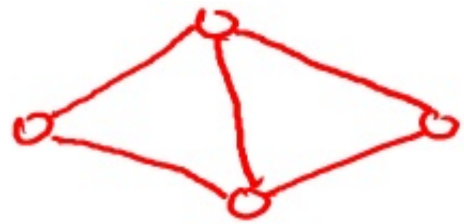
Design and Analysis of Algorithms I

Graph Algorithms

Representing Graphs

Graphs



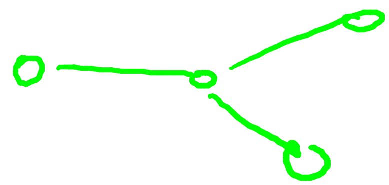
Two ingredients

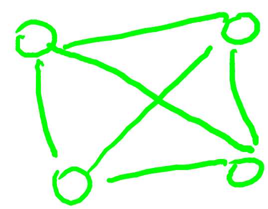
- Vertices aka nodes (V)
- Edges (E) = pairs of vertices
    - can be undirected [unordered pair] or directed [ordered pair] (aka arcs)

Examples: road networks, the Web, social networks, precedence constraints, etc.

Consider an undirected graph that has n vertices, no parallel edges, and is connected (i.e., “in one piece”). What is the minimum and maximum number of edges that the graph could have, respectively ?

- $n-1$ and $n(n-1)/2$
- $n-1$ and n^2
- n and 2^n
- n and n^n





Ter
ver

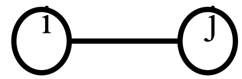
Sparse vs. Dense Graphs

Let n = # of vertices, m = # of edges.

In most (but not all) applications, m is $\Omega(n)$ and $O(n^2)$

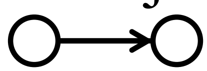
- in a "sparse" graph, m is or is close to $O(n)$
- in a "dense" graph, m is closer to $\theta(n^2)$

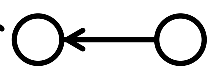
The Adjacency Matrix

Represent G by a n x n 0-1 matrix A where
$A_{ij} = 1 \Leftrightarrow$ G has an i-j edge 

<u>Variants</u>

- A_{ij} = # of i-j edges (if parallel edges)
- A_{ij} = weight of i-j edge (if any)
- $A_{ij} = \begin{cases} +1 \text{ if} \\ -1 \text{ if} \end{cases}$

+1 if 

-1 if 

Tim Roughgarden

Ter
ver

How much space does an adjacency matrix require, as a function of the number n of vertices and the number m of edges?

- $\theta(n)$
- $\theta(m)$
- $\theta(m+n)$
- $\theta(n^2)$

Adjacency Lists

<u>Ingredients</u>

- array (or list) of vertices
- array (or list) of edges
- each edge points to its endpoints
- each vertex points to edges incident on it

Te
ve

How much space does an adjacency list representation require, as a function of the number n of vertices and the number m of edges?

- $\theta(n)$
- $\theta(m)$
- $\theta(m+n)$
- $\theta(n^2)$

Adjacency Lists

Ingredients

- array (or list) of vertices
- array (or list) of edges
- each edge points to its endpoints
- each vertex points to edges incident on it

one-to-one correspondence !

Space
$\theta(n)$
$\theta(m)$
$\theta(m)$
$\theta(m)$
$\theta(m+n)$
$[or\ \ \theta(max\{m,n\})]$

Question: which is better?

Answer: depends on graph density and operations needed.

This course: focus on adjacency lists.

Design and Analysis of Algorithms I

Contraction Algorithm

Overview

Goals for These Lectures

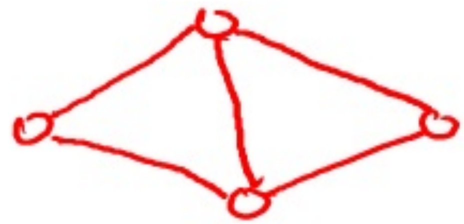
- Further practice with randomized algorithms
  - In a new application domain (graphs)
- Introduction to graphs and graph algorithms

Also: "only" 20 years ago!

