WEEK 8 L1

The Problem:

Input: Array A with n numbers, number i, $1 \leq i \leq n$

Output: ith order statistic (ith smallest element of A)

($i = \frac{n+1}{2}$ for n odd)

($i = n/2$ for n even)

Reduction to Sorting Approach

① Apply a Merge Sort, after sorted return ith element of array

Can we do better?

Fact: can't sort any faster than n log n (optional video)

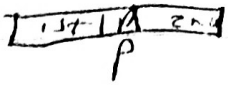
Next: $O(n)$ time (randomized ok) by modifying Quicksort

You can do it without randomization (deterministic) it's just a little more complicated, uses median of medians

D&C, find pivot, partition, look to right or left for jth statistic depending on what statistic you want, & end when the pivot is

Example: $i=10$, pivot ended up in 3 and we're looking for 8th statistic. We look right of the pivot for 2nd statistic.

RSelect (A (len(A) = n) , i)



0. if n = 1 return A[1]
1. Choose pivot p from A uniformly at random
2. partition A around p, let j = new index of p
3. if j = i return p
4. if j > i return RSelect (1st, i)
5. if j < i return RSelect (2nd, i-j)

Claim: it's correct
Proof: by induction

Running Time? Depends. ~~Best case O(n log n), worst case O(n^2)~~ Worst case

The best pivot is the median, but this is circular. If we did that,

Recurrence $T(n) \leq T\left(\frac{n}{2}\right) + O(n)$ (case 2, $T(n) = O(n)$)

Hope: random pivot is "good enough", "often enough"

WEEK 4 L2

Analysis of RSelect

Proof that it's O(n):

Lemma: RSelect uses $\leq cn$ operations outside the recursive call

Notation: RSelect is in phase j if current array size between $\left(\frac{3}{4}\right)^{j+1} n$ and $\left(\frac{3}{4}\right)^{j} n$

- X_j: for a phase j, it measures the number of recursive calls during phase j

Note: running time of RSelect $\leq \sum_{\text{phase } j} X_j \cdot c \cdot \left(\frac{3}{4}\right)^j n$ (*)

(X_j: RANDOM!; $\sum$: this over all phases; c: amount of work on each subproblem; $\left(\frac{3}{4}\right)^j n$: $\leq$ array size at phase j)

X_j = # of recalls in phase j

Probability of 25-75 split or better is 50%, "average" case!

$$E[N_j] = 1 + \frac{1}{2} \cdot E[N_j] \text{ etc } \ldots$$

Average number of coin flips to get "heads" is 2

(*) becomes $\leq 2cn \sum_{\text{phase } j} (3/4)^j = 2cn \cdot \frac{1}{1-\frac{3}{4}} = 8cn$, $O(n)$! :)

WEEK 4 LESSON 7, Graphs & contr

Minimum cuts problems (solutions) in Graphs are elegant. Nice.

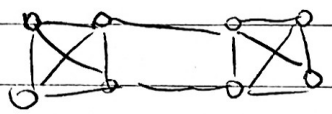
Vertices or Nodes (V)
Edges = (E)
can be directed or undirected

Definition of a cut of a graph (V,E):
it is a partition of (V) into two non empty sets A and B

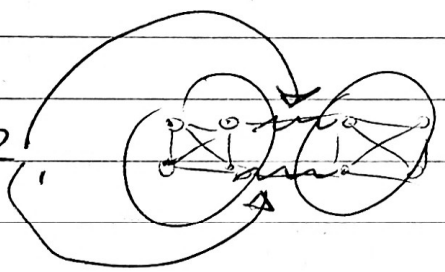
Definition: The crossing edges of a cut (A, B) are those with
- one endpoint in each of (A, B) [undirected]
- tail in A, head in B (directed)

The Minimum Cut problem: For undirected G(V,E
← parallel edges allowed in repr

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



min cut #s CEs are 2.



A few Applications:
- Identify the network bottlenecks / weaknesses
- Community detection in Social Networks
- Image segmentation
  - group of pixels
  - use edge weights (similar color has more weight)

WEEK 4 LESSON 8

What is input size for a graph? We have Vertices and Edges.

Edges are at ~~most~~ least $n-1$, at most $\frac{n(n-1)}{2}$

Sparse vs dense graphs

For n = # of V, m = # of E.

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

In a "sparse graph", m is $O(n)$ or close
In a "dense graph", m is closer to $O(n^2)$

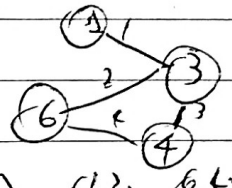
Adjacency Matrixes

$n \times n$ 0-1 matrix, directed or undirected or upper triangular.
OR use -1 to denote direction

If weighted, weight instead of (0 or 1)

Space needed for matrix is $\Theta(n^2)$

Adjacency Lists



- Array of vertices
- Array of edges
- Each edge points to its endpoints (3,6), (3,4), (1,3), (1,5) and can be directed or
- Each vertex points to edges incident on it [3, [1, 2, 3]]

Space needed for Adj List is $\Theta(m+n)$

Which is better then?

