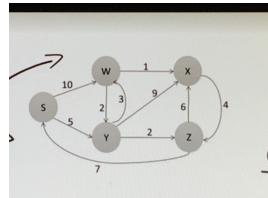


Dijkstra's algorithm

choose node	
$s(-, 0)$	$y(s, s), w(s, 10), x(-, \infty), z(-, \infty)$
$y(s, 5)$	$z(y, 7), w(y, 8), x(y, 14)$
$z(y, 7)$	$w(z, 8), x(z, 13)$
$w(y, 8)$	$x(w, 9)$
$x(w, 9)$	
	$x - w - y - s$ - cheapest to x from s
	$w - y - s$ - cheapest to w from s .
	$z - y - s$ - cheapest to z from s .
	$y - s$ - cheapest to y from s .

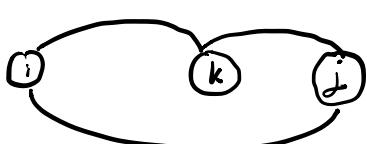


Warshall's algorithm

Dynamic Solution of overlapping solutions

- o $A = \text{Adjacency matrix}$
- o $0, 1$ if \exists nodes
(connected edges)
- o Is some arbitrary node i reachable from some other node j ?
- o Transitive closure $A \Rightarrow B \wedge B \Rightarrow C \Rightarrow A \Rightarrow C$

Label nodes $1 \dots n$



$R_{i,j}^0, R_{i,j}^1, R_{i,j}^2, \dots, R_{i,j}^4$
Direct links $\{k\}$

Recurrence:

R^k

$$R^k(i, j) = R^{k-1}_{i,j} \text{ or } (R^{k-1}_{i,k} \wedge R^{k-1}_{k,j})$$

$\rightarrow k$ goes from $1 \rightarrow n$ n passes.

$$R^0 = \begin{matrix} 1 & 2 & 3 & 4 \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$

$m=4$

Adjacency matrix.

$n^k(i, i) = R^0(i, i)$ or $R^0(i, k) \wedge R^0(k, i)$
reach iteration of k

$k=1$

$i=1 \quad j=1 \quad 2 \quad 3 \quad 4$

$i=2 \quad j=1 \quad 2 \quad 3 \quad 4$

$i=3 \quad j=1 \quad 2 \quad 3 \quad 4$

$i=4 \quad j=1$

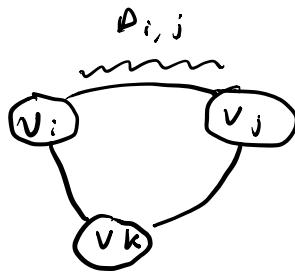
Floyd's walkthrough - all pairs shortest path.

\rightarrow Adjacency matrix

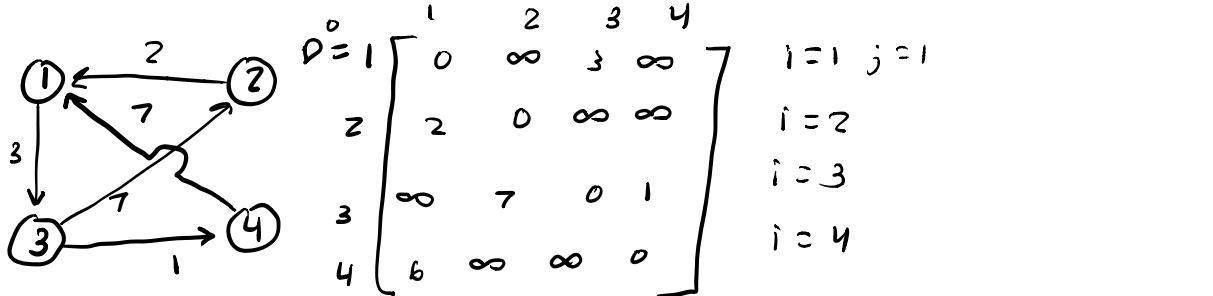
$$D = \begin{bmatrix} 0 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ u_i & & & \ddots & \\ & & & & \ddots \end{bmatrix}$$

$i \quad j$
 $1 \quad 2 \quad 3 \quad 4$

Build each iteration
 $k=1 \dots n \dots$



$D_{i,k} + D_{k,j}$ k as intermediate.
Find least cost.



$$D_{i,j}^k = D_{i,i}^0 + D_{i,1}^0 + D_{1,j}^0$$

$$D^0 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{bmatrix}$$

$i=1, j=1$
 $i=2$
 $i=3$
 $i=4$

$$k=1 \dots n$$

$$D^k(i,j) = \min \left\{ D^{k-1}(i,i), D^{k-1}(i,k) + D(k,j) \right\}$$

Backtracking

- ↳ Planning, Solutions w/ very large solution space.
- ↳ Problems that are impossible to solve.
- ↳ Combinatorial / exponential
- ↳ exhaustive Search
- ↳ eliminates useless / impossible subsets of results.