Graphs

Two ingredients

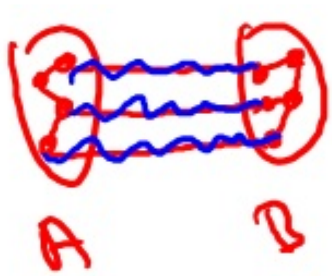
- Vertices aka nodes (V)
- Edges (E) = pairs of vertices
  - can be undirected [unordered pair] or directed [ordered pair] (aka arcs)



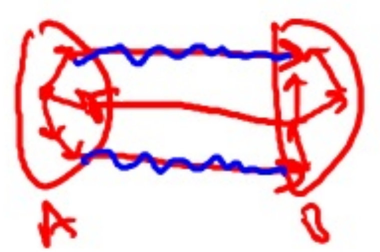
Examples: road networks, the Web, social networks, precedence constraints, etc.

Cuts of Graphs

<u>Definition:</u> a cut of a graph (V, E) is a partition of V into two non-empty sets A and B.



[undirected]



[directed]

<u>Definition</u>: the crossing edges of a cut (A, B) are those with:

- the one endpoint in each of (A, B) [undirected]
- tail in A, head in B [directed]

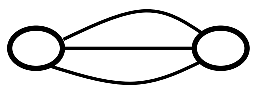
Roughly how many cuts does a graph with n vertices have?

- n
- n^2
- 2^n
- n^n

Ter
ver

The Minimum Cut Problem

- INPUT: An undirected graph G = (V, E).

[Parallel  edges allowed]

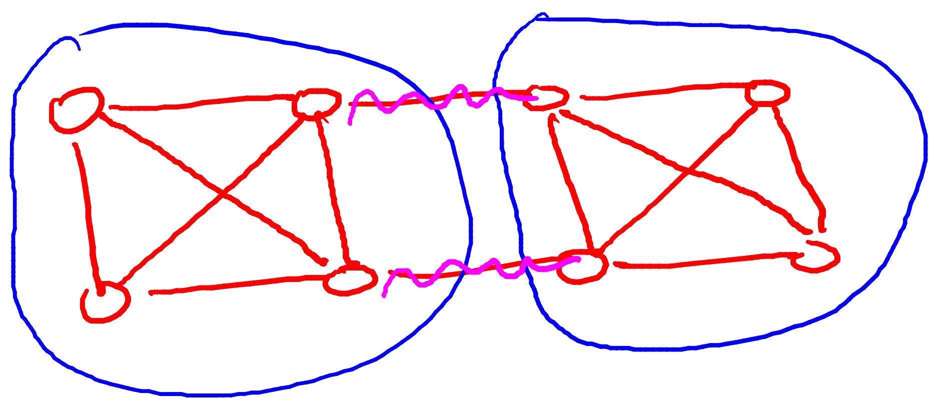
[See other video for representation of the input]

- GOAL: Compute a cut with fewest number of crossing edges. (a min cut)

Ter
ver

What is the number of edges crossing a minimum cut in the graph shown below?

- 1
- 2
- 3
- 4



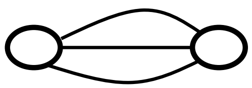
Contraction Algorithm

The Algorithm

Design and Analysis of Algorithms I

The Minimum Cut Problem

- INPUT: An undirected graph G = (V, E).

[Parallel  edges allowed]

[See other video for representation of the input]

- GOAL: Compute a cut with fewest number of crossing edges. (a min cut)

