CS 163 & CS 265: Graph Algorithms

Week 4: More paths

Lecture 4a: Widest paths and voting systems

David Eppstein
University of California, Irvine

Winter Quarter, 2024

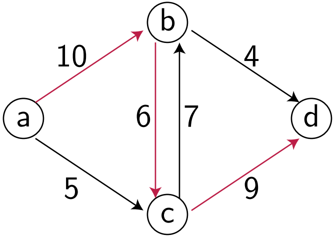
Widest paths

The widest path problem

Input: A directed or undirected graph with weights on edges

Measure quality of paths by the minimum weight of any path edge

Goal: path for which this minimum weight is as large as possible



abd: min(10, 4) = 4

abcd: min(10, 6, 9) = 6

acd: min(5, 9) = 5

acbd: min(5, 7, 4) = 4

We already saw the undirected version in the internet routing application of the minimum spanning tree problem!

Undirected widest path algorithms

Use maximum spanning tree algorithms

Or, for single-source and single-destination version in linear time:

- Find median edge weight
- If source can reach destination using only wider-than-median edges, throw away all smaller edges and recurse
- Else: collapse all connected components of wide edges into “supervertices” and recurse

Time: $O(m) + O(m/2) + O(m/4) + \cdots = O(m)$

Directed widest paths

Use Dijkstra with modified relax (use path width, not length) and with initial values $D[v] = -\infty$

```
def relax(u,v):
    if v in Q and min(D[u],length(edge uv)) > D[v]:
        D[v] = min(D[u],length(edge uv))
        P[v] = u
```

(Compare undirected Jarník, which uses length not min(D,length))

If edges are already sorted, can replace weights by positions in sorted order and use integer priority queue ⇒ linear time (bucket queue — see Wikipedia for details)

Directed single-source single-destination

Look for bottleneck edge in decreasing subsets of edges (initially, all edges)

Repeat until we have a single-edge subset:

- Use recursive median-finding to divide a subset of m/x edges into $O(2^x)$ subsets of $\lceil m/2^x \rceil$ edges, in linear total time
- Use integer priority queue Dikjstra to choose which of these subsets contains the bottleneck

Time $O(m \log^* n)$ where $\log^*$ is the smallest height of a tower $2^{2^{2^{\cdots}}}$ that is $\geq n$

$2^2 = 4$, $2^{2^2} = 16$, $2^{2^{2^2}} = 65536$, $2^{2^{2^{2^2}}} \approx 2 \times 10^{19728}$, $\ldots$

All-pairs directed widest paths

Sort all edges once before finding paths

Run integer priority queue Dijkstra from all starting vertices

$O(m \log n)$ for sorting, $O(mn)$ for $n \times$ modified Dijkstra: $O(mn)$

When number of edges is large, may be preferable to use an algorithm based on fast matrix multiplication with time $O(n^{2.688})$.

Voting systems

Elections with more than two candidates

(Two-candidate election: Just choose majority)

Assumption

Each voter has a preference ordering of candidates, which can be described as a permutation

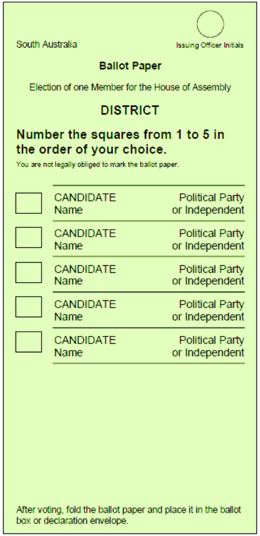
The problem

Given this information, who wins the election?

What algorithm should we use to turn the input (n permutations on c candidates) into an output (a winning candidate or ordering on candidates)?

Simplifying assumption

No ties (complication even for two candidates)



South Australia

○ Issuing Officer Initials

Ballot Paper

Election of one Member for the House of Assembly

DISTRICT

Number the squares from 1 to 5 in the order of your choice.

You are not legally obliged to mark the ballot paper.

□	CANDIDATE Name	Political Party or Independent
□	CANDIDATE Name	Political Party or Independent
□	CANDIDATE Name	Political Party or Independent
□	CANDIDATE Name	Political Party or Independent
□	CANDIDATE Name	Political Party or Independent

After voting, fold the ballot paper and place it in the ballot box or declaration envelope.

Plurality

Most common choice for US elections

- Use only the first-place choices of the voters
- The candidate with the most first-place votes wins

Often used in connection with runoff elections: the two candidates with the most first-place votes go on to a second two-way election, in which the majority wins.

But if we already know all of the voter permutations, we don't have to hold a second runoff election: just use the permutations to predict how each voter would vote in a second election

Instant runoff / single transferable vote

A widely used voting system

- Used in Australia since 1918
- Used for Academy Awards since 2010
- Starting to come into use in US elections e.g. Maine, Alaska

While there is still more than one candidate left:

- Use specified preferences for each voter to find their favorite remaining candidates
- For each candidate, count voters who choose them as favorite
- Eliminate the candidate with the fewest voters

(Can stop early if a candidate becomes favorite of >50% of voters)

Borda count

A different widely used system
(including elections in Iceland, Nauru, and Slovenia)

The favorite system of UCI election-system expert Prof. Don Saari

Method:

- If there are c candidates, then a candidate gets

  $c - 1$ points for being first in any permutation,
  $c - 2$ points for being second,
  $c - 3$ points for being third,
  $\ldots$ zero points for being last

- The candidate with the most points wins!

