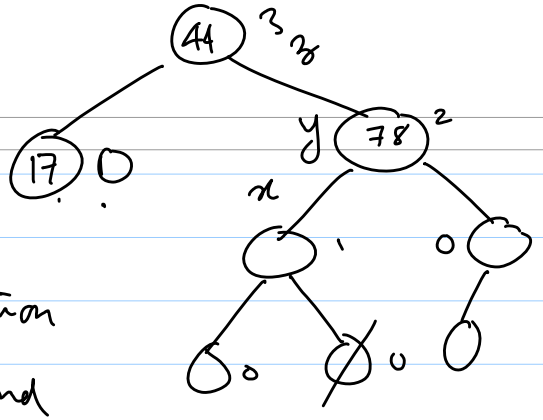
09.10.2018

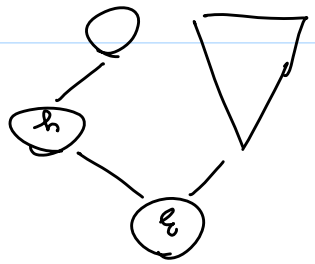
AVL Deletion

(not key by node)

- Delete node as per BST deletion
- Moving upwards from point of deletion update heights as required and find first point of imbalance if any.

- Use trinode restructuring to rebalance unbalanced node





Multiway search trees

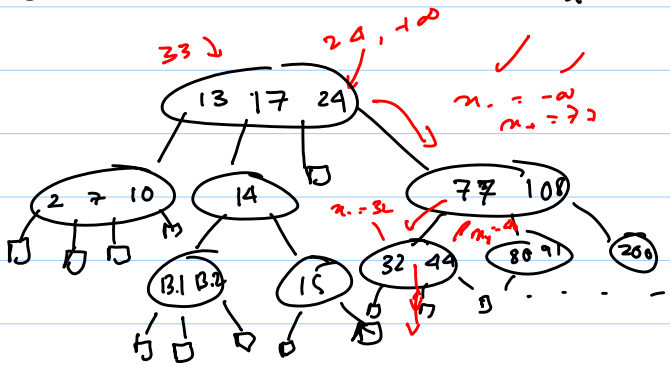
Def: A multiway search tree is one with foll prop

- Each internal node has at least 2 children
  - Each node contains a collection $k_1 \leq \dots \leq k_l$ of keys

- A node with l keys has $l+1$ subtrees assoc with it.
- All keys x in subtree i are st.

$$k_i \leq x \leq k_{i+1}$$

where $k_0 = -\infty$ and $k_{l+2} = +\infty$



Find 33

Find in multiway search tree

Root is $(k_1^r \ldots k_l^r)$, find (x)

Find $(x, \text{Dict}(k_1^r \ldots k_l^r))$

↳ Yes ✓

↳ NO $(x_-, x_{it}) \longrightarrow k_i^r, k_{i+1}^r$

$\hookrightarrow i \hookleftarrow$

Recursively continue search in subtree lying b/w k_i^r and k_{i+1}^r in subtree $i+1$.

12.10.2018

Multiway search trees restart

Def: We call a (general) tree node a d-node if it has d children.

Def: A multiway search tree node is a d-node ($d \geq 2$) that contains $d-1$ keys $k_1 \leq k_2 \leq \dots \leq k_{d-1}$

Def: A multiway search tree is a tree with the property that

- All leaves are 0-nodes
- If an internal node v is a d-node ($k_1, \dots k_{d-1}$) then it has d subtrees $T_{v,1} \dots T_{v,d}$ s.t.

$\max key(T_{v,1}) \leq k_1 < \min key(T_{v,2}) \leq \max key(T_{v,2}) \leq k_2 \dots$

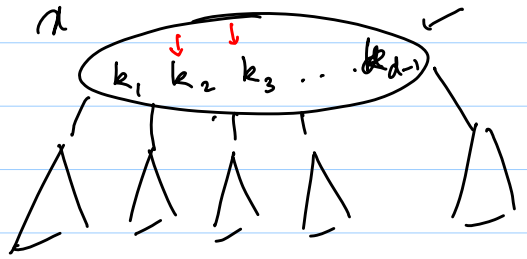
Find in a multiway search tree (x)

- Start from root
- At an internal node $v = (k_1 \dots k_{d-1})$ run a Dictionary-Search on $k_1 \dots k_{d-1}$.
  - If D-Search says found then return with key

Else D-search returns i st $1 \le i \le d$ such that if x exists it must lie in $T_{v,i}$.

Then continue with search $(T_{v,i}, x)$;

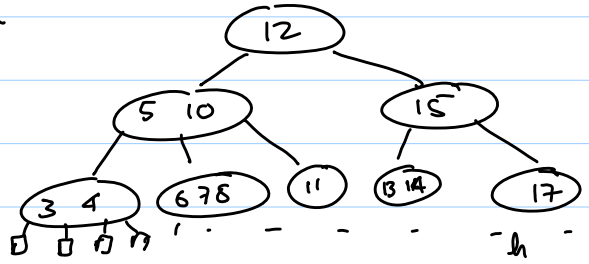
D-search returns x_-, x_+. If $x_- = -\infty$ then $i = 1$, if $x_+ = +\infty$ then $i = d$, else i is the index of x_+



2-4 Trees

Def: A 2-4 tree is a multiway search with d restricted to 0, 2, 3 & 4. Additionally, all leaves have the same depth

Q: Given n keys what is the height of a 2-4 tree.