Random Contraction Algorithm

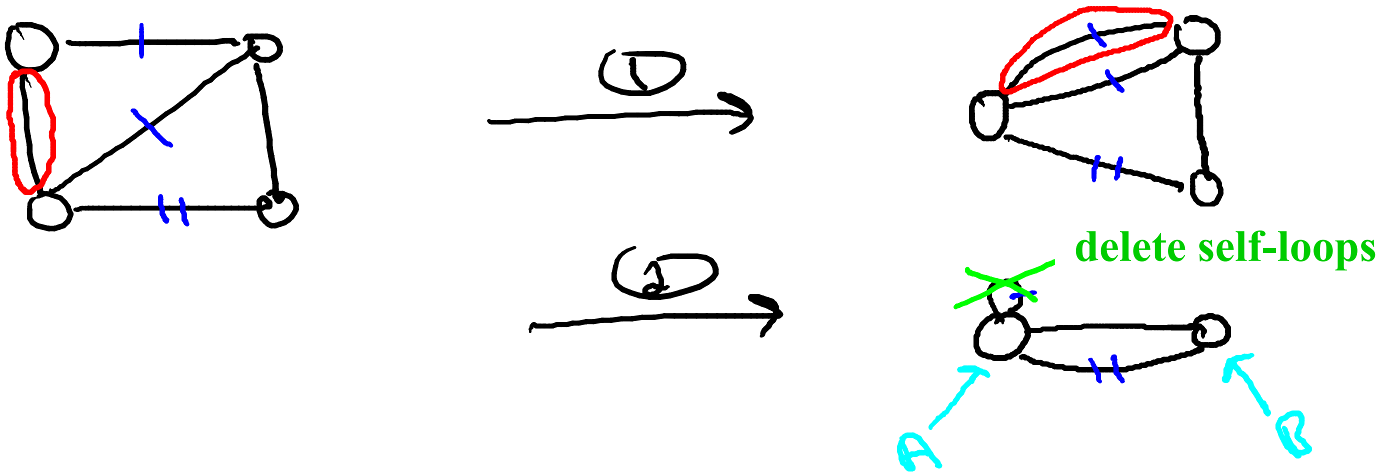
[due to Karger, early 90s]

While there are more than 2 vertices:

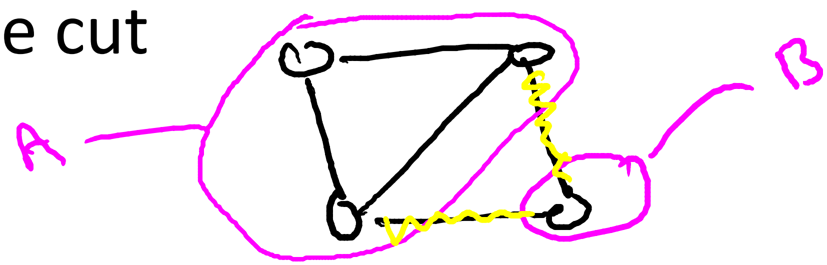
- pick a remaining edge (u,v) uniformly at random
- merge (or "contract") u and v into a single vertex
- remove self-loops

return cut represented by final 2 vertices.

