CS 163 & CS 265: Graph Algorithms

Week 8: Flow

Lecture 8a: Definition, properties, and applications

David Eppstein
University of California, Irvine

Winter Quarter, 2024

Baseball elimination

Background

Pro baseball teams are grouped into divisions

- Currently six divisions, five teams each
- Both numbers have varied over time

All teams play same the same total number of games per season, both against teams in their division and others

Baseball games always have a winner and a loser (no ties)

At the end of the season, the team from each division with the most wins goes on to the post-season series of championships leading to the World Series (as do two "wildcard" teams which are not important for this application)

If there is a tie for first place, the tied teams play another tie-breaker game

Elimination

Close to the end of the season, but with a small number of games remaining to play...

Team X still has a chance if some scenario for who wins each remaining game in the division (no matter how unlikely) leaves Team X with the most wins or tied for the most

Otherwise, Team X is mathematically eliminated

Example:

- Suppose remaining games are X vs W, and Y vs Z.
- Suppose also that Y and Z each have one more win than X.
- X can tie with the loser of the Y–Z game by winning the game against W, but will still be eliminated by the winner of the Y–Z game, so X is mathematically eliminated

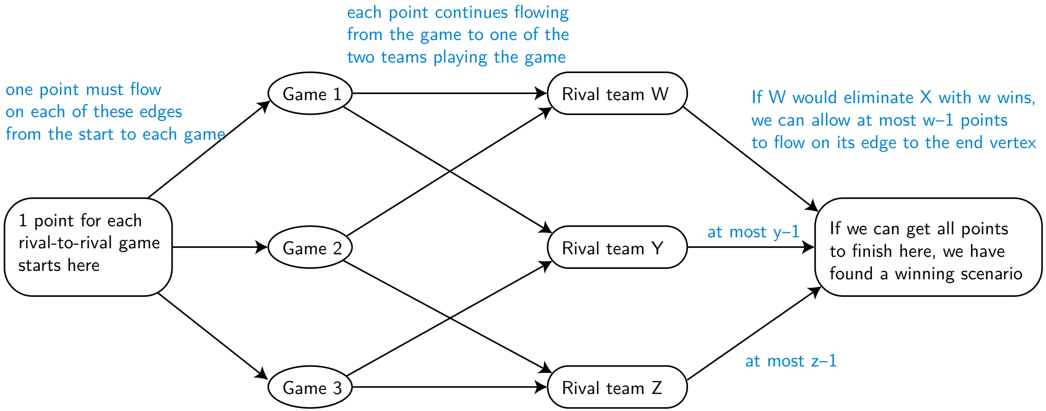
Some simplifying assumptions

If we're trying to test whether team X is eliminated, it's safe to only think about scenarios where X wins all of their remaining games, so we know how many wins team X will end up with

Similarly, if a rival team Y from X's division has a game against an out-of-division team, it's safe to assume that Y loses (because we're looking for the most favorable scenario for X).

The real question is: for the games between pairs of rival teams Y and Z, both from the same division, is it possible to split the wins so that no rival ends up with too many?

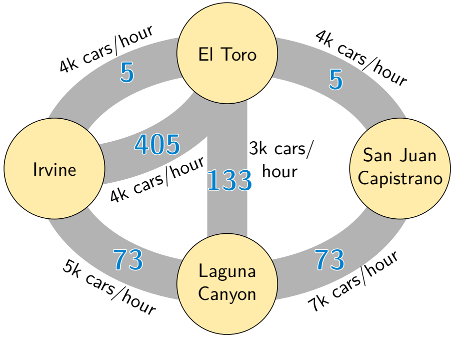
A graphical view


each point continues flowing from the game to one of the two teams playing the game
one point must flow on each of these edges from the start to each game
Game 1
Rival team W
If W would eliminate X with w wins, we can allow at most w−1 points to flow on its edge to the end vertex
1 point for each rival-to-rival game starts here
Game 2
Rival team Y
at most y−1
If we can get all points to finish here, we have found a winning scenario
Game 3
Rival team Z
at most z−1

This kind of graph (with quantities that can flow from vertex to vertex, with limits on how much can flow across each edge) is called a flow network, and finding a way to get everything from start to finish obeying the limits is a flow problem.