Answer: depends on graph density and operations needed

This course will focus on adjacency lists.
They are the best kind for search.

WEEK 4 LECTION 8

MCUT Problem:

Input: undirected G (V, E)

Goal: out of 2^n cuts, which is best?

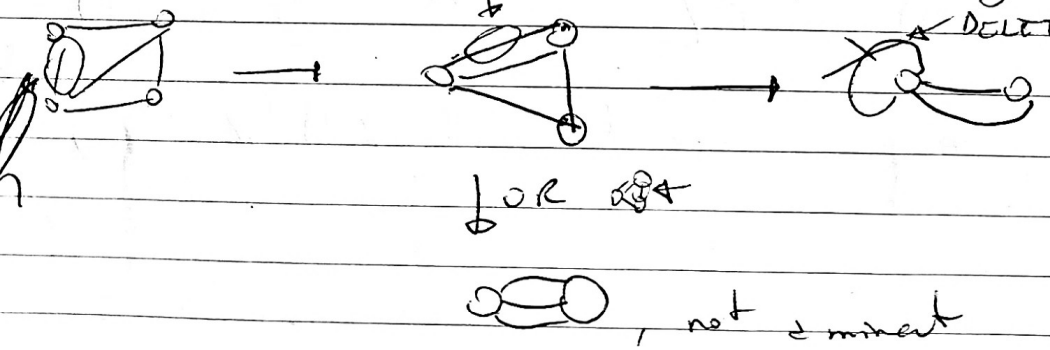
RCA:

while there are more than 2 vertices:

- Pick a remaining edge at random (u, v)
- merge (or "contract") u and v into single vertex
- remove self-loops (edges to itself and edges at same vertex)

return cut represented by final 2 vertices

Example

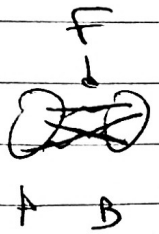


RCA sometimes identifies the min cut, sometimes doesn't.
Depends on choices it makes

The obvious question is then: Is this even a useful algorithm?

What is probability of success?

Fix $G=(V,E)$, n Vs, m Es. min cut (A,B), with k CE (call them F)



IF some edge of F is contracted (by sheer luck), RCA won't output A, B

IF RCA never contracts any edge of F, vertices of A and B stick together and we will get (A,B)

The correct output.

Thus $\Pr[\text{output is } (A,B)] = \Pr[\text{never contracts an edge of } F]$

Let S_i = event that edge of F is contracted in ith (ctr of $n-2$) iteration

In first iteration, we have probability k/m of screwing up.

Key observation: degree of each edge is at least k

$\text{total degree} = 2m \geq k \cdot n \rightarrow m \geq \frac{kn}{2}$

(every edge has 2 vertexes; # of edges)

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

$$Pr\left(\overline{S_1} \cap \overline{S_2}\right) = \underbrace{Pr\left[\overline{S_2} \mid \overline{S_1}\right]}_{1 - \frac{k}{\text{remaining edges}}} \cdot \underbrace{Pr\left[\overline{S_1}\right]}_{\geq \left(1 - \frac{2}{n}\right)} =$$

↑ resulting in terms of vertices, $m > \frac{k(n-1)}{2}$

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

$$P = \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{n-(n-3)}\right) \cdot$$

$$\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} =$$

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

↑ very low success rate, but better than random selecting between 2^n cuts

Solution: run RCA a large number of times, until you're sure you have found it.

How many times do we need to do this?

T_i = event that (A, B) is found on the ith try.

By definition different T_is are independent

So $P_r[\text{all } N \text{ trials fail}] = \prod_{i=1}^{N} P_r[\overline{T_i}] \leq \left(1 - \frac{1}{n^2}\right)^N$.

For all numbers $x \in \mathbb{R}$, $1 + x \leq e^x$

$\leq \left(e^{-\frac{1}{n^2}}\right)^N$ so for $N = n^2 \ln(n)$ $= \frac{1}{n}$

Running time is now polynomial! ($\Omega(n^2 m)$)

But can get big speedups to roughly $O(n^2)$ with more ideas

LESSON 10 WEEK 4

Number of minimum cuts for a graph.

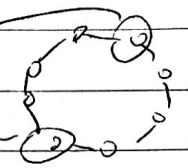
Question: what's the largest number of min cuts a graph with n vertices can have?

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

Why?

Lower Bound for max

Consider a cycle whatever you take the min cut



$\binom{n}{2} = \frac{n(n-1)}{2}$ possible min cuts

$T \geq \binom{n}{2}$

Upper Bound for max

Success probability: $\Pr[\text{output}(A;B)] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$

output $(A;B)$ disjoint

$\frac{T}{\binom{n}{2}} \leq 1$ at most all cuts are min cuts!

