Week 3

Randomized Selection

Input: given i^{th} element and array A (i^{th} order statistic)
and find i^{th} order statistic
(i.e. i^{th} small element)

example: median $\left(\frac{i}{n} \text{ or } \frac{i+1}{2} \right)$

Reduction to Sorting
$O(n \log n)$
1) apply merge sort
2) return i^{th} element

Can we do better? Yes!

$\Theta(n)$ time, (randomized) -
by modoficating QSort

Recall partition function of Quick Sort

. Pivot is on it's position! after partition

how to find i^{th} order?

- Suppose we are looking for 5^{th} element in input array of len 10.
- We partition the array and the ~~ny~~ pivot winds up in the third position
- So we should look for 2^{nd} order statistic $(5-3)$ on the right side of the pivot

RSelect (array A, length n, order st. i)

if $n = 1$ return A[1]

choose pivot p from A at random

partition A around p

< p	p	> p
Left	p	Right

j = new index of p

if $j = i$ return p

if $j > i$ return RSelect(L side of A, $j-1$, i)

if $j < 1$ return RSelect (R side of A, $n-j$, $i-j$)

Running Time? -

depends on "quality" of pivot

the worst case - $\Theta(n^2)$

Best Pivot - the median

$$T(n) \leq T\left(\frac{n}{2}\right) + O(n)$$

$\Downarrow$ [case 2 of Master Method]

$$T(n) = O(n)$$

for medians

The avg time is $O(n)$

A Determenistic Choose Pivot

ChoosePivot(A, n)

Logically break A into $\frac{n}{2}$ groups of size 5 each

sort each group

copy $n/5$ medians (i.e. middle elements) into new array C

recursive compute median of C

return the pivot

$\Downarrow$

DSelect(array A, len n, order stat i)

break A into groups of 5

$C = \frac{n}{5}$ middle elements

$p = DSelect(C, \frac{n}{5}, \frac{n}{10})$

} Choose Pivot

Same as before

Partition A around p

if $j=i$ return p

if $j<i$ return DSelect (L part A , $j-1$, i)

if $j>i$ return DSelect(R part A, $n-j$, $i-j$)

Theorem – it runs in $O(n)$ time

(not good as Rselect in practice)

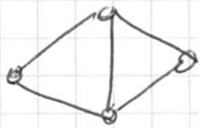
– worse constrains
– not-in-place

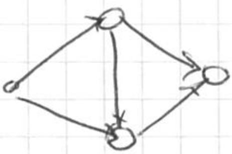
Graphs and Minimum Cuts

Graph contains

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

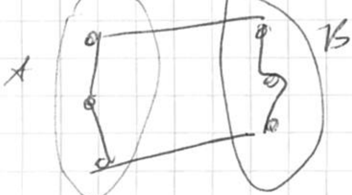
Examples: road networks, the Web,
social networks





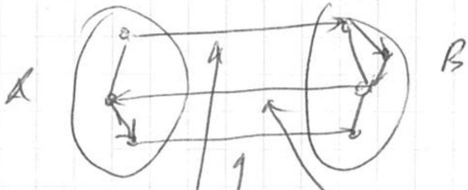
Cut

a cut is a partition of V into 2 two non-empty sets A and B



Crossing edges are

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



Graph with n verticies can be cut into 2^n cuts

crossing edges

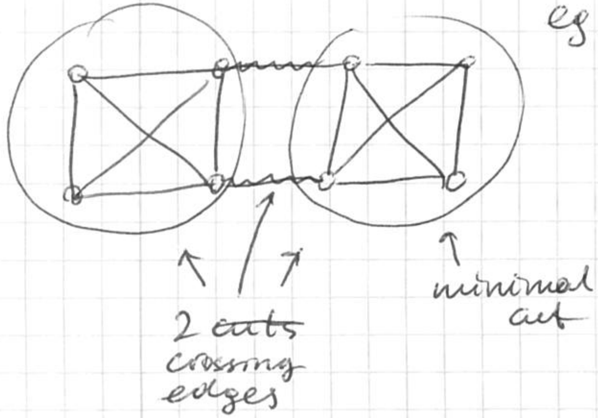
non-crossing edge
(head in B, tail in A)

The minimal cut problem

input an undirect graph $G = (V, E)$
[parallel edges allowed]

goal;
compute a cut with fewer number of crossing edges (a min cut)

eg:



Applications:

- identify weaknesses / bottlenecks
- detect communities in social networks
- image segmentation
  - input - 2D array

    identify objects in an image

Graph Representation

$$\begin{cases} \text{minimal number of edges} - n-1 \\ \text{max} - \dfrac{n(n-1)}{2} \ (C_n^2) \end{cases}$$

complete graph

Sparse vs Dense Graphs

n = number of vertices

m = number of edges

in most application m is $\Omega(n)$ and $O(n^2)$

„sparse" – close to the lower bound

„dense" – close to $O(n^2)$

The adjacency matrix

$n \times n$ matrix where

$A_{ij} = 1$ if G has i-j edge

Variants

A_{ij} = number of edges (in parallel gr)

A_{ij} = weight of i-j edge

$$A_{ij} = \begin{cases} +1 & \circ \longrightarrow \circ \\ -1 & \circ \longleftarrow \circ \end{cases}$$

Space required - $\Theta(n^2)$

(not good for sparse)

vertex (вершина)
edge (грань)

Adjacency List

$\Theta(n)$ • array of vertices

$\Theta(m)$ • array of edges

$\Theta(m)$ • each edge points to its endpoint

$\underline{\Theta(m)}$ • each vertex points to edges incident on it

$\Theta(n+m)$

Space required - $\Theta(n+m)$

number of vertices

number of edges

Random Contraction Algorithm

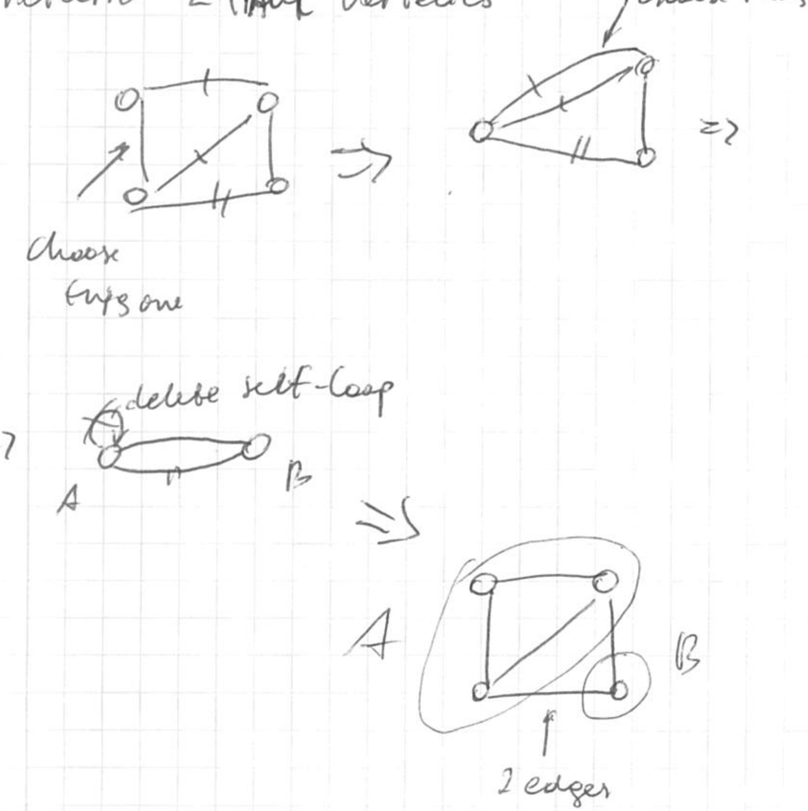
for Minimal Cut Problem

While (there are more than 2 vertices)