If no of nodes at depth $d = n_d$

$2n_d \leq n_{d+1} \leq 4n_d$

$2^{d+1} n_0 \leq n_{d+1} \leq 4^{d+1} n_0$

Total no of nodes $= \sum_{i=0}^{h} n_i$

$\leq \sum_{i=0}^{h} 4^i n_0 = \frac{4^{h+1}-1}{3} \cdot n_0$

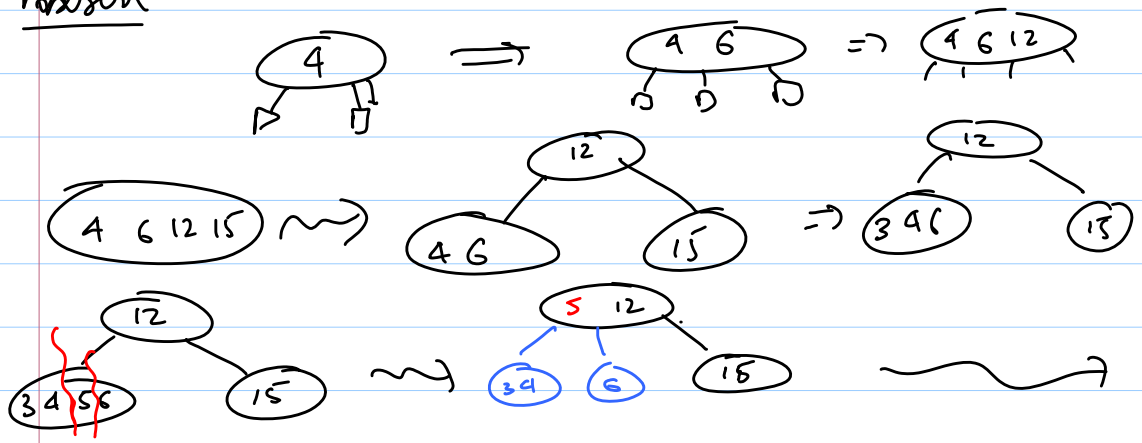
$= \frac{4^{h+1}-1}{3}$ $\Rightarrow$ Max keys $= 4^{h+1}-1$

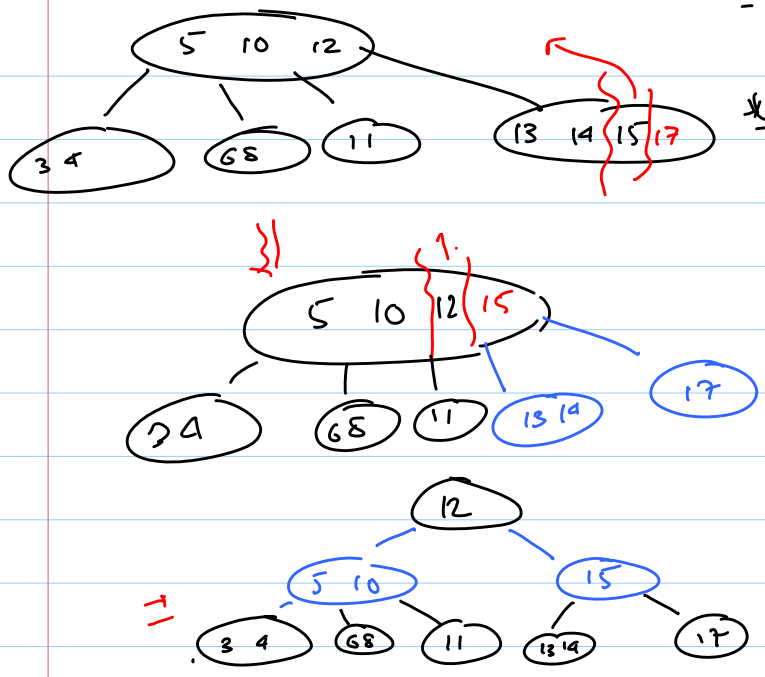
$\Rightarrow$ $\#\text{keys} \leq 4^{h+1} - 1$

and $\#\text{keys} \geq 2^{h+1} - 1$ ✓

$\log_4(n+1) \leq h \leq \log_2(n+1)$

Insert





Alg Insert (x)

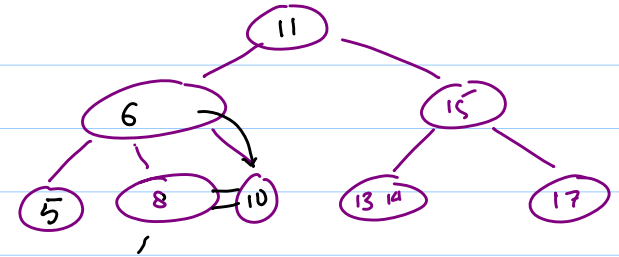
- Search (x) to reach a leaf
* Insert x in the parent of the leaf
  - ↳ If no overflow → stop
  - ↳ If overflow promote 2nd or 3rd key to parent and repeat from *
  - ↳ If no parent create new root.