Maximum flow

Another application: Traffic engineering



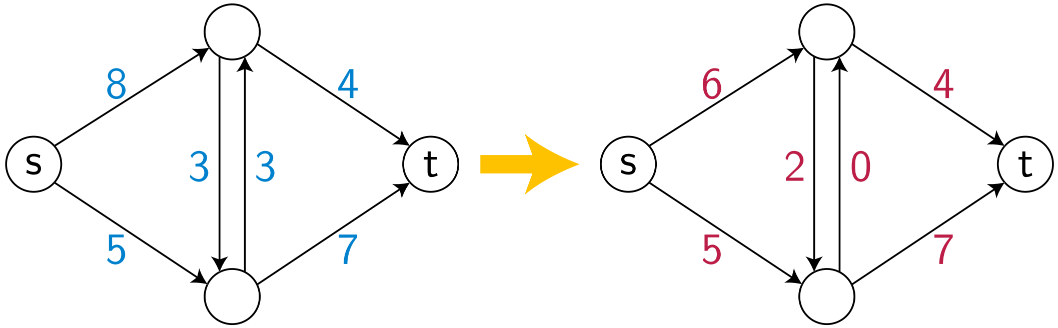
How many cars/hour can get from Irvine to San Juan Capistrano?

Widest path (73) has 5k cars/hour, but we can do much better!

5k (73) + 4k (5) + 2k (405+133+73) = 11k

(Also useful for internet routing if multiple paths are allowed)

The maximum flow problem



Input: Directed graph, designated start vertex s, end vertex t

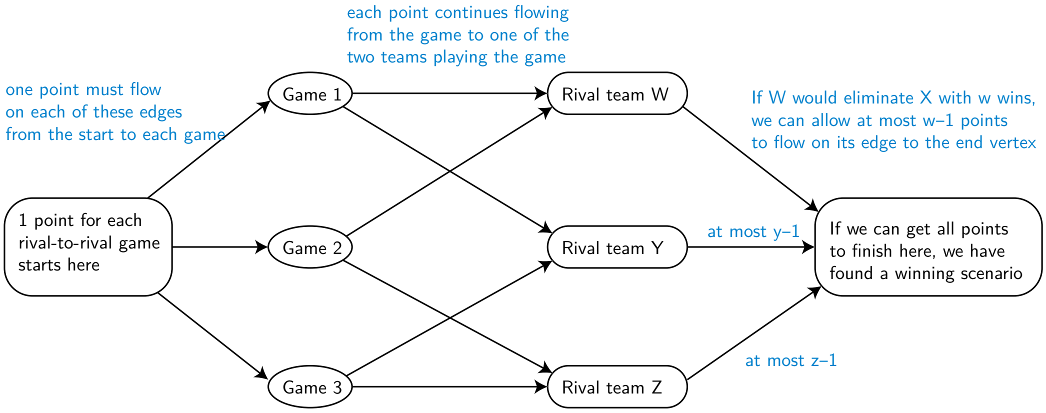
Every edge is labeled with a positive number, its capacity

Output: flow amounts on each edge, $0 \leq$ flow amount $\leq$ capacity

At vertices other than s and t, total flow in $=$ total flow out

Goal: Maximize flow out of $s =$ flow into t

Baseball elimination as maximum flow


each point continues flowing from the game to one of the two teams playing the game
one point must flow on each of these edges from the start to each game
Game 1
Game 2
Game 3
Rival team W
Rival team Y
Rival team Z
If W would eliminate X with w wins, we can allow at most w–1 points to flow on its edge to the end vertex
1 point for each rival-to-rival game starts here
at most y–1
If we can get all points to finish here, we have found a winning scenario
at most z–1

Left and center edges have capacity = 1

RIght edges have capacity = # of allowable wins

Goal: Find a flow with total flow amount = total # of games

Properties

Can be formulated as a linear program

Find flow amounts $f_{i,j}$ for each edge $i \to j$

Satisfy linear constraints $0 \le f_{i,j} \le c_{i,j}$
(where $c_{i,j}$ is the capacity of edge $i \to j$

and

$\sum f_{i,j} = \sum f_{j,k}$ for all vertices $j \notin \{s, t\}$

Maximize linear function $\sum f_{s,i}$

Existence and computability

Even when the capacities are irrational numbers,
there always exists a maximum flow

