EX. 1

DAG - directed acyclic graph

A directed graph G is a DAG

$\Leftrightarrow$ G has a topological sort

$\Leftrightarrow$ For any edge $e=(u,v)$ in G, we will have $u.finish > v.finish$ after DFS

If there exist a cycle:

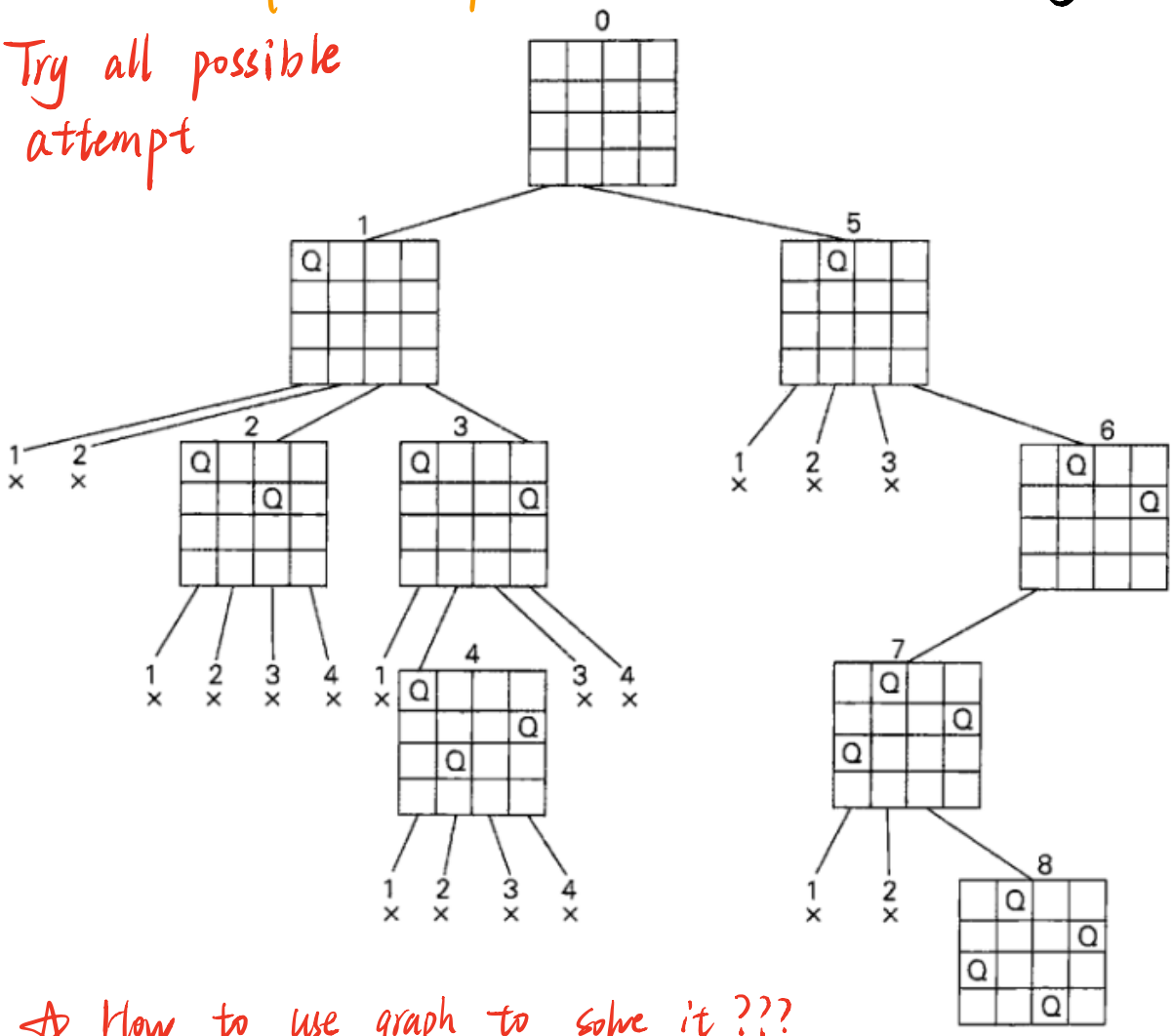
suppose DFS reach v first, we will have $v.finish > u.finish$

(contradiction)

Ex. 2

(2) n-queens problem — back tracking

Try all possible attempt



☆ How to use graph to solve it ???

One solution :

- Vertices : All possible positions (row, col)
- Edges : All possible attempts $((row_1, col_1), (row_2, col_2))$, $col_1 \neq col_2$

Try possible attempt from the top, label visited position and check the validity.

(22) Travelling salesman problem — branch-and-bound

Use lower bound to exclude some possibilities

e.g.