An example

Preferences:

A, B, C:	11%	B, A, C:	14%	C, A, B:	12%
A, C, B:	24%	B, C, A:	19%	C, B, A:	20%

Plurality:

- A gets $11\% + 24\% = 35\%$,
- B gets $14\% + 19\% = 33\%$
- C gets $12\% + 20\% = 32\%$

Instant runoff:

- C is eliminated
- 12% of voters switch C to A, 20% switch from C to B
- B (53%) beats A (47%)

Borda count:

- Two points for being first choice, one point for second choice
- A gets $2 \times (11 + 24) + 1 \times (14 + 12) = 96$
- B gets $2 \times (14 + 19) + 1 \times (11 + 20) = 97$
- C gets $2 \times (12 + 20) + 1 \times (24 + 19) = 107$

How to choose which system to use?

Idea: Let's first figure out which properties we want the system to obey, and then choose a system that has those properties

- Majority: Candidate with a majority of first-place votes wins
- Weak majority: Candidate with all first-place votes wins

- Symmetry: all voters treated equally
- Weak symmetry: There is no "dictator", a super-voter whose preference is used as the result for all elections

- No spoilers: If candidate X does not win the election, then removing X does not change the outcome among the remaining candidates

Arrow's impossibility theorem

No election system can guarantee the weak majority,
weak symmetry, and no spoiler properties!

Kenneth Arrow proved this in his 1951 Ph.D. dissertation at Columbia University

He later won the Nobel Prize in Economics (1972) and National Medal of Science (2004)

CC-BY image Kenneth Arrow, Stanford University from Wikimedia commons

Another example

Majority + symmetry + no spoilers is impossible for preferences:

A, B, C:	24%	B, A, C:	14%	C, A, B:	20%
A, C, B:	11%	B, C, A:	19%	C, B, A:	12%

Head-to-head comparisons:

- A beats B, 55 to 45
- B beats C, 57 to 43
- C beats A, 51 to 49

Spoilers:

- If B wins, C is a spoiler
- If C wins, A is a spoiler
- If A wins, B is a spoiler

The Condorcet property and widest path voting

A weaker no-spoiler property

Condorcet property

If there is a candidate X who wins every head-to-head comparison then X should win the whole election

First example

Head-to-head results:

- B beats A, 53 to 47
- C beats A, 51 to 49
- C beats B, 56 to 44

C is the Condorcet winner

Second example

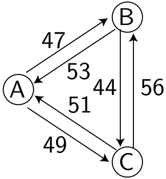
Cyclic head-to-head results (A beats B, B beats C, C beats A): No Condorcet winner

Nicolas de Condorcet (1743–1794), from the Palace of Versailles

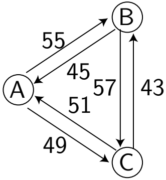
Widest path / Schulze method

Direct edges between all candidates $X \to Y$, weight $= \#$ voters who prefer X to Y

first example:

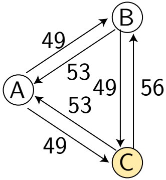


second example:

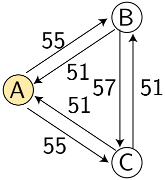


Relabel each edge $X \to Y$ with the width of the widest path from X to Y in the original graph (e.g. in first example, path A→C→B has width $\min(49, 56) = 49$)

first example:



second example:



Find a candidate who wins all the pairwise comparisons in the relabeled graph

Properties of the widest path method

Easy: if there is a Condorcet winner, method finds it

X wins all head-to-head $\Rightarrow$ edge weights into X $<$ 50%, out $>$ 50%

Path width into X is at most weight of best incoming edge, $<$ 50%

Path width out is at least as good as the one-edge path, $>$ 50%

Harder: The method always finds a winner

Head-to-head comparisons using widest-path widths are transitive:

If best path $X \rightsquigarrow Y$ is wider than best path $Y \rightsquigarrow X$
and best path $Y \rightsquigarrow Z$ is wider than best path $Z \rightsquigarrow Y$
then best path $X \rightsquigarrow Z$ is wider than best path $Z \rightsquigarrow X$

So they give a consistent ordering we can use to sort all candidates

Elections using the widest path method

Some cities:

Silla, Spain; Turin, Italy, Southwark, England

Various national-level Pirate Parties

Some technical organizations:

ACM, IEEE, Wikimedia, some Linux groups

Biggest obstacle:
Hard to explain to voters who aren't computer scientists

Morals of the story

Widest paths can be computed efficiently,
as easy or easier than shortest paths

No reasonable election system can prevent spoiler candidates

Widest paths can be used to define a system that obeys
a weaker no-spoiler property, the Condorcet property,
not obeyed by other commonly-used alternatives

Obstacles to its adoption are social rather than technical