Recitation 12

Overhead:
- Quiz

mean	90.77
median	94
mode	94
max	112

- check your grades match!

Agenda
- radix
- graph: configuration
- BFS + DFS
- Quiz

RADIX sort

problem: counting sort performs badly for large input range
solution: perform a series of counting sorts on small input ranges
- perform a series: go from least significant to most significant.
- counting sorts: stable!
- input ranges: digits, the "radix"

running time = (# of rounds of sorting)(# inputs + input range)

ex: degenerate (counting sort) = $1(n+k)$
decimal w/ d-digits = $d(n+10)$
words w/ 8 letters = $8(n+26)$
b-bit binary #, radix r = $\frac{b}{r}(n+2^r)$
↑ # bits to consider each time.

- best choice of r? $r \le \lfloor \lg(n) \rfloor \rightarrow n+2^r = n+2^{\lg n} = 2n$.

Bucket Sort

idea: split input into "buckets" with few elements each. Then sort.
- into "buckets": like counting sort! but for more than integers.
- few elements: E(# elts) = 1, need to know distribution!
- Then sort.: insertion good for small

it's like generalized counting sort.

GRAPH SEARCHING

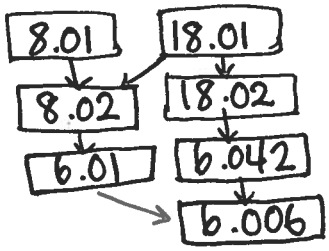
→ configuration graph

vertex : a possible configuration
edges : transition between configurations.

ex: games - vertex = game state
edge = move.

prerequisites graph - vertex = class
edge = prereq.

(hopefully no loops!)



→ graphs vs. trees

tree = a graph with no directed cycles.
- can choose any node as "root" then define an unambiguous parent/child hierarchy.

graph = anything goes.

→ path : sequence of nodes $x_1 \to x_2 \to x_3 \to \cdots x_k$ s.t. $(x_i, x_{i+1}) \in E$ $\forall$

→ reachable
1. x reachable from x (identity)
2. u R x and u → v then v R x (transitive)
3. directed: NOT reflexive
undirected: reflexive xRy then yRx.

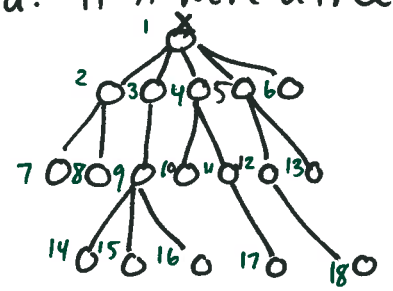
Thm u reachable from x then ∃ path from x to u.

[who cares? well if someone swaps stickers on your rubiks cube, the sdn may not be reachable.
or if you make a wrong move in a game a winning config may not be reachable.

GOAL: traverse all nodes reachable from X.

Breadth First Search

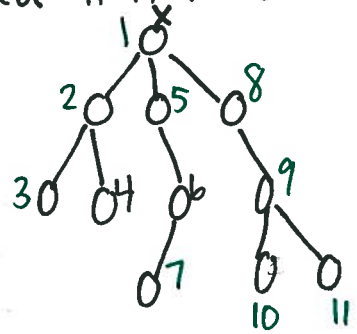
idea: if it were a tree we would traverse the breadth first.
i.e. all nodes in level i done before level i+1



- finds shortest paths!
- use a FIFO queue

Depth First Search

idea: if it were a tree would traverse the depth first.

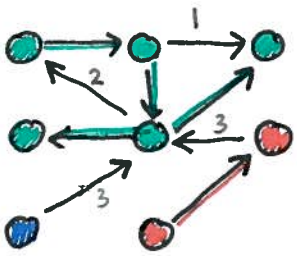


- uses LIFO queue (a stack)
- less memory requirements.

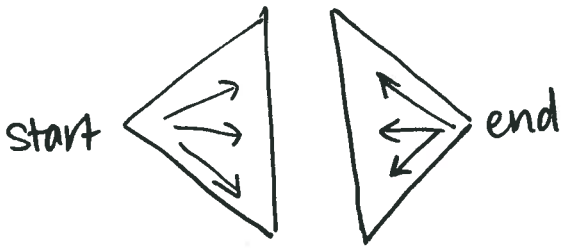
GOAL 2:
- classify nodes into trees and edges into types.
  - perform DFS on all nodes.

↳ tree (colored)
non-tree
1 forward
2 backward
3 cross
4 self.

ex



Bidirectional path search.
Given start + end node: use BFS on both simultaneously to find shortest path from start to end.



better than DFS: finds shortest
BFS: quicker.

WARMUP Answers

a. Give the running time to sort n phone numbers using (i) RADIX (2) counting

(i) radix — digits = 10, base = 10 — xxx-xxx-xxxx

$d(n+b) = 10(n+10)$

(ii) counting

$(n+b^d) = n+10^{10}$

b. RADIX sort the following.