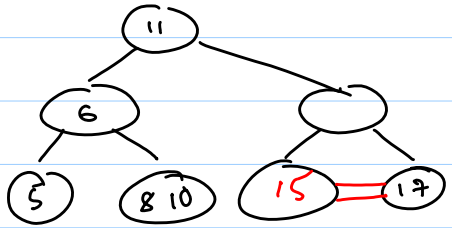
23.10.2018

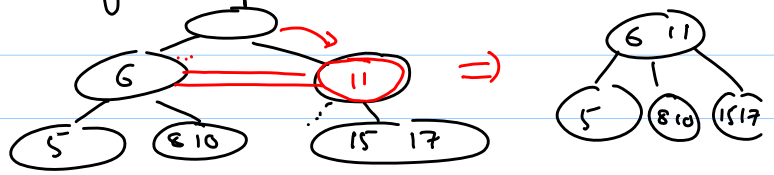
2-4 tree deletion

Alg:

- If key is not in lowest level replace it with appropriate key from lowest level ("inorder predecessor/succ") and do leaf-level deletion
- leaf-level deletion
  - Delete key. If no underflow stop, else



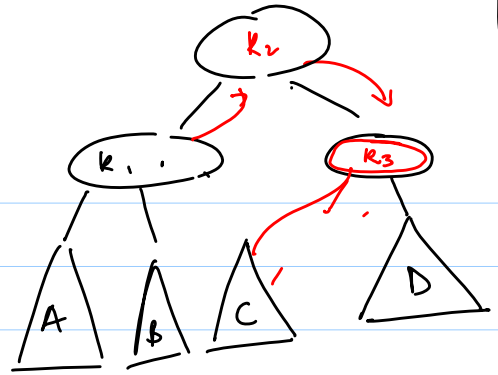




Generalization

→ (a, b) trees.

↳ B^+ Trees (self study)



$\log n!$

$= \sum_{i=1}^{n} \log i$

$\leq n \log n$

$\geq \sum_{i=n/2}^{n} \log \frac{n}{2}$

$= \frac{n}{2} \log \frac{n}{2}$

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

Red-black trees → Balanced BST.

Sorting:

Data $D = \{a, b, \ldots\}$

$S : D \xrightarrow{b_{ij}} \{1, \ldots n\} \Rightarrow x_1 \ldots x_n$

if $i < j \Rightarrow x_i \leq x_j$

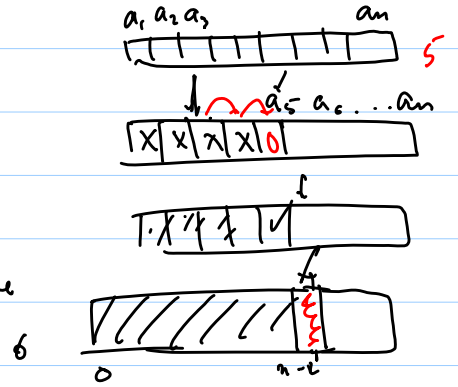
26.10.2018

Sorting is the process of finding an order respecting permutation of the input sequence => 1 (in the w.c)

sequence has to be found out of n! => $\Theta(n \log n)$ time in the worst case.

Algorithms already seen

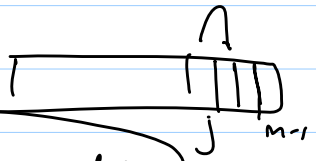
- Insertion sort → $\Theta(n^2)$ time
- Selection sort → $\Theta(n^2)$ time
- Heap sort → $\Theta(n \log n)$ time



Bubble sort

- There n rounds
- In each round start from A[0] and move through the array comparing A[i] to A[i+1] (i=0 to n-2)
  If they are in the wrong order, flip them.

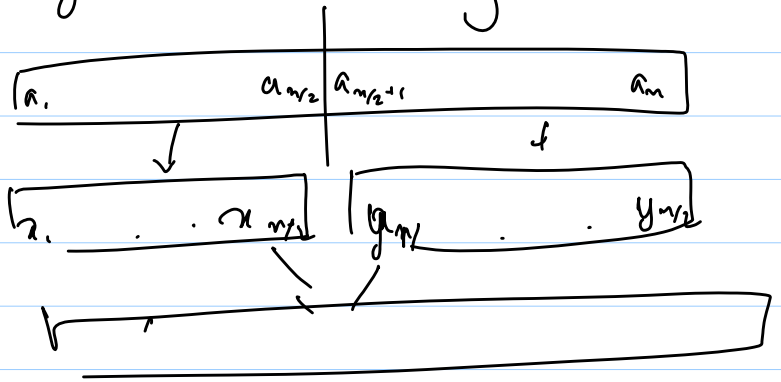
$\Theta(n^2)$ w/c Tm



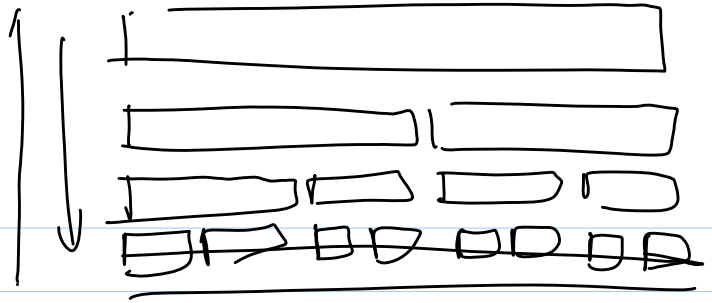
Round i ensures ith largest reaches its correct position

Merge sort

- If array size = 1 return array
- else
  - Divide array into 2 equal parts
  - Recursively merge sort each part
  - merge the two sorted arrays.



