7/4/2018 Randamized Selection

Q: Get the nth smallest value from the array.

A: Pattern 1:
① Apply mergesort
② Return the nth element of sorted array.

Can we do it faster than $n \log n$?

⇒ No. Sorting cannot be faster than $n \log n$.

Then let' not use sorting

A: Pattern 2: Modify QuickSort (Randomized)

A: Pattern 3: Deterministic algorithm
(Choose pivot in deterministic way)

Quick ~~Sort~~ Select (Randomized Selection)

RSelect (array A, length n, order statistic 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 (left part of A, $j-1$, i)
⑤ If $j<i$ return RSelect (right part of A, $n-j$, $i-j$)

⇒ $O(n)$ in average.

Deterministic selection

⇒ Use additional linear time function to find right pivot (median of median) on top of ~~ra~~ quickselect. Add overhead but reduce the worst-case runtime a lot.

~~Choose Pivot (A, n)~~

DSelect (Array A, length n, order statistic i)

1. Break A into group of 5, sort each group
2. Make group C contains middle elements from each n/5 groups.
3. Do p = Select (C, n/5, n/10) (Continue recursively).
4. Partition A around p
5. If j = i return p
6. If j < i return Select (left part, j−1, i)
7. else if j > i return Select (right part, n−j, i−j)

※ Not as good as quick ~~so~~ select.

① Worse constants
② not-in-place

※ Comparision-based sorting algorithm cannot go faster than O(n log n).

e.g. Mergesort, Quicksort, Heapsort

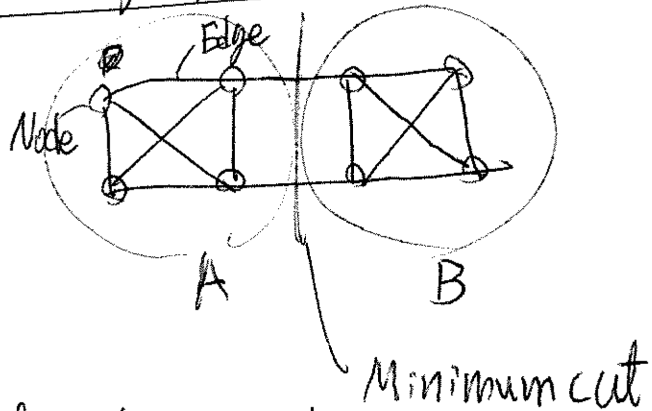
non-e.g. Bucket sort, Counting sort, Radix sort

Bucket sort: Good for data from distributions.

Counting sort: Good for small integer.

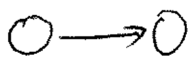
7/4/2018 Graph

- Cut of graph is useful



- Detect network weakpoints
- Detect community in socialnetwork
- Image segmentation.

- Directed node (Ordered)



- UnDirected node (Unordered)

- Sparse and Dense Graph

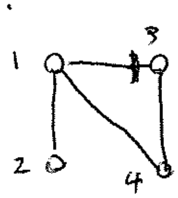
Let n = # of vertices, m = # of edges.

In "sparse graph" #m is $O(n)$ or close
In "Dense graph" #m is closer to $O(n^2)$
Most of the applications, m is $\Omega(n)$ and $O(n^2)$.

- Way to store graph structure

- Adjacency matrix $\theta(n^2)$
- Adjacency list $\theta(m+n)$

e.g.



Matrix

$$
n\left\{\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 1 & 1 & 1 \\
2 & 1 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 & 1 \\
4 & 1 & 0 & 1 & 0
\end{array}\right.
$$