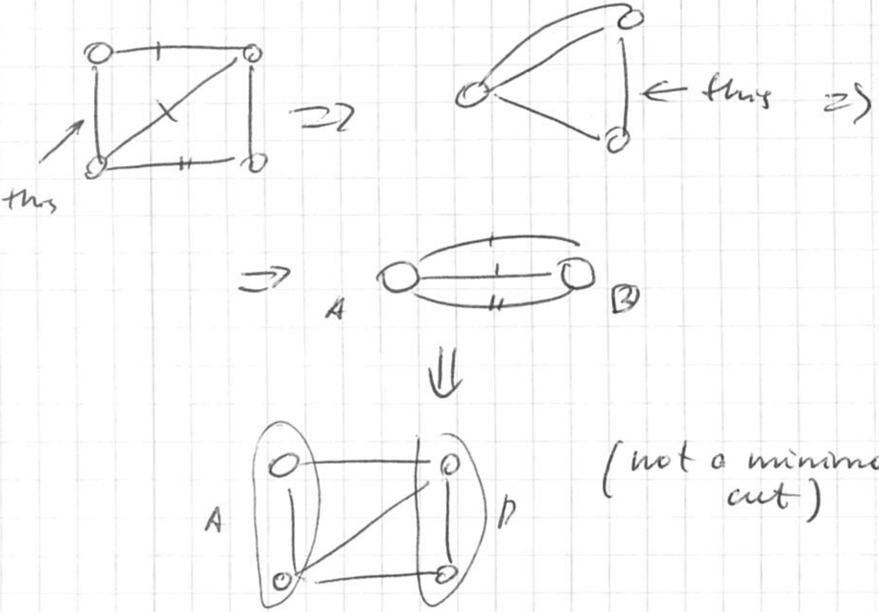
- pick a remaining edge (u, v) at random
- merge (or "contract") u and v into a single vertex
- remove self-loops
- ~~remove se~~

return 2 final vertices

e.g.



e.g. exact same input



(not a minimal cut)

Key question:

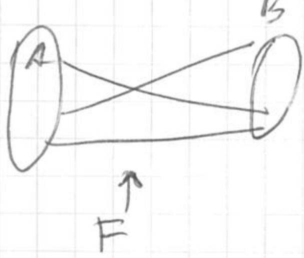
What is the probability of success?

Setup.

$G = (V, E)$ n vertices
m edges

Fix a min cut (A, B)

Let k = # of crossing (A, B) edges (let's call them F)



What could go wrong?

1. If one of F edges is chosen for contraction

$\Rightarrow$ algorithm will now output (A,B)

2. If none of F never chosen,

$\Rightarrow (A,B)$ is output

success!

Thus $Pr[\text{output} = (A,B)] =$

$= Pr[\text{never contract any edge of } F]$

So, for iteration i

S_i = event that an edge of F contracted in iteration i

Goal: Compute $Pr[\bar{S}_1 \wedge \bar{S}_2 \wedge \bar{S}_3 \wedge \ldots \wedge \bar{S}_{n-2}]$

F not contracted

k/m - ~~fr that~~

k - crossing edges

m - number of edges

$$Pr[S_1] = k/m$$

First Iteration

degree (number of incident edges) of each vertex is at least k

smallest amount of vertex possible

$$\underbrace{\sum_v \text{degree}(v)}_{\geq kn} = 2m$$

true for any undirected graph

so we have $m \geq \frac{kn}{2}$

$$Pr[S_1] = \frac{k}{m} \Rightarrow \underbrace{Pr[S_1] \leq \frac{2}{n}}_{\text{small number}}$$

Second Iteration

$$Pr[\bar{S}_1 \wedge \bar{S}_2] = Pr[\bar{S}_2 | \bar{S}_1] \cdot Pr[\bar{S}_1]$$

$1 - \frac{1}{\#\text{ remaining edges}}$ $\quad \hookrightarrow \left(1 - \frac{2}{n}\right)$

Note:

$\Rightarrow$ all degrees in contracted graph are at least k.

\# of remaining edges

$$\geq \frac{1}{2}\underbrace{k}_{\text{degree}}\underbrace{(n-1)}_{\text{number of vertices}}$$

So $Pr[\bar{S}_2 | \bar{S}_1] \geq 1 - \frac{2}{(n-1)}$

All iterations

$$Pr[\bar{S}_1 \wedge \dots \wedge \bar{S}_n] =$$

$$= Pr[\bar{S}_1]\, Pr[\bar{S}_1 | \bar{S}_2] \cdot \dots \cdot Pr[\bar{S}_{n-2} | \bar{S}_1 \wedge \dots \wedge S_{n-3}] \geq$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\dots\left(1 - \frac{n}{n-(n-4)}\right)\left(1 - \frac{2}{n(n-3)}\right) =$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \dots \cdot \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)} \geq \frac{1}{n^2}$$

low success probability

Repeated Trials:

Solution: try N times and remember the smallest cut found

how many trials we need?

T_i – ~~trial~~ cut (A, B) is found on the i^{th} trial

$\{T_i\}$ are independent

so $Pr[\text{all } N \text{ trials fail}] =$

$$= Pr[\bar{T}_1 \wedge \bar{T}_2 \wedge \bar{T}_3 \dots \bar{T}_n] =$$

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

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

$$= \frac{1}{n}$$

Running time polinomial $\Omega(n^2 m)$