Depth-First Search

depth search

depth = 0 → depth = 1 : DFS(G, 0)

⋮

finish one path

↓

backtrack to node with another path

DFS(G, 1)

search nodes in graph

Breadth-First Search

using queue

enqueue first node

↓

dequeue one node

↓ if have adj...

enqueue adjacent node(s)

if queue.length ≠ 0

↓ if = 0 : stop

if no...

stop

Dijkstra's Algorithm

graph : shortest path from A to B

initialization : Q = (A: 0), (B: ∞), (C: ∞), (D: ∞)

iteration...

↓

extract vertex A, update vertices B, C, D ...

A: d = 0, B: d = 1, C: d = 2

Q = (A: 0), (B: 1), (C: 2) ...

↓

delete extracted point from Q

Ford-Fulkerson algorithm

Graph
Network flow

initial network (G) → flow/capacity (0)

residual graph (G_R) → flow

iteration: Augment path : A - B - C - D → Flow = Flow +

network: flow / capacity

residual graph: flow in different direction

Edmonds-Karp Algorithm

Graph
network flow

iteration

path with fewest edge

path < s, v_1, v_3, t >

residual network

final result

residual network + flow network

Prim's algorithm

Tree: minimum spanning tree in graph

find minimum-weight edge

if form cycle

skip

else pick it and insert in new graph

Kruskal's algorithm

sort edge's weight in increasing order

pick first edge → insert → pick next edge

if form a cycle, skip

else, pick it and insert in new graph