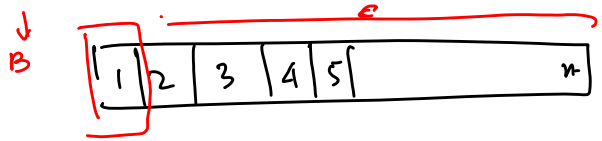
Merge sort
takes $O(n \log n)$ time



Quicksort

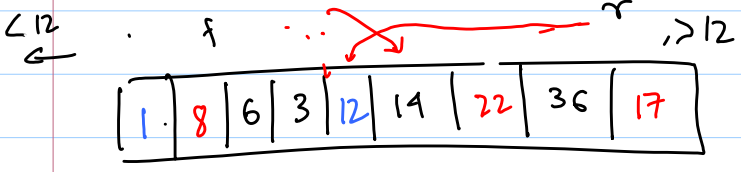
- Pick an element x (arbitrarily) from $A[\,]$
- Create two array $B[\,]$ and $C[\,]$ st $B[\,]$ contains elements $\leq x$ and $C[\,]$ contains elements $> x$.
- Recursively quicksort $B[\,]$ and $C[\,]$ and output as $B[\,] \; x \; C[\,]$.

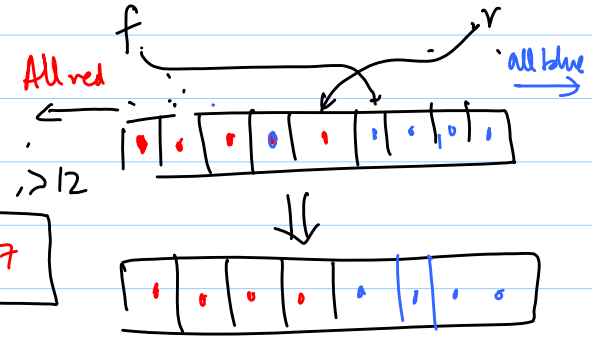
$T(n) = T(k_1) + T(k_2)$ + n where $k_1 + k_2 = n-1$



$T_n = T(n-1) + n$

$T(1) = 1$





~~qs(A)~~ , qs(A, 0, n-1)

qs(A, 0, 8)

pivoting

qs(A, 0, 3)

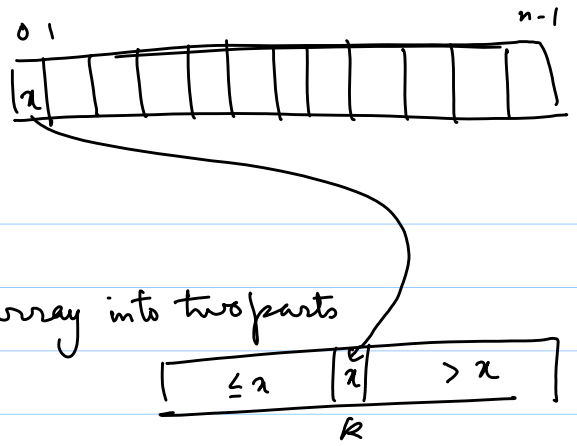
qs(A, 5, 8)

27.10.2018

Pivoting in-place:

A[0] is called the pivot

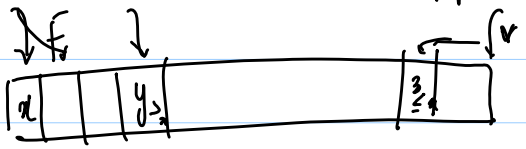
The process of recasting the array into two parts is called pivoting



Alg

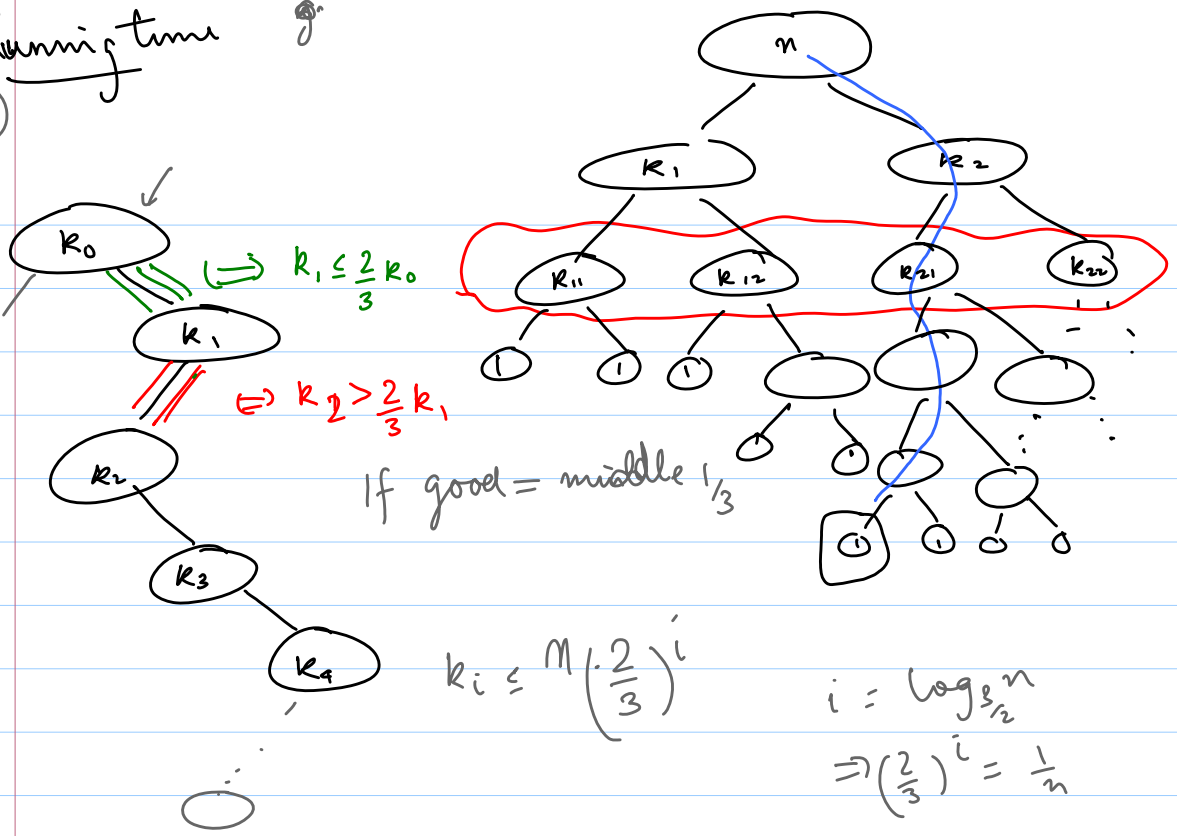
- Proceed with front and rear pointers (initialize to 0 & n-1 respectively)

