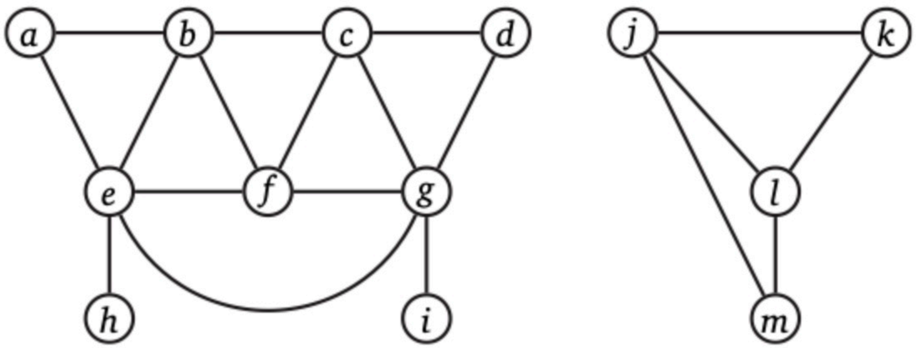
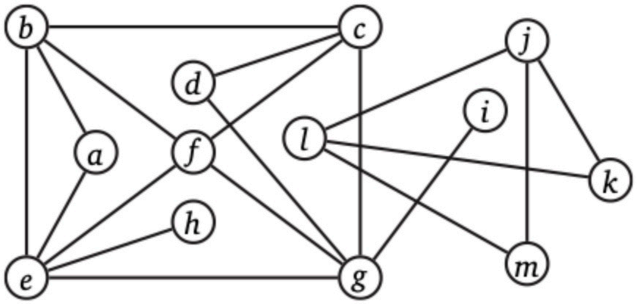


planar drawing

A graph $G=(V,E)$

V: an arbitrary finite set called vertices (or nodes)

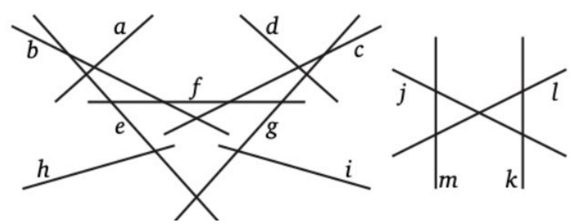


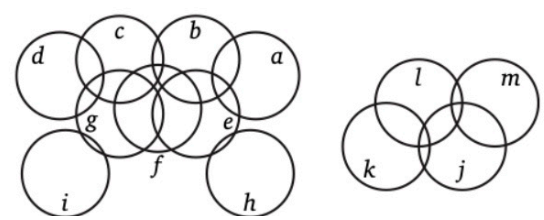
E: some pairs of vertices

uv: $\{u,v\}$ undirected: unordered

$u \to v$: (u,v) directed: ordered

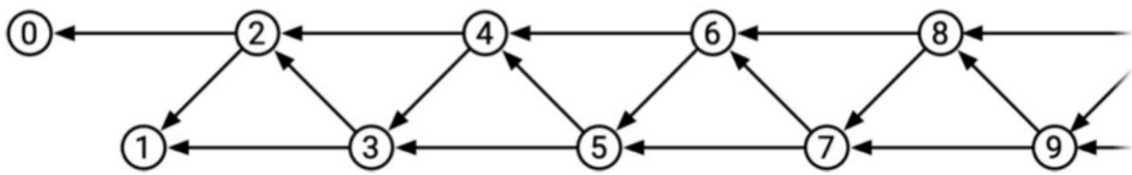
intersection graph:





dependendency graph: vertices: subproblems in an instance of a recursive algo

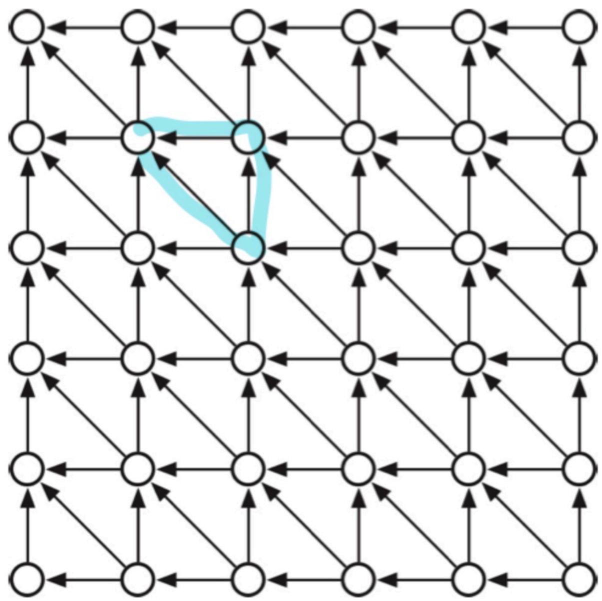
edge $u \to v$: u directly calls v



RecFibo

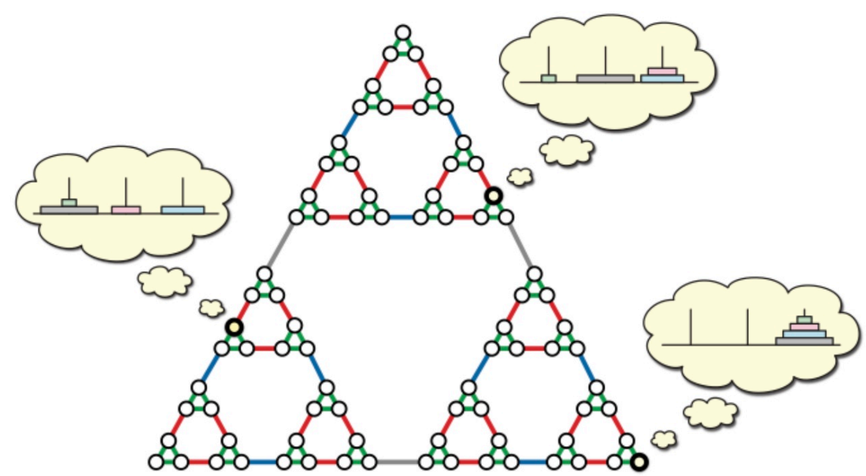
DAGs

Edit Distance

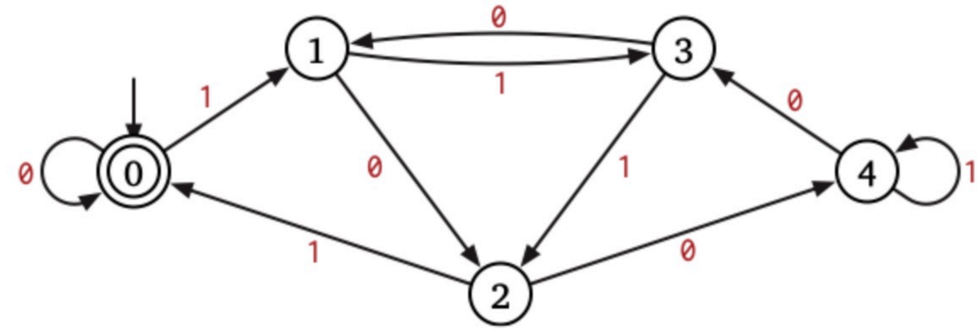


4-disk Hanoi:

Configuration graph:

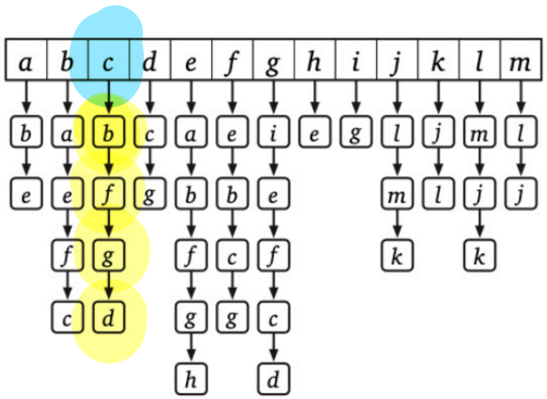


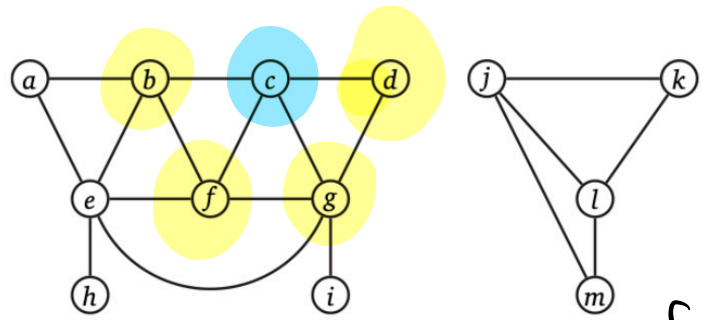
state-transition graph:



Data structures:

Default: Adjacency list





\# neighbors of u

$uv \in E$: $O(\min(\deg(u), \deg(v)))$

$u \to v \in E$: $O(\deg(u))$

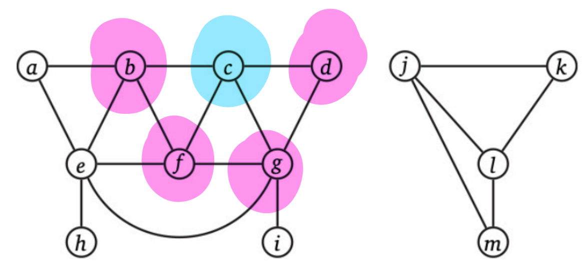
find all neighbors of u in $O(\deg(u))$ ✓

if $u \to v$: v is in u's list

$O(V+E)$ space ✓

Adjacency matrix

	a	b	c	d	e	f	g	h	i	j	k	l	m
a	0	1	0	0	1	0	0	0	0	0	0	0	0
b	1	0	1	0	1	1	0	0	0	0	0	0	0
c	0	1	0	1	0	1	1	0	0	0	0	0	0
d	0	0	1	0	0	0	1	0	0	0	0	0	0
e	1	1	0	0	0	1	1	1	0	0	0	0	0
f	0	1	1	0	1	0	1	0	0	0	0	0	0
g	0	0	1	1	1	1	0	0	1	0	0	0	0
h	0	0	0	0	1	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	1	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	0	1	1	1
k	0	0	0	0	0	0	0	0	0	1	0	1	0
l	0	0	0	0	0	0	0	0	0	1	1	0	1
m	0	0	0	0	0	0	0	0	0	1	0	1	0



$u \to v \in E$ iff $M_{u,v} = 1$

$uv \in E$? : $O(1)$ ✓

Find all neighbors of u: $\Theta(V)$

Space: $\Theta(V^2)$

Reachability: Given a vertex s, what other vertices can we reach (i.e. for what v is there an (s, v)-path or walk?)

For an undirected: what vertices are in s's (connected) <u>component</u>.

maximal set of vertices with paths between them

depth-first search

```
RecursiveDFS(v):
    if v is unmarked
        mark v
        for each edge vw
            RecursiveDFS(w)
```

B

```
IterativeDFS(s):
    Push(s)            queue
    while the ~~stack~~ is not empty
        v ← ~~Pop~~ Pull
        if v is unmarked
            mark v
            for each edge vw
                Push(w)
```

BFS: finds unweighted shortest paths

```
WhateverFirstSearch(s):
    put s into the bag          ← O(1)
    while the bag is not empty
        take v from the bag     ← ≤ 2E+1
        if v is unmarked
            mark v              ← O(V) times
            for each edge vw
                put w into the bag
```

≤ 2E times

Will mark all vertices reachable from s <u>and no others</u>.

If bag takes constant time per operation...

