which own p_i.

Lastly, we must ask at least cp_i questions to customer i (to avoid wasting their time) and no more than cp_i' questions (to keep the survey a reasonable length. Give a polynomial-time algorithm to design such a survey or find that no such survey exists.

- What problem should we reduce to?
- How should we transform survey design?
- Are there any problems with the transformation?
- How can we fix this problem?

Baseball Elimination

Team	Wins	LA	Col	SF	Ari	SD
Los Angeles	90	0	0	0	7	4
Colorado	88	0	0	0	0	1
San Francisco	87	0	0	0	0	4
Arizona	86	7	0	0	0	4
San Diego	75	4	1	4	4	0

You want to determine if a team can end up with the most wins at the end of the season (or tied for most wins, forcing a playoff).

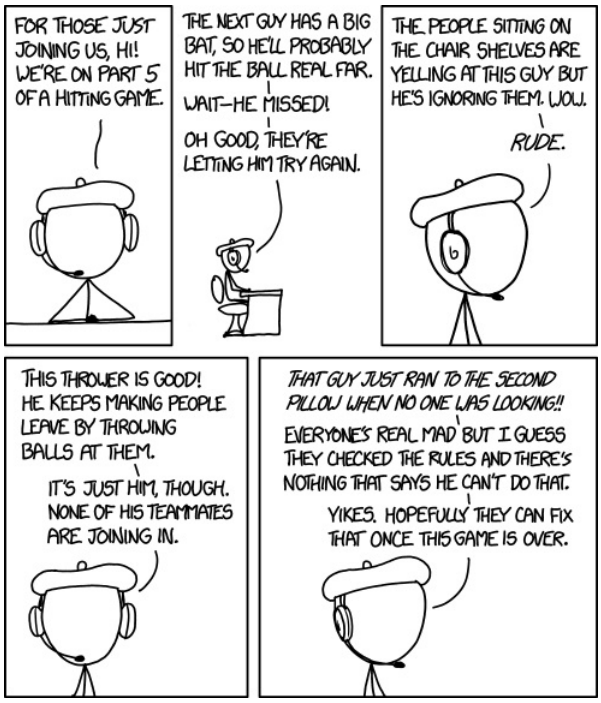


Figure 1: XKCD #1593 . The thrower started hitting the bats too much, so the king of the game told him to leave and brought out another thrower from thrower jail.

- Which teams are clearly eliminated?
- There is another team that is also mathematically eliminated: which one?

We want to design a poly-time algorithm that determines whether a specific team has been mathematically eliminated. For this example, let's consider San Francisco. It's not clear off-hand which problem to reduce to, but that's not an issue, because they're all graph problems: we can start by creating a graph and then decide which problem we're reducing to later. The flow will probably indicate who wins which games.

- What nodes should we add to this graph?
- Should we include San Francisco in our graph, or do we already know how to allocate their games?

We need to make sure that only teams involved in a matchup can win that specific game. We'll therefore add a node for each matchup.

- How should we finish our transformation?
- What problem are we reducing to?
- How do we extract the answer?