Prop: A[0] ... A[f-1] ≤ x
A[r+1] ... A[n-1] > x



- Proceed to separate left {y ≤ x} & {y > x}
  - At then end put A[0] at the last location of the left array (y ≤ x)

Running time



$k_i \le n \left(\frac{2}{3}\right)^i$

$i = \log_{3/2} n$

$\Rightarrow \left(\frac{2}{3}\right)^i = \frac{1}{n}$

good parent – child #a relationship
= child #a gets $\leq 2/3$ of parents key.

good parent = (good #a relationship) & (good #b relationship)

Bottomline: Since in the real world keys are often randomly distributed QS works well in practice b/c if they are randomly distr then w.r.t. $= \Theta(n \log n)$.

30.10.2018

recurse(Arr, lower-end, upper end, rank)

$\frac{\log_{3/2} n}{\log_2 n} = \log_n \log_{3/2} 2$

$\log_{3/2} n = \frac{\log_n n}{\log_n 3/2}$

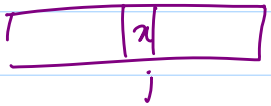
Quicksort 2.0?

- Instead of choosing A[0] as pivot, spend $\Theta(n)$ time to find the median of A[].
- Rest remains the same.

Randomized QuickSelect

Q: Given A[] find kth smallest element. $(0 \le k \le n-1)$

- Pick A[0] = x.
- Pivot A[0] around x & say x is now in A[j]
  - If $j > k$: recurse (A, 0, j, k)
    If $j < k$: recurse (A, j, n-1, k-j)

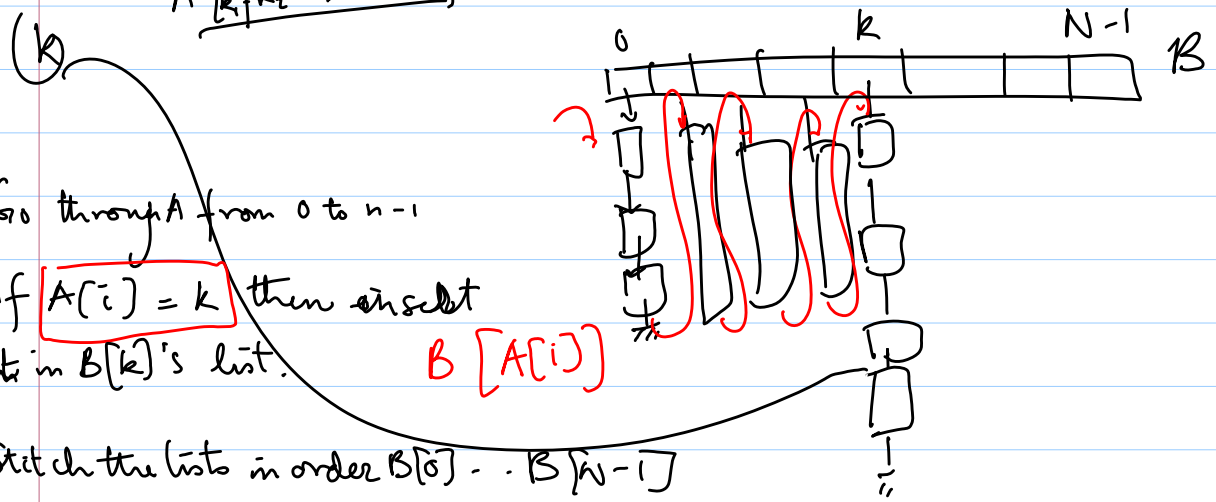


Bucket Sort

$n \log N$

Suppose we have n keys all belonging to $\{0, \ldots N-1\}$

A $[k_1 | k_2 | \ldots | k_n]$



- Go through A from 0 to n-1
- If $A[i] = k$ then insert it in $B[k]$'s list.
- Stitch the lists in order $B[0] \ldots B[N-1]$

Running time = $\Theta(n+N)$

01.11.2018

$1786 \leq 17862$ $\quad 02786 \geq 17862$

$$\begin{array}{r} {}^{1}{}^{1}7\,{}^{1}3\,6 \\ 2\,8\,4 \\ \hline 1\,0\,2\,0 \end{array}$$