Example



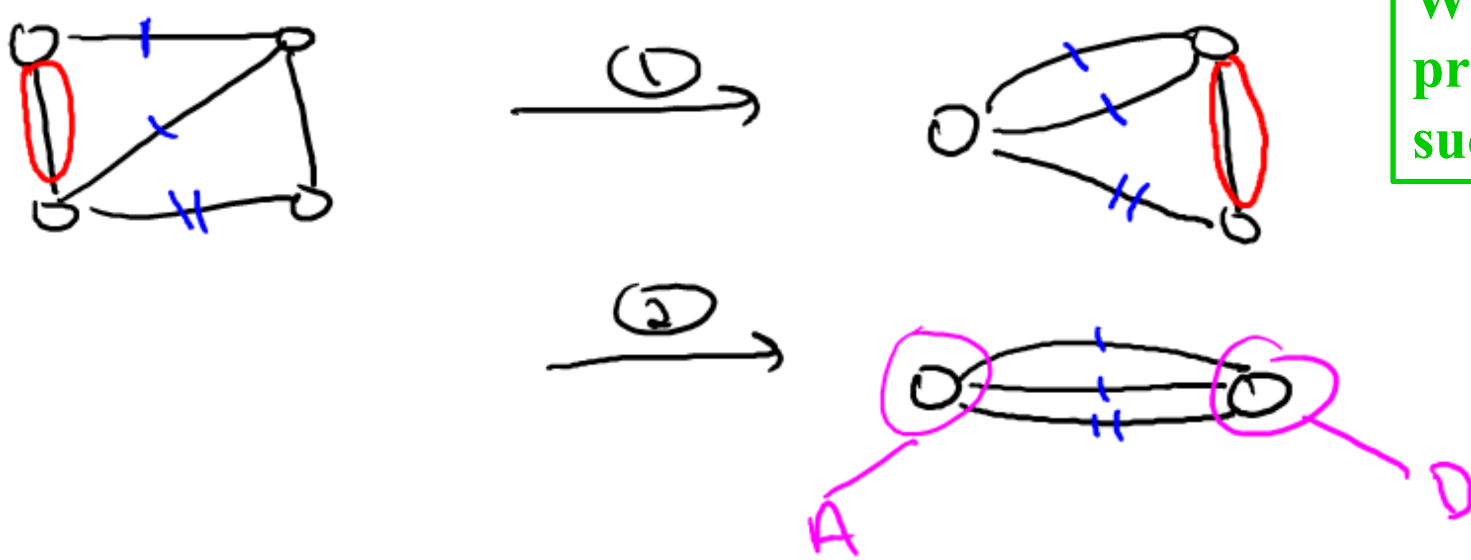
=> Corresponds to the cut



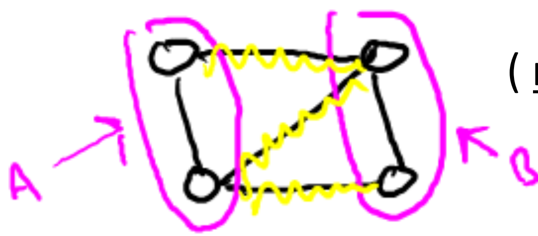
(a min cut!)

Example (con'd)

<u>KEY QUESTION:</u> What is the probability of success?



- Corresponds to the cut



(<u>not</u> a min cut!)

A Few Applications

- indentify network bottlenecks / weaknesses
- community detection in social networks
- image segmentation
    - input = graph of pixels
    - use edge weights

        [(u,v) has large weight ⇔ "expect" u,v to come from some object]

    hope: repeated min cuts identifies the primary objects in picture.

Design and Analysis of Algorithms I

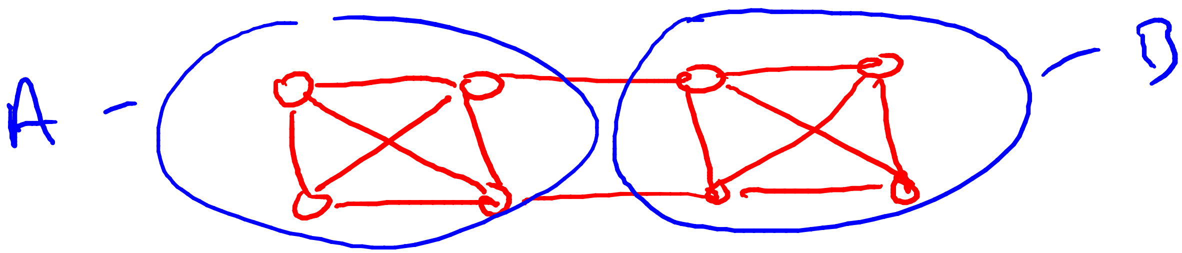
Contraction Algorithm

The Analysis

The Minimum Cut Problem

Input: An undirected graph G = (V, E).
[parallel edges  allowed]
[See other video for representation of input]

Goal: Compute a cut with fewest number of crossing edges.
(a min cut)



Random Contraction Algorithm

[due to Karger, early 90s]

While there are more than 2 vertices:

- pick a remaining edge (u,v) uniformly at random
- merge (or "contract") u and v into a single vertex
- remove self-loops

return cut represented by final 2 vertices.

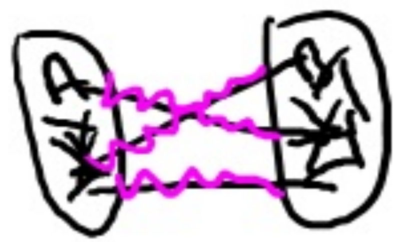
The Setup

<u>Question</u>: what is the probability of success?

Fix a graph G = (V, E) with n vertices, m edges.

