- Sprint 7: "email"
- Project team & future forms

Search graph

DFS
BFS — optimal # "hops"

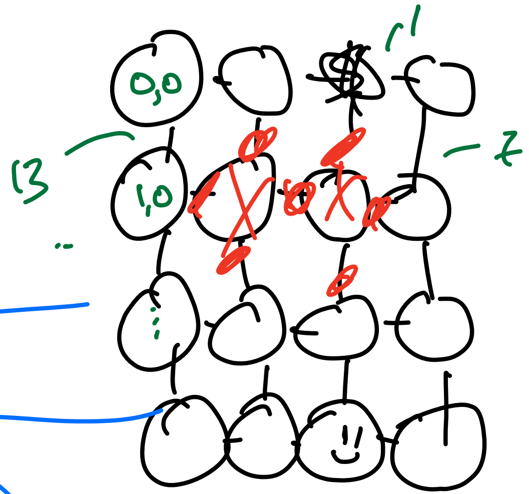
Dijkstra's — optimal total cost

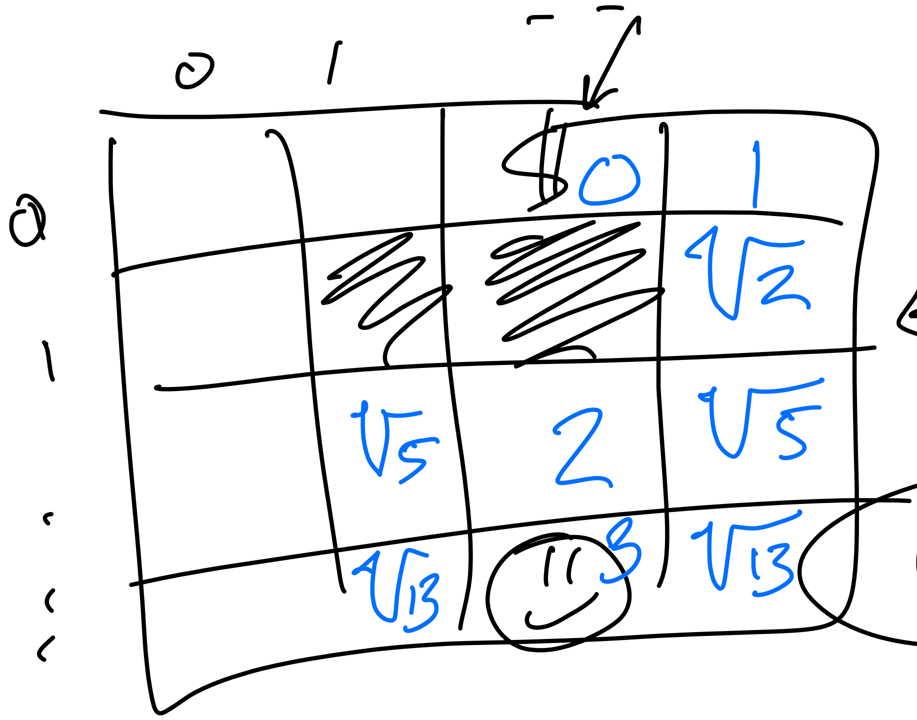
	0	1	2	3
0	5		$ 5	4
1	4	▨	▨	3
2	3	2	1	2
3	2	1	☺	1

3	2	1	2
4	3	2	3

...

Wasted effort





← ignore edge weight

d[n, g]

Greedy Best-First Search

NOT OPTIMAL

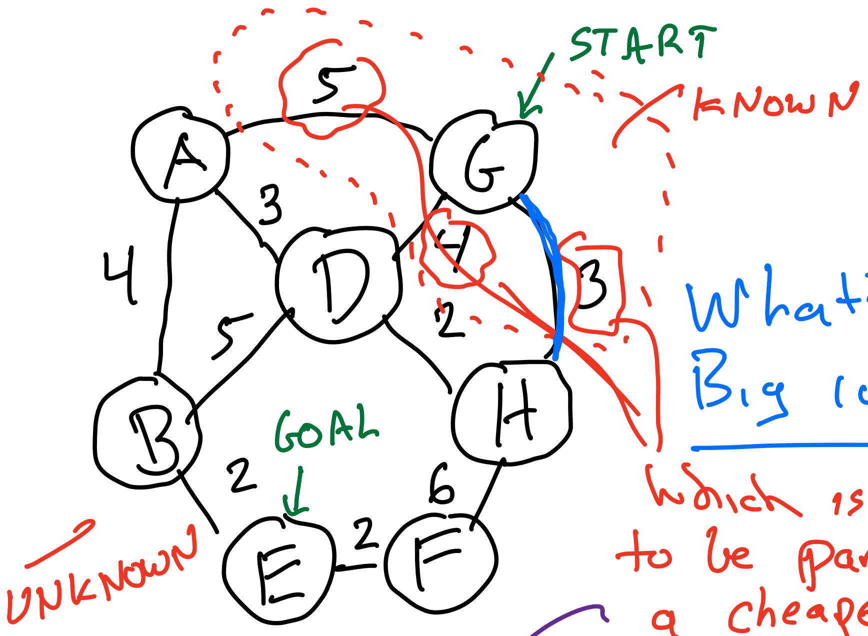
oakland
START
S.F.
GOAL

I am not a good artist :)