$O(V+E)$ time

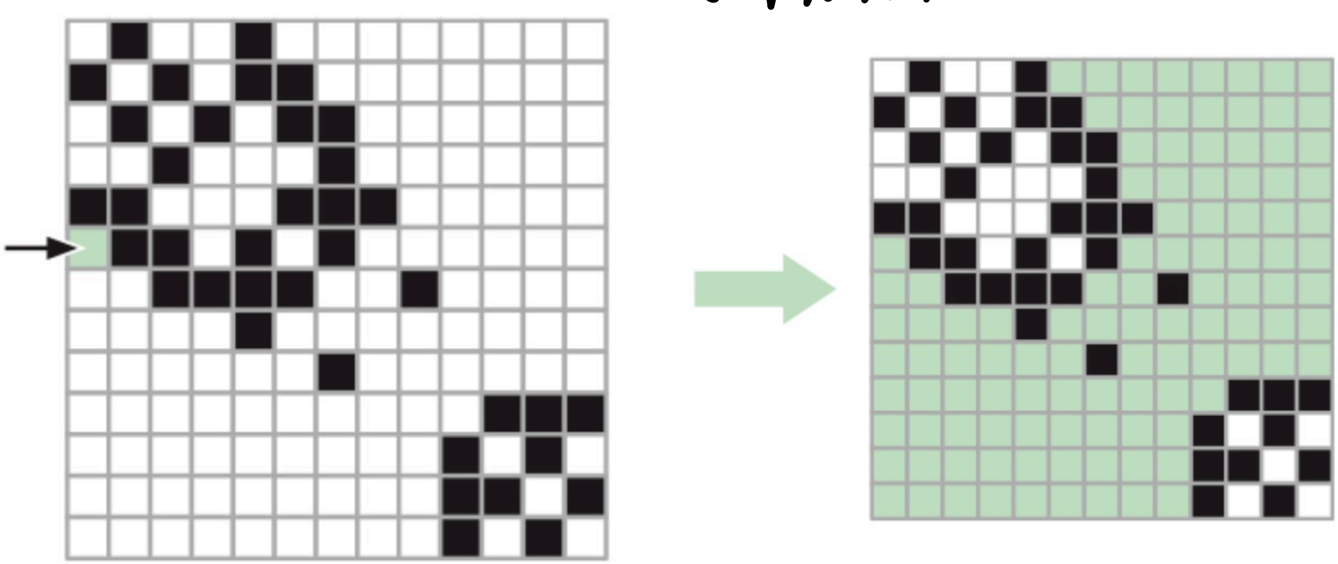
Can do directed graphs by looping over all $v \to w$.

Graph Reductions:

Pixel map: A 2D array whose values are colors.

Each element of the pixel map is a pixel.

A <u>connected region</u>: maximal set of pixels with a path between them using hops between same-color neighbors.



← <u>flood-fill</u>

Reduce to reachability.

Let $G = (V, E)$ $\quad$ V: pixels of pixel map

$E = \{uv \mid u + v$ are neighbors of the same color$\}$

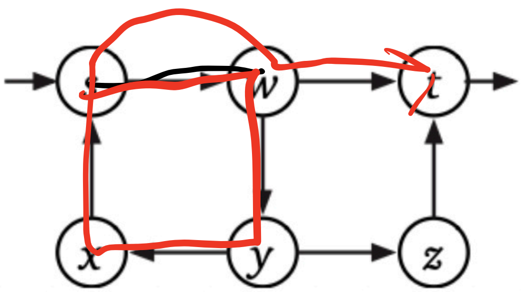
s: pixel we chose to color

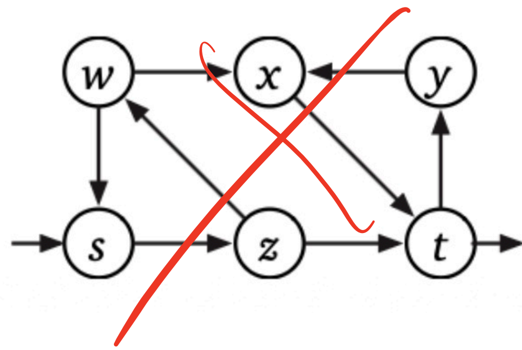
Call WhateverFirstSearch(s) + color the marked vertices.

Analysis: $|V| = n^2$
$|E| \leq 2n^2$ takes $O(V+E) = O(n^2)$

Given a directed graph $G=(V,E)$ & $s,t \in V$.

Is there a walk from s to t of length $0 \bmod 3$?





Reduce via graph layering.

Build new $G'=(V',E')$ based on G, & call WFS in G'.