Radix sort :

Def: Lexicographical ordering of "strings"

Suppose we have a totally ordered "alphabet" $\{a_1, a_2, \dots, a_k\}$

$(a_1 \leq_A a_2 \leq_A \dots \leq_A a_k)$

$x = x_1 x_2 \dots x_\ell$

$y = y_1 y_2 \dots y_\ell$ are strings on our alphabet

We say that $x \leq_L y$ if either $x_1 <_A y_1$

or $\{ x_1 =_1 y_1$ and $x_2 \dots x_\ell \leq_L y_2 \dots y_t \}$

λ = empty string : $\lambda <_L$ non empty string

Stable sorting

Given $(x_1, y_1), (x_2, y_2) \ldots \ldots (x_n, y_n)$

Sort on first coordinate to get

$(x_1', y_1'), (x_2', y_2') \ldots \ldots (x_n', y_n')$

Suppose $x_i', x_{i+1}' \ldots x_j' = x_k$

The sort is said to be stable if the pairs

$(x_k, y_{i_1}), (x_k, y_{i_2}) \ldots (x_k, y_{i_j})$

appear in the same order in the sorted as in the unsorted sequence.

(7, 14), (6, 2), (7, 3), (5, 17), (7, 1),

(8, 26)

Not stably sorted → (5,17), (6, 2), (7, 3), (7, 14), (7,1), (8,26)

Stably sorted → (5,17), (6,2), (7, 14), (7, 3), (7,1), (8,26)

Insertion sort

| (6,2) | (7,14) | (7,3) |

Radix sort

273, 078, 932, 017, 645, 368

932	→	017	017
273		932	078
645		645	273
017		368	368
078		273	645
368		078	932

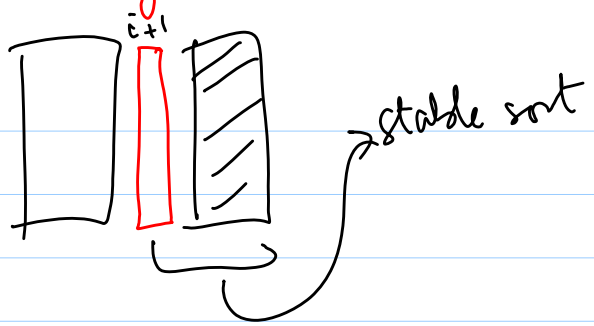
Given d digit numbers with columns d to 1 . . . 1.

From i = 1 to d

stably sort the current array by the ith column

$O(dn)$

Correctness by induction



IH: After i rounds the numbers are sorted if we only look the i least significant digits.

IS: Sort column i+1 (stable). Now group the numbers by the elements of the alphabet. Within a block the nos are sorted by IH & stability. Across blocks by correctness of your bucket sort □

2.11.2018

simple graph : $(a,a) \notin E \ \forall a \in V$.
undirected graph : $(a,b) \in E \Rightarrow (b,a) \in E \ \forall a \neq b \in V$
Multigraph : E is a multiset

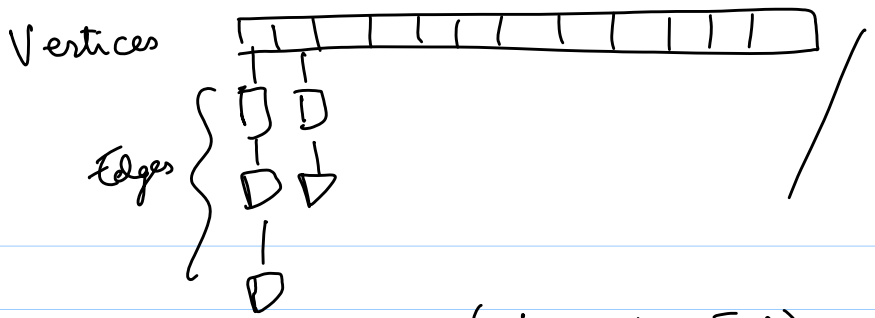
The Graph ADT

Defⁿ : A graph is a tuple (V, E) where V is a set of "something" and E is a sub"set" of $V \times V$.

Twitter
- V – set of users
- E – set of (follower, followed)

FB
- V – set of users
- E – set of friend pairs

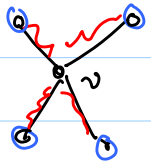
Collab
- V – set of authors of research papers
- E : $(u,v) \in E$ if u & v published together



A- Graph ADT methods (V is of ADT A)

Graph operations
- vertices() — A-List
- edges() - A x A List

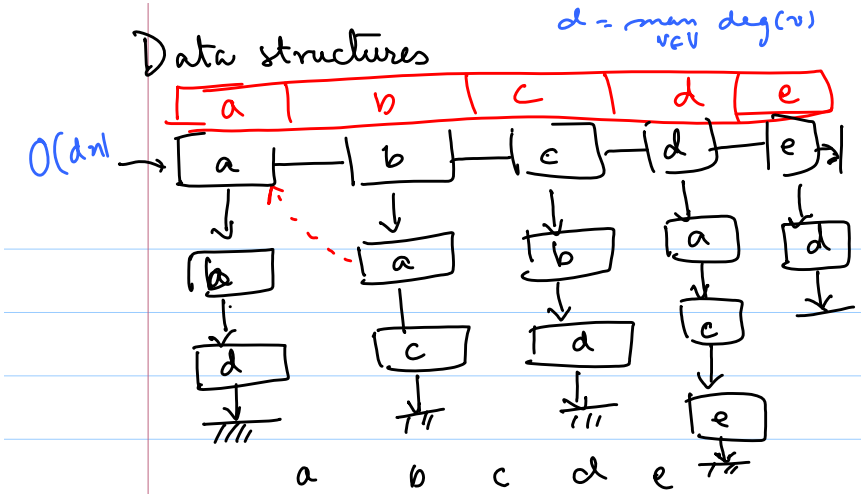
Vertex operations
- adjacent edges() - A x A List
- adjacent vertices() - A List