	V_1	V_2	V_3	V_4	V_5
V_1	0	14	4	10	20
V_2	14	0	7	8	7
V_3	4	5	0	7	16
V_4	11	7	9	0	2
V_5	18	7	17	4	0

cost(V_1, V_5) = 20, etc.

	min-out	min-in	sum
V_1	2	2	4
V_2	3.5	2.5	6
V_3	2	2	4
V_4	1	2	3
V_5	2	1	3

The cost $c(v, w)$ can be divided up into the cost $c(v, w)/2$ to *leave* v *in direction* w and the cost $c(v, w)/2$ to *reach* w *from direction* v

Then for each vertex there is a ***minimum cost for leaving*** as well as a ***minimum cost for arriving***

$$min_{\text{leave}}(v) = \frac{1}{2}\min\{c(v, w) \mid w \in W\} \qquad min_{\text{arrive}}(w) = \frac{1}{2}\min\{c(v, w) \mid v \in V\}$$

The ***cost for visiting*** a node v is the sum $min_{\text{leave}}(v) + min_{\text{arrive}}(v)$

Starting for 1, the lower bound is $2 + \underbrace{6+4+3+3}_{\text{other}} + 2 = 20$ (2: out; 2: in)

Then explore 1-2, 1-3, 1-4, 1-5 to get 31, 24, 29, 41

e.g., for 1-2, first add 14 (cost $V_1 \to V_2$), 3.5 for leave V_2 (not to V_1), and cost for visiting other nodes: 5.5, 3, 3 (not from V_1 and not to V_2), and add 2 for going back to 1 (not from 2)

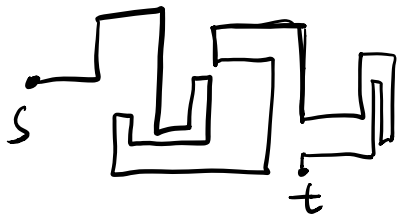
$\Rightarrow 14 + 3.5 + 5.5 + 3 + 3 + 2 = 31$

- Pick the one with smallest lower bound : 1-3
  Then explore 1-3-2, 1-3-4, 1-3-5 similarly .....
  We get lower bounds 24, 30.5, 40.5
- So we pick 1-3-2, then explore 1-3-2-4, 1-3-2-5
  $\Rightarrow$ 37 for 1-3-2-4-5, 31 for 1-3-2-5-4
  $\because 31 < 40.5$, exclude 1-3-5
  similarly, exclude 1-2 and 1-5
- Explore the left choices ....

How to implement? Several components

- given a prefix of some path, calculate the lower bound of cost
- logic of walking further
- logic of comparing and excluding
- .....

(iii) maze problem — backtracking again.



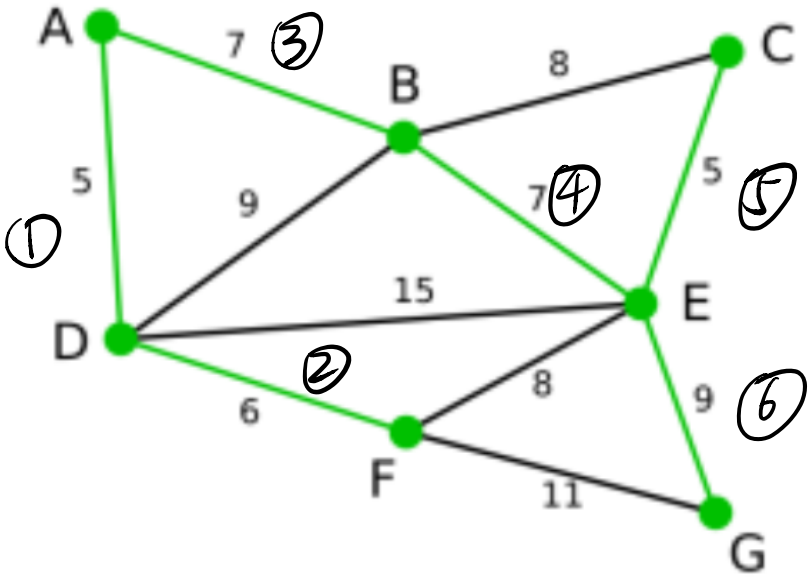
Find a path from s to t:

- recursion — explore all directions (all-direction DFS)
- label visited vertices to avoid searching the same path

EX.3 MST — minimum spanning stree

find subset of edges that covers all vertices with the smallest total cost.

Prim algorithm:



Refer to the lecture note for more info.

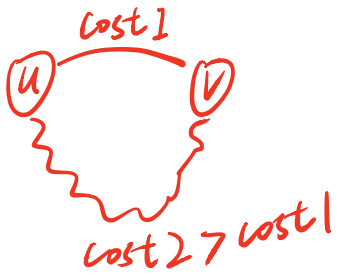
kruskal's algorithm:

- Sort the edge list.
- check all edges, low-cost-first.
- label the edge (u.v) if u and v are in two different connected components.
- Stop when we label |V|-1 edges.

Why it is correct?

Suppose for a low cost edge (u,v), where u and v are in two different connected components:

If we do not label (u,v), we will use a higher-cost path to connect these two components



Then the resulted spanning tree will have higher total cost — not a MST

Disjoint set — used for recording and merging connected components