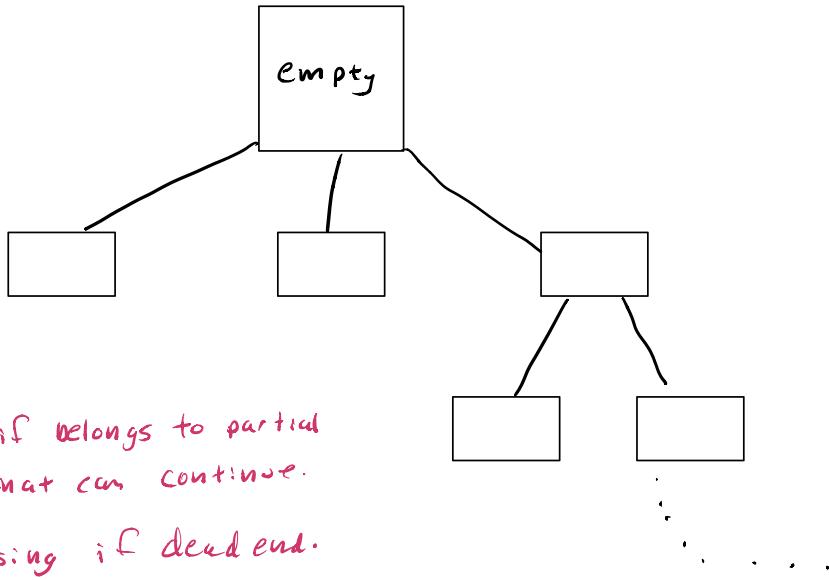
1) Construct a potential solution 1 component at a time.

2) If there are no potential values of remaining components then we do not continue w/ this potential solution.

→ Search space tree: D-F

Back tracking space tree generation.

- ↳ construct by adding a component to existing state as long as adding component will not violate a constraint
- ↳ If no options for next component end state path
- ↳ Backtrack up a level to prior state and try next component.



n-queens problem

4 queens

NxN chess board

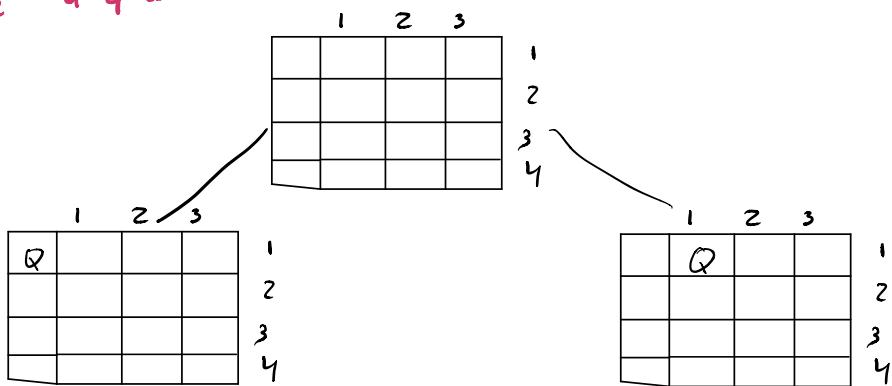
Place queens on board so that no queen can attack another

$n=1$ trivial

$n=2$ no

$n=3$ no

Trace 4 queens



General Backtracking logic

✓ Partial sol'n First : Components

Algorithm Backtrack ($x[1 \dots i]$)

If $x[1 \dots i]$ A solution returns it

else

for each element $\in S_{i+1}$ [within constraints]

$x[i+1] \leftarrow x$

Backtrack ($x[1 \dots i+1]$)

Q	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

board

CheckSafe (Board, row, col)

Board $[row, col]$ is it safe? If yes, place queen on board at the row and column n.

```
bool placeQueens (board, N, row) {
    if (n==2 || n==3) return false
    if (row==N) return true // Board complete
    for (int col=0; col<N; col++) {
        if (placementSafe (board, N, row, col))
            board[row][col]=1;
        if (placeQueens (board, N, row+1))
            return true;
        else
            board[row][col]=0;
    }
    return false;
}
```

7	x	9	2	
10 x		6		x
5	8	1		3
x	11			x
	2	4	x	

Knights visits every square.



loop move 1
 move 2
 move 3
 move 4 } 8 possible moves

↳ Greedy technique:

Making Change

25, 10, 5, 1

30 \$ → 25 + 5, 2 coins.

$$\begin{array}{l} (25, 10, 5) \\ \hline 30 \end{array} \xrightarrow{1+5=6} 3$$

Optimal sol'n
→ least # of coins possible.

Greedy approach → take best available alternative at each step and hope that a sequence of "best" alternatives leads to best optimal sol'n.

- 1) Feasible constraints
- 2) locally optimal. - all feasible
- 3) Irrevocable.

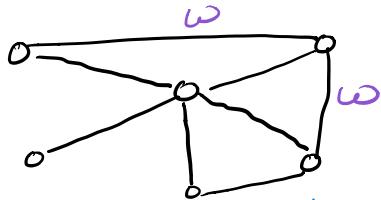
9.1 Prim's algorithm

→ minimum spanning tree problem.