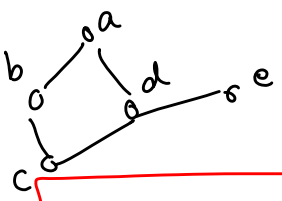


Edge operations + Dynamic operations (insert/delete etc)

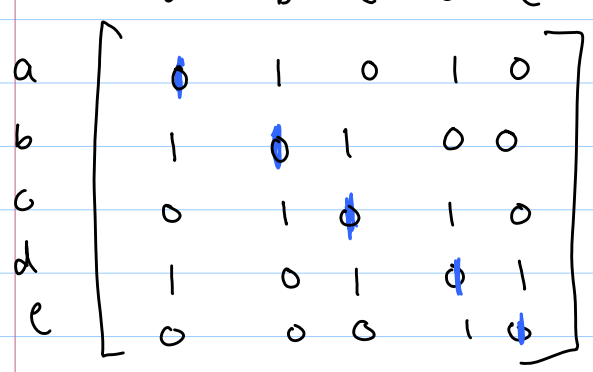
Data structures

$d = \max_{v \in V} \deg(v)$





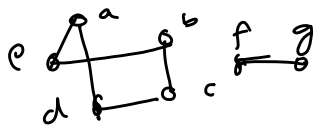
Adjacency list



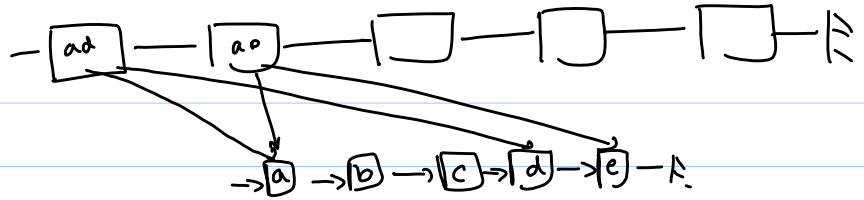
Adjacency matrix

6.11.2018

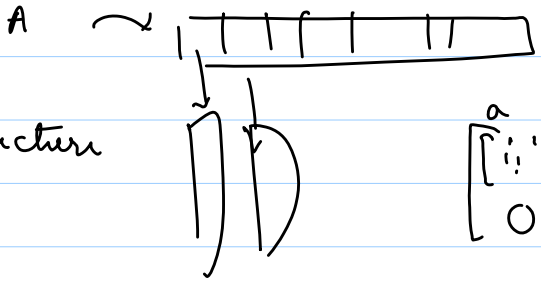
Neo4J, Titan GraphDB



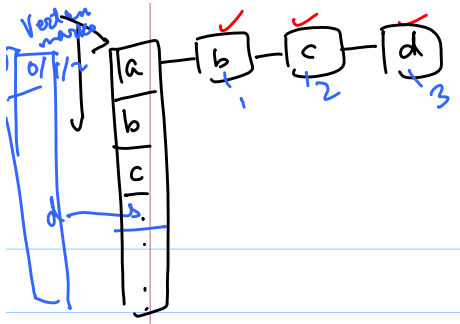
Edge list

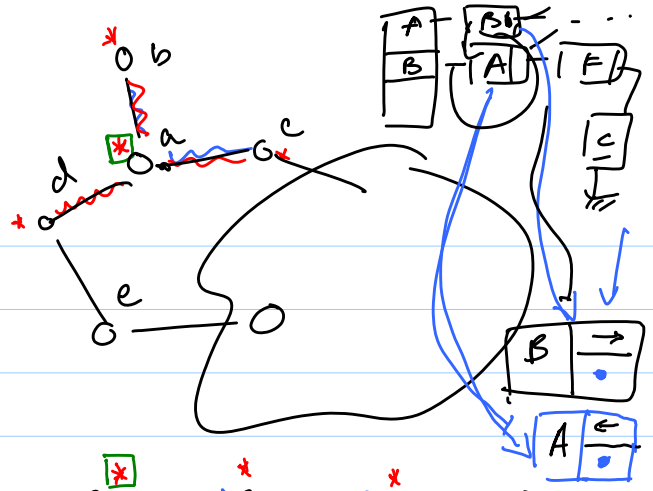


Graph traversal



- Vertex based data structure
  Adj list/Matrix





0 — ○ ✱ — u_1 u_2 u_3 — —

1 — ✱ ✱ u_1 u_2 u_3 . . .

