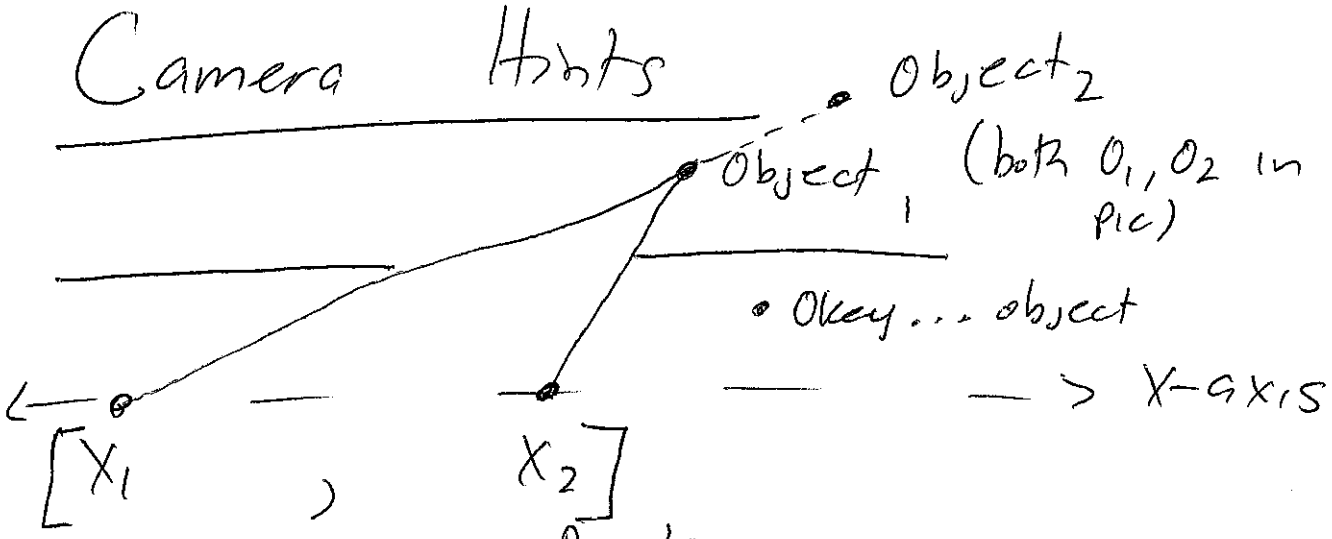
Camera Hints



Pic = 4

Client #1 [-15, 12] ✓

Client #2 [-10, 8] ✓

Client #3 [3, 16] ✓

Scheduling problem

To compare equality btw double a, b

Math.abs(a-b) < = 1e-9

BFS Applications

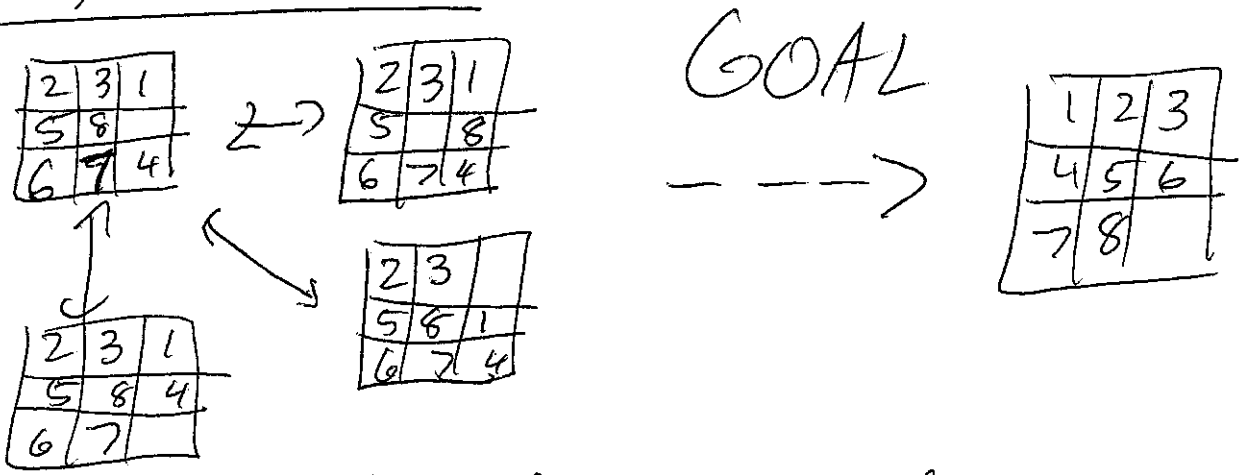
Typical use: find shortest distance (min # edges to traverse) from a starting vertex to all others in an undirected unweighted graph.

Two Different Application

(1) Eight Puzzle

(2) Rush Hour

Eight Puzzle



What is the fewest # of moves necessary to solve a given puzzle?

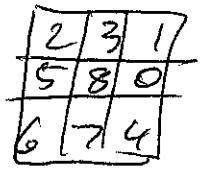
(1) Each board position is a vertex

(2) We connect 2 board positions (vertices) if we can reach one from the other in a single move.