. . . rather than an infinite sequence of flows converging to the maximum but never quite reaching it

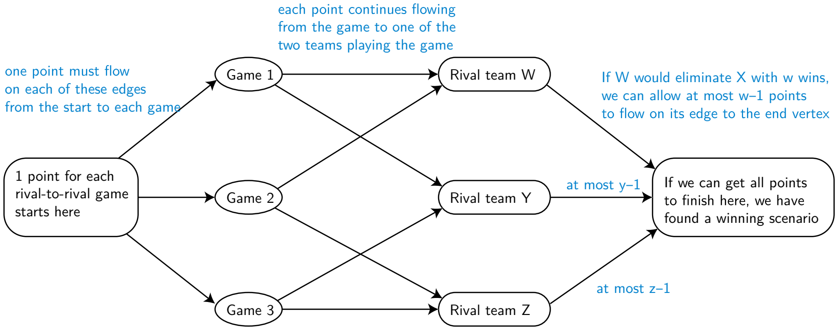
We can find it with an algorithm that runs in polynomial time

Linear programming and some specialized flow algorithms have running time that depends on the numerical value of the capacities, but it can also be solved in time that is polynomial in just n and m, independent of capacity

Integrality

If all the capacities are integers, then there exists a maximum flow all of whose flow amounts are integers

Not true of some other linear programming problems

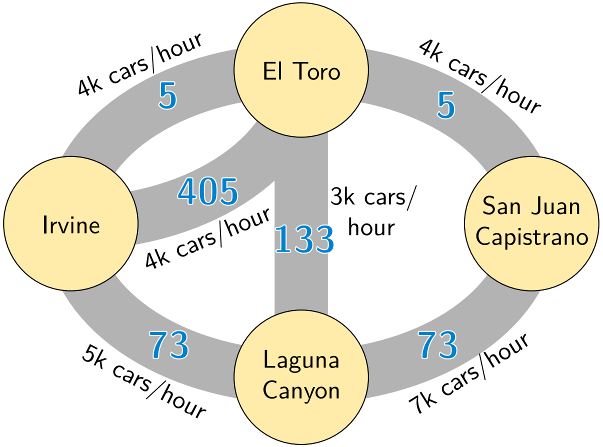


each point continues flowing from the game to one of the two teams playing the game
one point must flow on each of these edges from the start to each game
1 point for each rival-to-rival game starts here
Game 1
Game 2
Game 3
Rival team W
Rival team Y
Rival team Z
If W would eliminate X with w wins, we can allow at most w–1 points to flow on its edge to the end vertex
at most y–1
at most z–1
If we can get all points to finish here, we have found a winning scenario

We need this property for baseball elimination, corresponding to the fact that every game has a winner (no ties)

Decomposition into paths

We can always decompose the flow into multiple paths

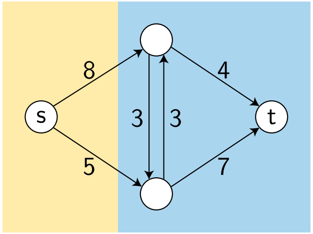


… as we did in the traffic example

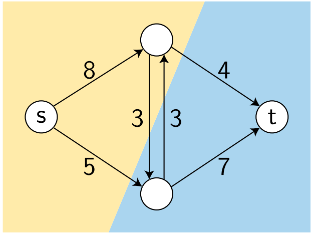
The max-flow min-cut theorem

Cuts

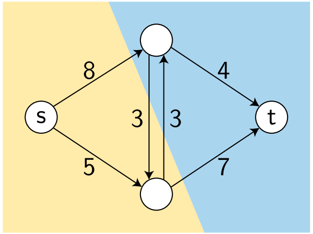
Split vertices into 2 subsets, with s and t in different sets