Subgraph connected.

no cycles (true).

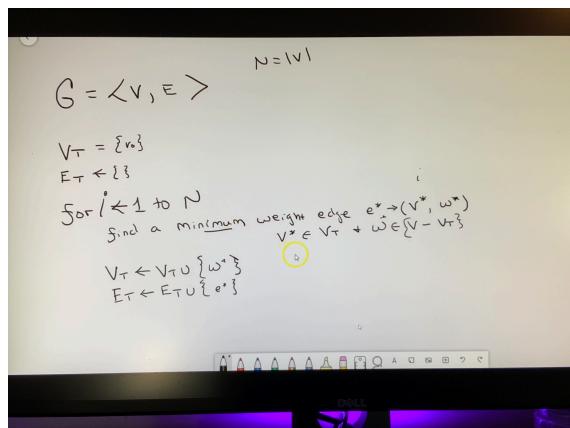
min total weight (cost)

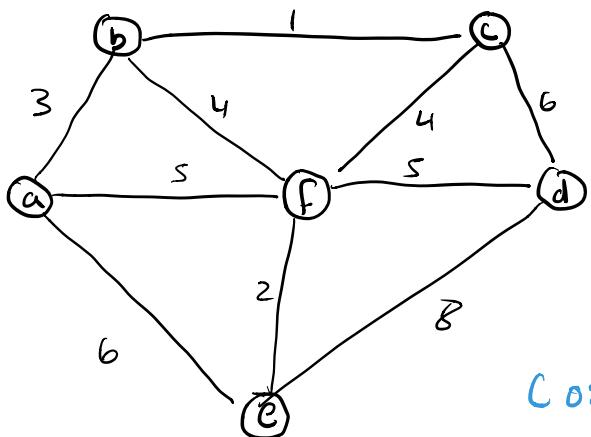


Exhaustive → generate all spanning tree & pick cheapest.

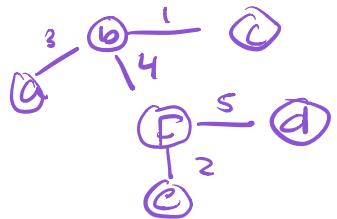
Prim's

- Iteratively construct minimal spanning tree by adding one vertex at a time.
- Choose next vertex not in tree w/ cheapest edge to tree.





Prim's trace



$$\text{Cost} = 1 + 3 + 4 + 5 + 2 = 15.$$

Choose a to add to set.

$$\underline{b(3, a)}, c(\infty, -), d(\infty, -), e(6, \infty), f(5, a)$$

Choose b (cheapest)

$$\underline{c(b, 1)}, d(\infty, -), e(6, a) \quad \underline{f(4, b)}$$

Choose c

$$d(b, c) \quad e(6, a) \quad \underline{f(4, b)}$$

Choose f

$$d(f, s) \quad \underline{e(f, 2)}$$

Choose e

$$d(s, f) \rightarrow s < 8!$$

Complexity

each pass \rightarrow choose node. $|V|$

$N \cdot N-1$

Complete \hookrightarrow [create list of edges] $-N-1$ $|V|$ $\underline{\underline{=}}$

$N-1$

Reachable from node.

Kruskals algorithm

- \hookrightarrow # of edges.
- \hookrightarrow minimum spanning tree 1 element at a time by choosing edges that do not create cycles.

Kruskals

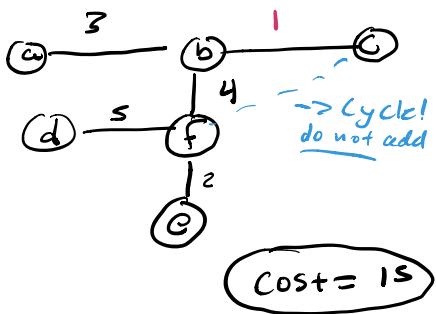
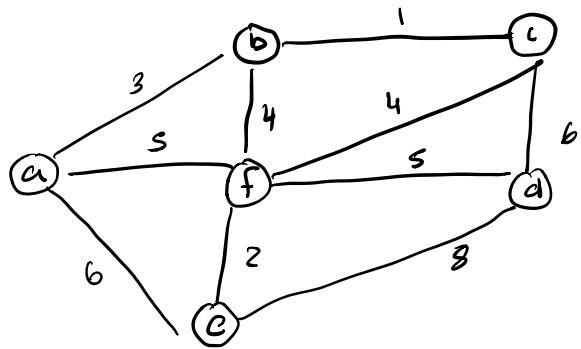
$$G = \langle V, E \rangle$$

$$E_T \leftarrow \{\}$$

Sorted edges by weight

Iterate through node count.

Choose cheapest edge that connects components w/ creating a new cycle.



Sort edges

bc	ef	ab	bf	df	c f	a f	d f	a e	c d	d e
1	2	3	4	4	4	s	s	b	b	8

cycle cycle

$n \log n$