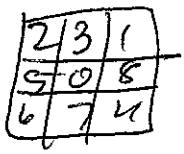
DON'T NEED TO STORE THE WHOLE GRAPH!!!

HashMap<BOARD, Integer> distances

Queue<BOARD> q



→ 0

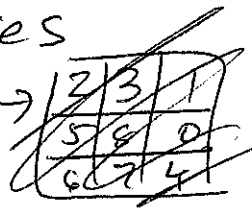


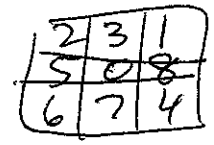
→ 1

BOARD POSITION

$\frac{9!}{2}\left(\begin{matrix}300{,}000 \\ 400{,}000\end{matrix}\right)$

#edges ≤ 2×9!





enqueue

Two ideas to store board for HM, Q:

(1) String "231580674" } either is fine

(2) Integer 231580674 }

2 TYPES OF STORAGE

(1) 2D array → good making moves

(2) String/Integer → good HM, Q

We need 2 functions

(1) 2D array → String/Integer compress/encrypt

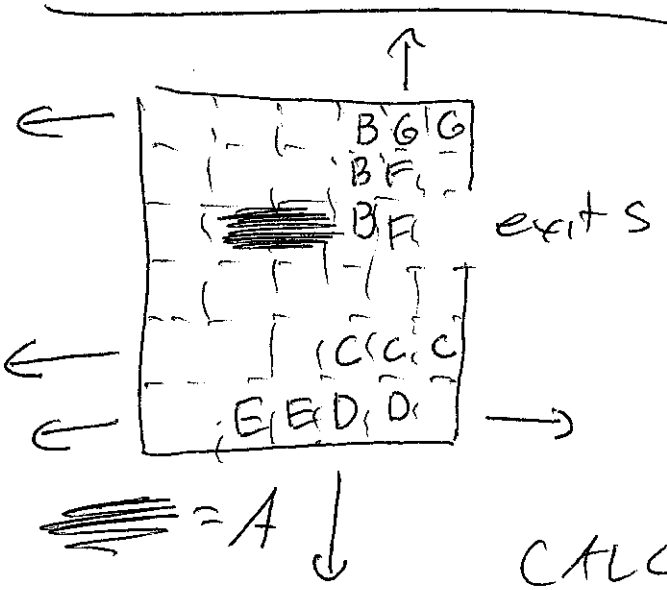
(2) String/Integer → 2D array expand/decrypt

Solve 1 puzzle ≈ 1,000,000 moves.

What about solving 10,000 puzzles? $10^4 \times 10^6 = 10^{10}$ BAD.

TRICK: RUN ONE BFS FROM END BOARD TO ALL OTHERS SINCE MOVES ARE SYMMETRIC!

Rush Hour



1 MOVE - SLIDING A CAR AS MANY SQUARES AS YOU CAN IN A SINGLE DIRECTION

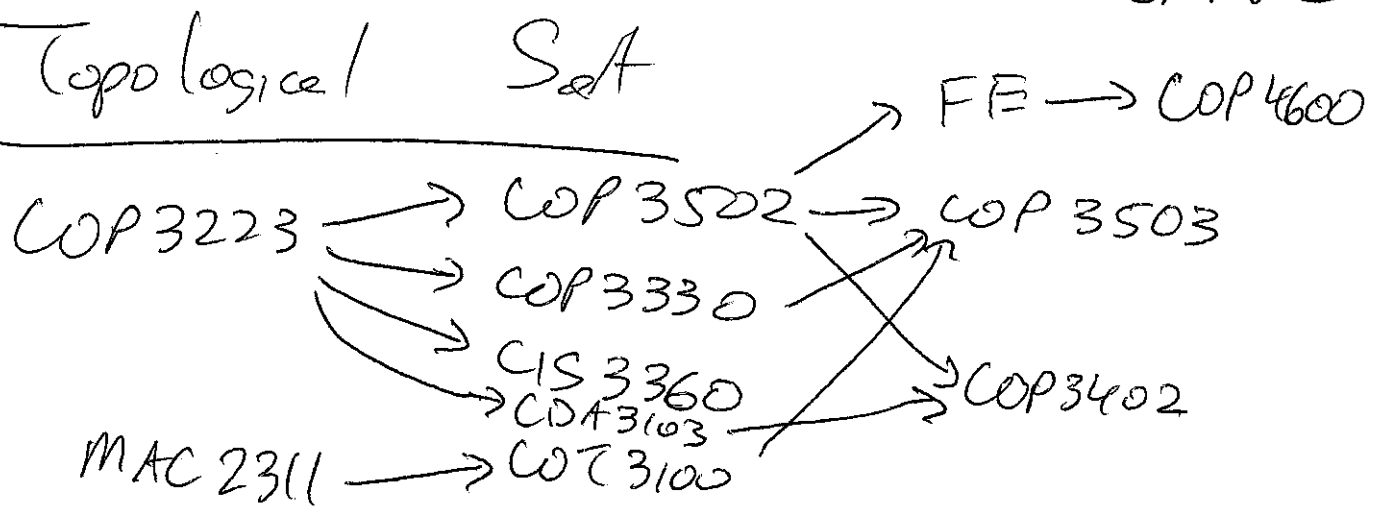
CALC FEWEST # MOVES TO FREE YOUR CAR

(1) each board position is a vertex

(2) we connect 2 board positions if we can obtain one from the other in a single move.