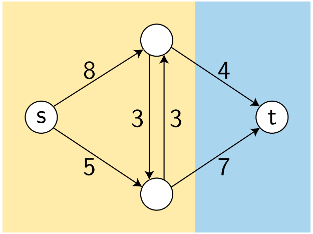
capacity = 8 + 5 = 13



capacity = 5 + 3 + 4 = 12



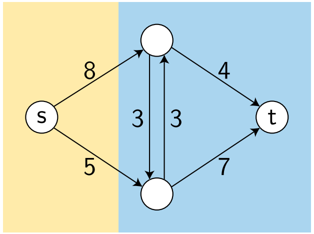
capacity = 8 + 3 + 7 = 18



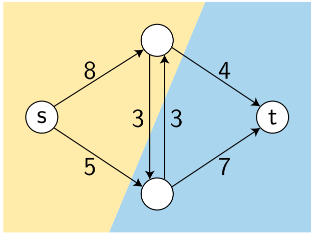
capacity = 4 + 7 = 11

Capacity of a cut

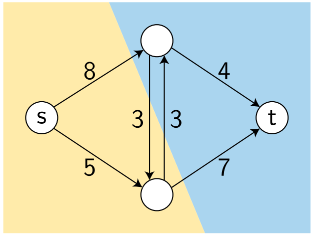
Sum capacities of edges from s's subset to t's subset



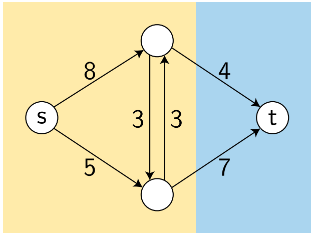
capacity $= 8 + 5 = 13$



capacity $= 5 + 3 + 4 = 12$



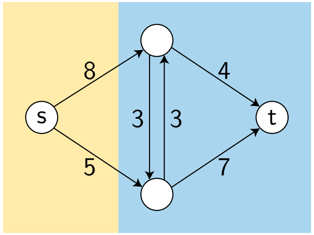
capacity $= 8 + 3 + 7 = 18$



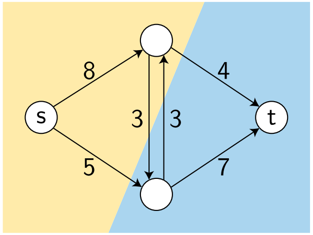
capacity $= 4 + 7 = 11$

Minimum cut problem

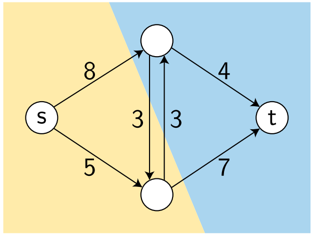
Given a flow network, find cut with smallest capacity



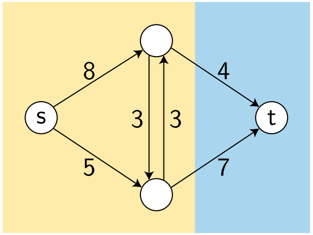
capacity = 8 + 5 = 13



capacity = 5 + 3 + 4 = 12



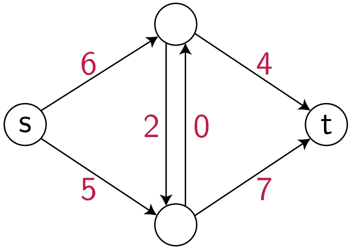
capacity = 8 + 3 + 7 = 18



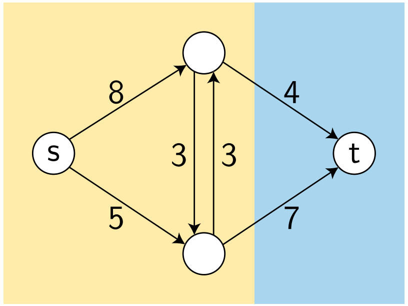
capacity = 4 + 7 = 11

The max-flow min-cut theorem

For every flow network, maximum flow = minimum cut



flow out of s = 6 + 5 = 11



capacity = 4 + 7 = 11

max flow $\leq$ min cut is easy (can't get more flow across the cut)

We will show that, for every flow (maximum or not), either there is an equal cut, or we can increase the flow $\Rightarrow$ max flow $\geq$ min cut

Morals of the story

Definition: at all vertices except s and t, flow in equals flow out

Maximum flow: obey capacity constraints, maximize flow out of s

Applications include baseball elimination, traffic, matching (next week)

Existence of integer flows for integer capacities

The minimum cut problem and max flow min cut theorem