Dijkstra's Algorithm

BFS with a "priority queue"

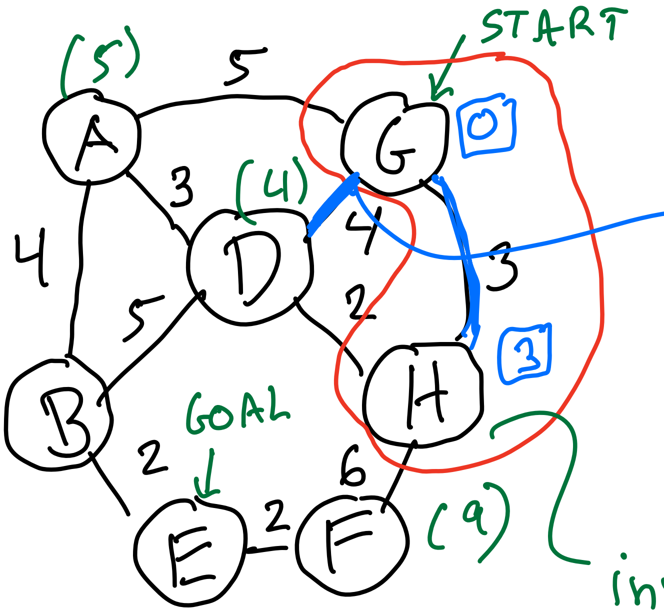
↳ Explore the node that costs the least to reach from already-explored territory.



What's the Big idea?

which is guaranteed to be part of a cheapest path somewhere

No negative edge weights

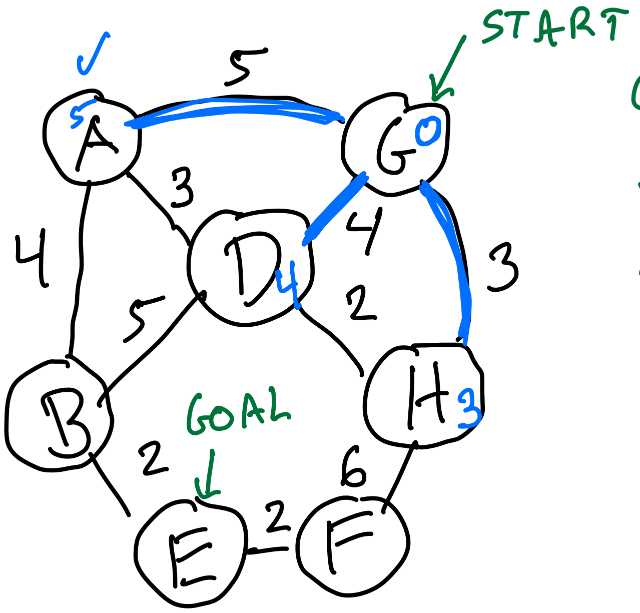


which edge do we add next?

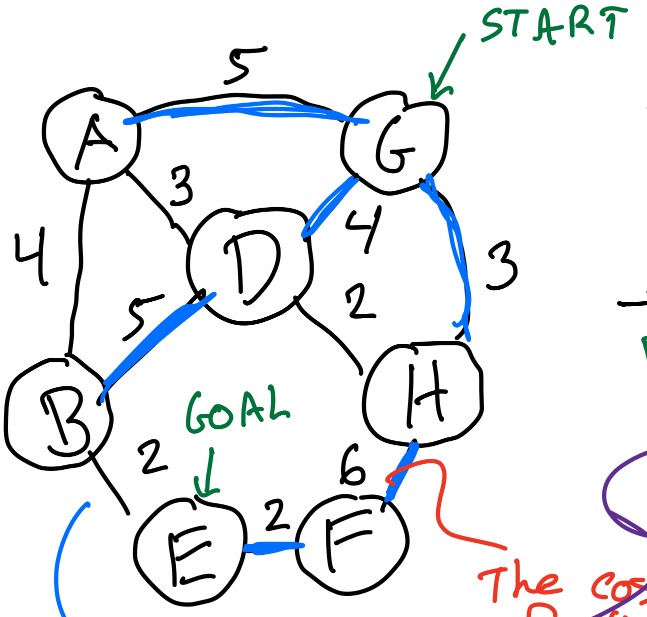
part of cheapest path somewhere

invariant:

cheapest path tree



G → 0 ✓
H → 3 ✓
D → 4 ✓
A → 5 ✓
F → 9
B → 9



G 0
H 3
D 4
A 5

B → 9 ✓
F → 9 ✓
E → 11 ✓

upper bound on ~~total~~ cost to this node

The cost of going 1 step farther

could have chosen this edge, too

where do these #s come from?

UN explored

$\text{dist}[n] \leq \text{dist}[m] + w$ ← GBFS gradient

$+ h(n, g)$

$m \xrightarrow{w} n$

explored

Is n a good choice globally

Dijkstra's inequality

① What if the graph is too big to load into memory?

② How can we take "real distance" into account? ✓ A*

③ How will Copilot do?

④ BTW: Unit testing? ✓ PBT