2 — ✱

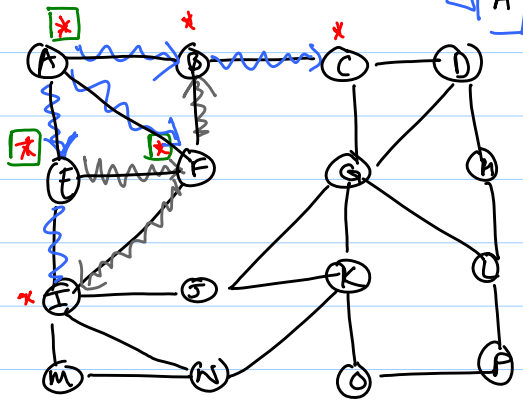
Breadth First Search (Trav. w/ a Q)

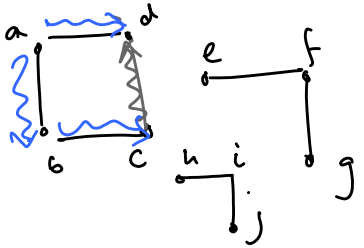
Tree edge

Back edge

I C

B - Store newly discovered vertices in queue.



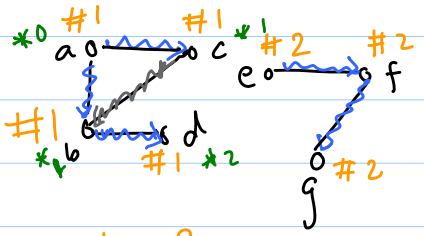


Running time : $\mathcal{O}(n+m)$

where $n = |V|$, $m = |E|$

13.11.2018

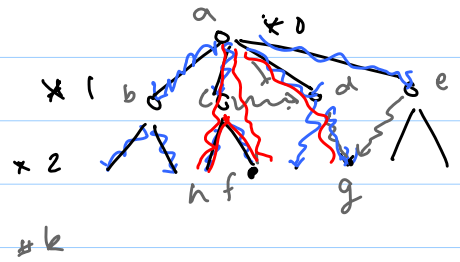
Finding connected components using BFS $\Rightarrow$



Is $u \leftrightarrow v$? Ans :: is $\#u = \#v$?

⇝ — Cross edges

→ Same level or to lower level



Properties of BFS

Also DFS:

- BFS visits all nodes of the graph
- Discovery edges form a spanning tree of G.

  Given $u \hookrightarrow v$, $u \rightsquigarrow a$ and $v \rightsquigarrow a \Rightarrow u \rightsquigarrow v$

  where a is the vertex (of the comp containing u & v where BFS starts)

Only BFS:

- For each vertex v at level i for a BFS started at s each path from s to i has $\geq i$ edges and the tree path has i edges

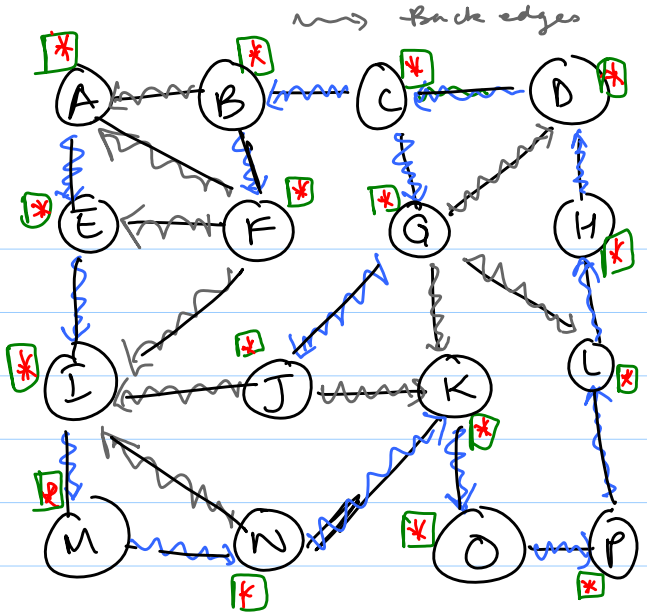
$G = (V, E)$

$G' = (V, E')$ is a spanning tree of G if $E' \subseteq E$

Σ $u \hookrightarrow v$ in G $\Rightarrow u \hookrightarrow v$ in G' and G' is a tree

Depth First Search

Every opposing endpoint of the edge being processed is in a stack



Syllabus ⊆ Webpage

15.11.2018

Weighted Graphs

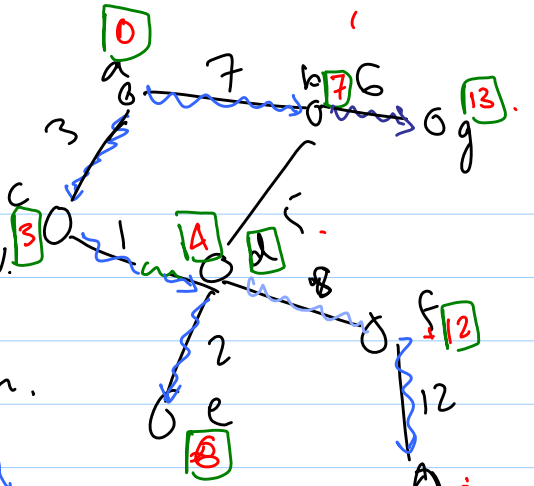
Single source

Shortest paths: Dijkstra's algorithm.

Find shortest path from a to v $\forall v \in V$.

Assume

$w : E \to \mathbb{R}_+$ is the weight fn.



Alg initial

- Put all estimates in a heap — $\Theta(n)$

→ Delete min from heap — $O(\log n)$

- Finalize deleted vertex's distance

- Update its neighbours — $\Theta(\log n)$ per neighbour

Time = $O((n+m)\log n) = O((|V|+|E|)\log|V|$