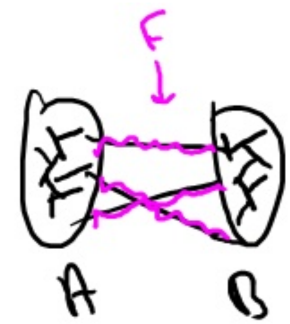
Fix a minimum cut (A, B).

Let k = # of edges crossing (A, B). (Call these edges F)



What Could Go Wrong?

1. Suppose an edge of F is contracted at some point
   ⇨ algorithm <u>will not</u> output (A,B).



2. Suppose only edges inside A or inside B get contracted ⇨ algorithm <u>will</u> output (A, B).

<u>Thus</u>: Pr [output is (A, B)] = Pr [never contracts an edge of F]

Let S_i = event that an edge of F contracted in iteration i.
<u>Goal</u>: Compute $\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge \ldots\ldots \wedge \neg S_{n-2}]$

What is the probability that an edge crossing the minimum cut (A, B) is chosen in the first iteration (as a function of the number of vertices n, the number of edges m, and the number k of crossing edges)?

- k/n
- k/m
- k/n^2
- n/m

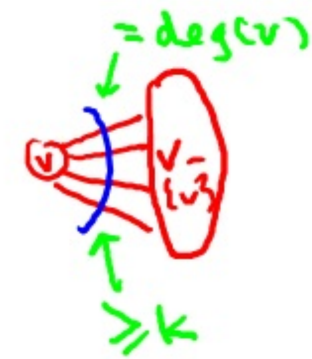
$$\Pr[S_1] = \frac{\text{\# of crossing edges}}{\text{\# of edges}} = \frac{k}{m}$$

Ter
ver

The First Iteration

<u>Key Observation</u>: $\underbrace{\text{degree}}_{\text{\# of incident edges}}$ of each vertex is at least k

<u>Reason</u>: each vertex v defines a cut ({v}. V-{v}).



Since $\underbrace{\sum_v degree(v)}_{\geq kn} = 2m$, we have $m \geq \frac{kn}{2}$

Since $\Pr[S_1] = \frac{k}{m}$, $\Pr[S_1] \leq \frac{2}{n}$

The Second Iteration

<u>Recall:</u> $\Pr[\neg S_1 \wedge \neg S_2] = \underbrace{\Pr[\neg S_2|\neg S_1]}_{= 1 - \frac{k}{\text{\# of remaining edge}}} . \Pr[\neg S_1]$

$\geq (1 - \frac{2}{n})$

what is this?

Note: all nodes in contracted graph define cuts in G (with at least k crossing edges).

➢ all degrees in contracted graph are at least k

So: # of remaining ed $\geq \frac{1}{2}k(n-1)$

So $\Pr[\neg S_2|\neg S_1] \geq 1 - \frac{2}{(n-1)}$

All Iterations

<u>In general</u>:

$$
\begin{aligned}
&\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge \ldots. \wedge \neg S_{n-2}] \\
&\quad = \underline{\Pr[\neg S_1]}\,\underline{\Pr[\neg S_2 | \neg S_1]}\,\Pr[\neg S_3 | \neg S_2 \wedge \neg S_1] \ldots\ldots \Pr[\neg S_{n-2} | \neg S_1 \wedge \ldots \wedge \neg S_{n-3}] \\
&\ \geq (1-\tfrac{2}{n})(1-\tfrac{2}{n-1})(1-\tfrac{2}{n-2})\ldots..(1-\tfrac{2}{n-(n-4)})(1-\tfrac{2}{n-(n-3)}) \\
&\ = \frac{\cancel{n-2}}{n} \cdot \frac{\cancel{n-3}}{n-1} \cdot \frac{\cancel{n-4}}{\cancel{n-2}} \ldots\ldots \frac{2}{\cancel{4}} \cdot \frac{1}{\cancel{3}} = \frac{2}{n(n-1)} \geq \frac{1}{n^2}
\end{aligned}
$$

<u>Problem</u>: <u>low</u> success probability! (<u>But</u>: non trivial)

recall $\simeq 2^n$ cuts !

Repeated Trials

Solution: run the basic algorithm a large number N times, remember the smallest cut found.