~~[illegible]~~ space: $n \times n$ → Big for n^{10} size

List

Vertices $\underbrace{[[2,3,4],[1],[1,4],[1,3]]}_{n}$ → Good for graph search

Edges $\underbrace{[[1,2],[1,3],[1,4]]}_{m}$

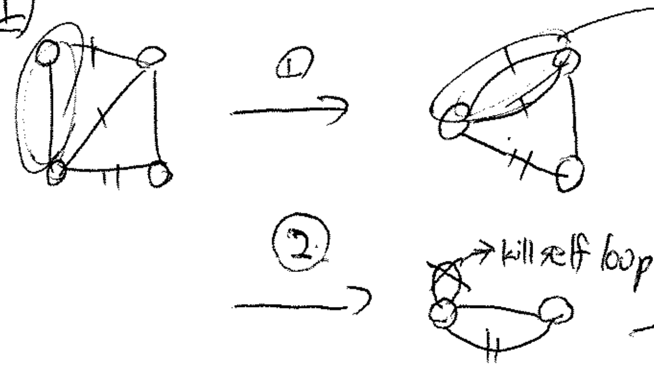
space: $n+m$ → Good for sparse graph

Random Contraction algorithm

※ Random strategy works for graph too!

Q: find the minimum cutting edge.

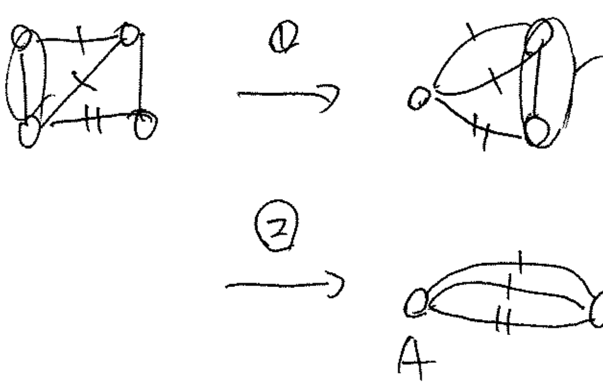
e.g. ①



Choose the edge randomly

→ Two vertices represent the two part that can be cut by min-cut edge

e.g. ②



random choice

→ Output is 3, which is not a min cut

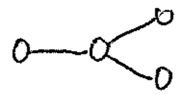
⇒ ~~Dec~~ Depends on the random choice, it end up in wrong answer.

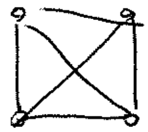
⇒ What is probability of success?

⇒ $\frac{1}{n^2}$ //

⇒ Too low? Not really. ~~Rea~~ bruteforth go up to $\frac{1}{2^n}$ ⇒ repeat $n^2 \cdot \log_2 n$ times and chance success will be $\frac{n-1}{n}$ //

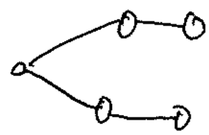
No. of edges between n no. of node

min : $n-1$  One edge per one node

max : $\underbrace{n}_{\text{All nodes}}\underbrace{(n-1)}_{\text{Connected to others}}\underbrace{/2}_{\text{No parallel}}$  All nodes have edges to others.

No. of minimum cuts

- Graph can have several min-cuts.

e.g. Tree graph with n nodes ~~han~~ has $n-1$ min-cuts. 

Question: What is the largest no. of min-cuts with graph of n nodes can have?

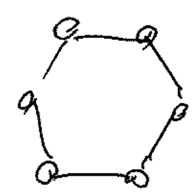
Answer: $\binom{n}{2} = {}_nC_2 = \frac{n(n-1)}{2}$ //

~~Whe~~

Why?

Lower bound: Cycle of n

Any pair of two edges will make min-cut.



has $\geq \binom{n}{2}$ min-cuts //

Upper bound:

Let $(A_1, B_1), (A_2, B_2) \cdots (A_t, B_t)$ be the min-cuts of a graph with n vertices.

$Pr[\text{Output} = (A_i, B_i)] \geq \frac{2}{n(n-1)} = \frac{1}{\binom{n}{2}}$ for all $i = 1, 2, \cdots t$

$\Rightarrow$ sum of the probabilities has to be at most 1.

$\Rightarrow \frac{t}{\binom{n}{2}} \leq 1 \Rightarrow t \leq \binom{n}{2}$ --- Upper bound of no. is $\binom{n}{2}$ //