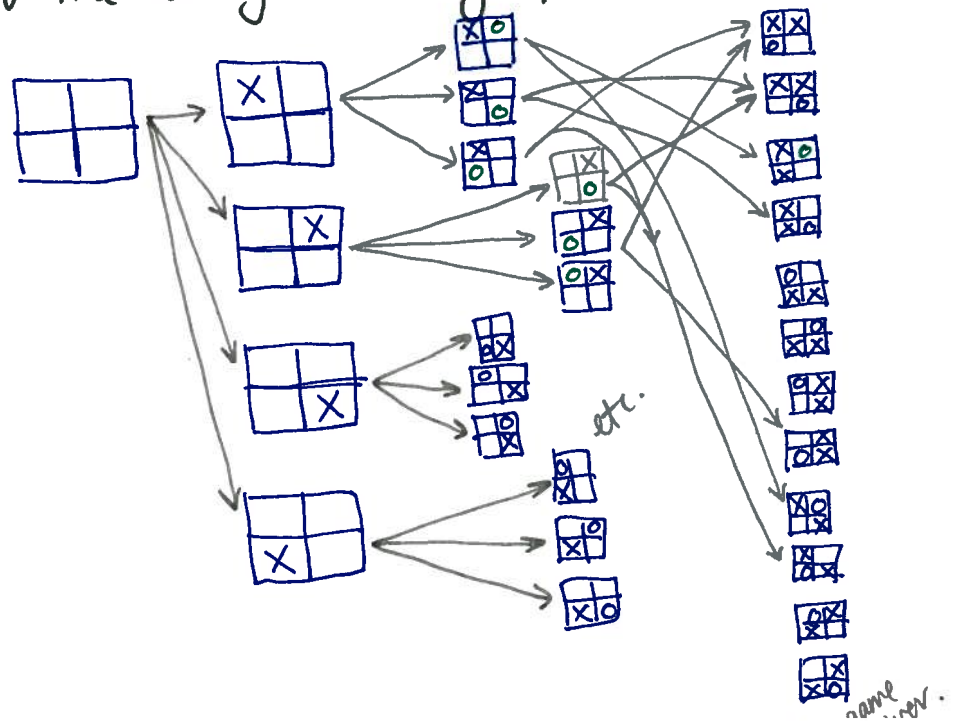
input

TAP 4	D I ^{3}E	T^{5}A P	C A T
C A T 5	D O^{4}G	C^1 A T	D I E
D O G 2	L O^5 L	D^2 I E	D O G
L O L 3	T A^1 P	D^3 O G	L O L
D I E 1	C A^2 T	L^4 O L	T A P

c. What other sorts could you use in RADIX other than Counting sort?

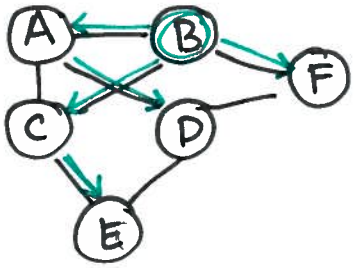
any stable sort
- insertion
- merge
- ...

d. Draw the configuration graph for 2x2 tic-tac-toe i.e. x/o ...



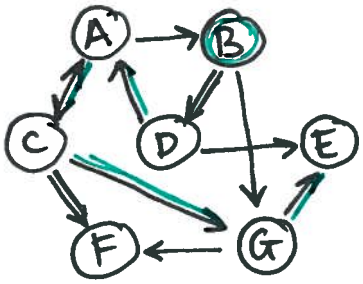
see how even a simple game has a "complex" graph.

e. Traverse the graph using Breadth first Search. starting @ B



R	Q
B	B
B A C F	A C F
B A C F D	C F D
B A C F D E	F D E
"	D E
"	E

f. same DFS

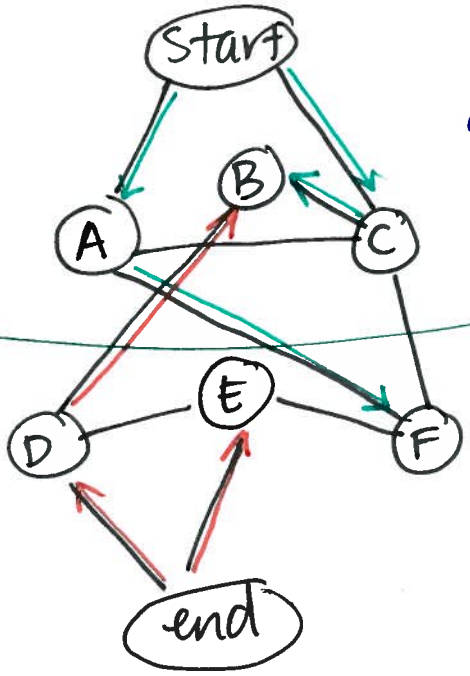


R	Q
B D A C F G E	B
	D G
	A E G
	C E G
	F G E
	G E
	E

g. find a path from start to end using BFS on both simultaneously (one step from start side, then one step from end side . . . .)

start

	Q	R
0:	start	start
1:	A C	start A C
3:	C F	start A C F
5:	F	start A C F B



end

	Q	R
0:	end	end
2:	D E	end D E
4:	E B	end D E B

first intersection so stop!

- try doing it purely from start to see how many more steps it takes!