A = left 1, B = down 1, C = left 1, 2, 3
D = right 1, E = left 1, F = down 1
G = no moves

Topological Sort



Provide a valid ordering of classes

Iterative Algorithm

(1) Calculate the in degree of each vertex

COP3223	MAC2311	COP3502	3330	3360	3103	3100	FE	3503	3402	4600
0	0	1	1	1	1	1	1	3	2	1

(2) Create a queue of vertices. Add to that queue all vertices of degree 0.

Q → 3223 → 2311

(3) Create an array to store result

(4) for (i = 0; i < n; i++) // loop through vertices num times

(a) if the queue empty, return false/null

(b) res[i] = queue.poll() (Add the next class from queue to your schedule)

(c) Update all indegrees of edges leaving res[i]. If any of these hit 0, add to the queue. 3/1/18 (6)

(5) If you get here return res.

	COP3223	2311	3502	3330	3360	3103	3100	FE	3503	3402	4600
Sched	0	0	1	1	1	1	1	1	3	2	1
3223	0	0	0	0	0	0	1	1	3	2	1
2311	0	0	0	0	0	0	0	1	3	2	1
3502	0	0	0	0	0	0	0	0	2	1	1
3360	0	0	0	0	0	0	0	0	2	1	1
3103	0	0	0	0	0	0	0	0	2	0	1
3330	0	0	0	0	0	0	0	0	1	0	1
3100	0	0	0	0	0	0	0	0	0	0	1
FE	0	0	0	0	0	0	0	0	0	0	0

3402
3503
4600

Q → ~~3223~~ → ~~2311~~ → ~~3502~~ → ~~4600~~ 3360 → ~~3103~~ → ~~3330~~ → ~~3100~~ → FE

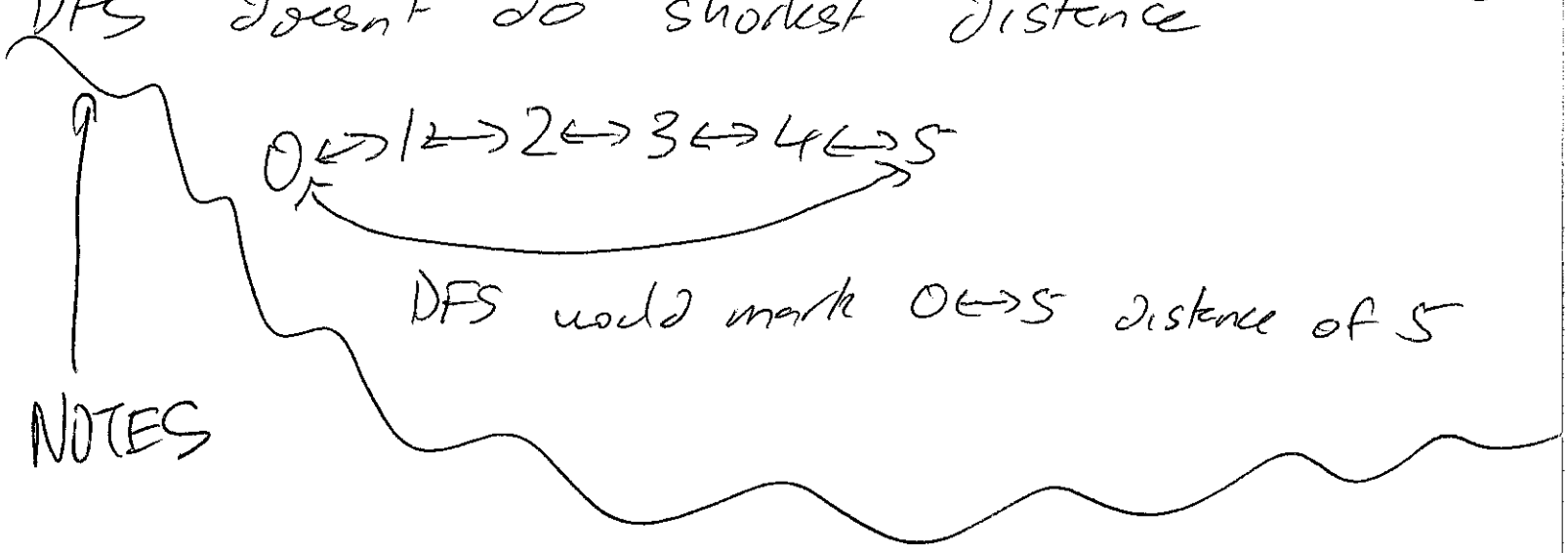
3402

4600

~~FE~~ ← 3503

3/1/18 (1)

DFS doesn't do shortest distance



NOTES