$T \leq \binom{n}{2}$

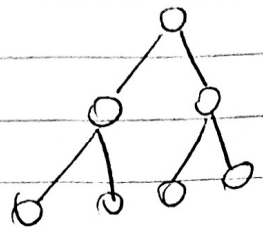
$T = \binom{n}{2} \Leftrightarrow \binom{n}{2} \leq T \leq \binom{n}{2}$

50

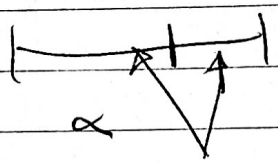
Problem Set Blotz

Controlling $\alpha - \beta$

delta not n, delta not $\binom{n}{2}$ or $(2^n - 2)$

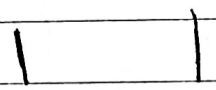


$$\frac{1}{n^2} \leq \frac{2}{n(n-1)}$$



median is this or here

α



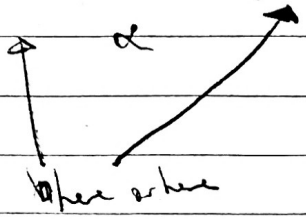
$(h-1-x)$ $\quad x$

$(1-x)$

$\alpha = 0.5$ $p = ?$

$p(h-1-x) \leq \alpha n$

$n - n \times 0 - 1$ [illegible] $x^2 \leq \alpha$



Here or here

every time

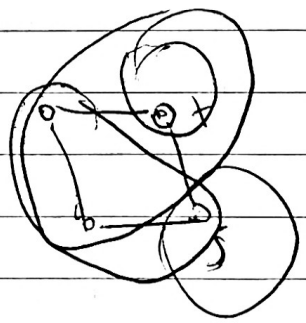
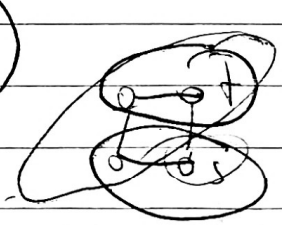
$$n\left(\alpha\right)^{d} = 1$$

size 1 = n

size 2 = $n - (1-\alpha)n = \alpha n$

$$n\left(\alpha\right)^{d} = 1$$

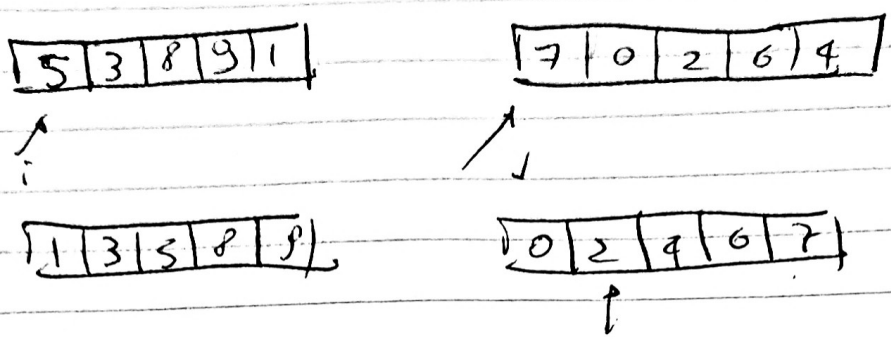
$$\log_{\alpha}(n) = d$$

graph  there are $\binom{n}{2}$ mincuts, of which 

$\frac{\binom{n}{2}}{2^n}$

52

Exam



$f(1)$ $g(1) \geq 1$

increasing

$f_n = O(g(n))$

$2^{f(n)} = O(2^{g(n)})$

$(2^n) = O(n)$

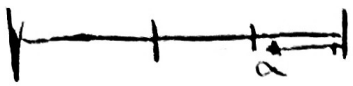
$2n = O(n)$

$2^{f(n)} =$

$2^{2n} = 4^n = O(2^n)$

$4^n \leq$

$0 < \alpha < .5$



$0,5 - \alpha$

p

(1-p) prob of fucking up,

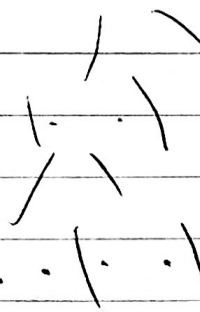
$$(1-p)^d = \epsilon$$

$$d \log(1-p) = \log \epsilon$$

$$d = \frac{\log \epsilon}{\log(1-p)} = \log_{1-p}(\epsilon)$$

$$\frac{\log_\epsilon(b)}{\log_\epsilon(a)} = \log_a(b)$$

$$\log_2(7) = \frac{\log 7}{\log 2}$$



$T_{(A)} \leq$

$a = 2$
$= 2$
$d =$

$a = 7$
$b = 2$
$d = 2$

$b^d = 4 < 2$
$(2^x) \; x \log_2(7)$