Question: how many trials needed?

Let T_i = event that the cut (A, B) is found on the i^{th} try.

➢ by definition, different T_i's are independent

So: $\Pr[\text{all N trails fail}] = \Pr[\neg T_1 \wedge \neg T_2 \wedge \ldots \wedge \neg T_N]$

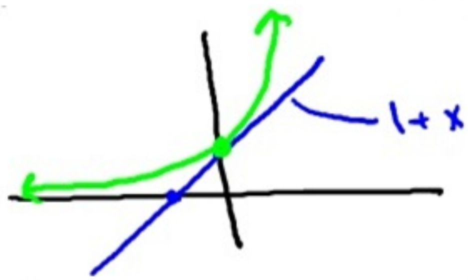
$$= \prod_{i=1}^{N} \Pr[\neg T_i] \leq \left(1 - \frac{1}{n^2}\right)^N$$

By independence !

Repeated Trials (con'd)

$\Pr[\text{all trials fail}] \leq (1 - \frac{1}{n^2})^N$

Calculus fact: $\forall$real numbers x, $1+x \leq e^x$



So: if we take $N = n^2, \Pr[\text{all fail}] \leq (e^{-\frac{1}{n^2}})^{n^2} = \frac{1}{e}$

If we take $N = n^2 \ln n, \Pr[\text{all fail}] \leq (\frac{1}{e})^{\ln n} = \frac{1}{n}$

Running time: polynomial in n and m but slow $(\Omega(n^2 m))$

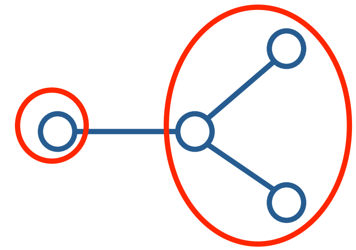
But: can get big speed ups (to roughly $O(n^2)$) with more ideas.

Contraction Algorithm

Counting Mininum Cuts

Design and Analysis of Algorithms I

The Number of Minimum Cuts



NOTE: A graph can have multiple min cuts.
[e.g., a tree with n vertices has (n-1) minimum cuts]

QUESTION: What's the largest number of min cuts that a graph with n vertices can have?

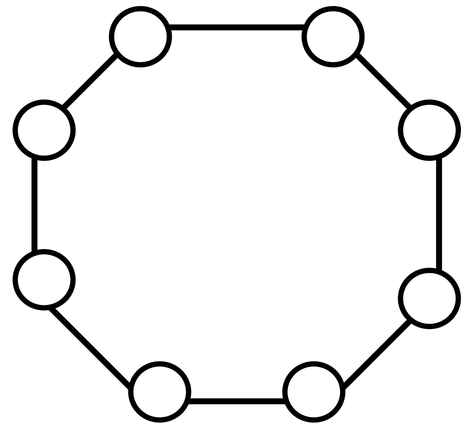
ANSWER: $$\binom{n}{2} = \frac{n(n-1)}{2}$$

The Lower Bound

Consider the n-cycle.

<u>NOTE</u>: Each pair of the n edges defines a distinct minimum cut (with two crossing edges).

➢ has $\geq \binom{n}{2}$ min cuts



(n = 8)

The Upper Bound

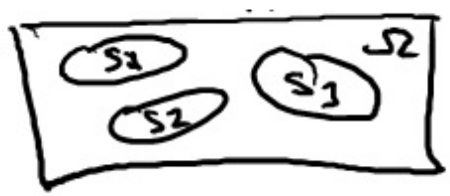
Let (A_1, B_1) , (A_2, B_2) , …, (A_t, B_t) be the min cuts of a graph with n vertices.
By the Contraction Algorithm analysis (without repeated trials):

$$\Pr[\underbrace{output = (A_i, B_i)}_{S_i}] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}} \quad \forall i = 1, 2, .., t$$

Note: S_i's are disjoint events (i.e., only one can happen)

- ➢ their probabilities sum to at most 1

Thus: $$\frac{t}{\binom{n}{2}} \leq 1 \Rightarrow t \leq \binom{n}{2}$$



QED !