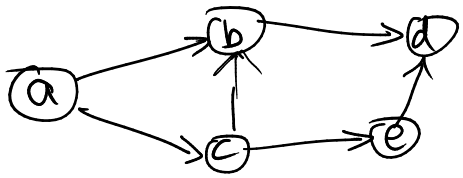
Graphs

DAG → graph
- directed
- acyclic

Examples:
- Scheduling with dependencies
- DP problems

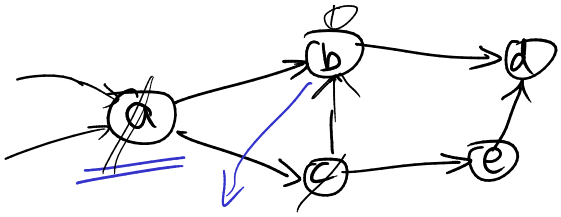


Topological Ordering

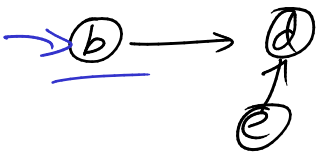
- Find one
- Count
- Lex. smallest ←
- At i^{th} pos → possible nodes (?)
- Shortest paths (-ve weights)

- a c b e d
- a c e b d

1 → {a} 3 → {b, e}



del a

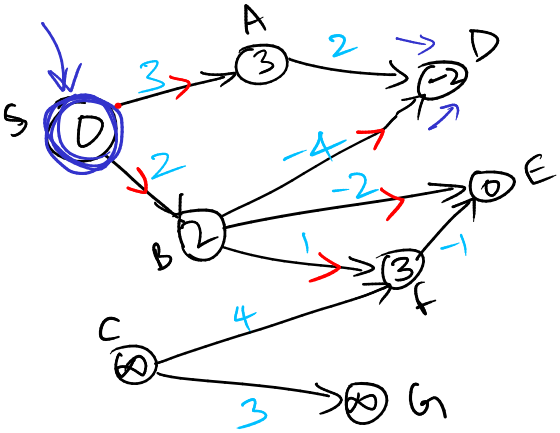


$\{u \mid in_u = 0\}$ ← set / heap

a c b

- $O(N^2)$
- $O(N+M)$
- $O(M + N \log N)$

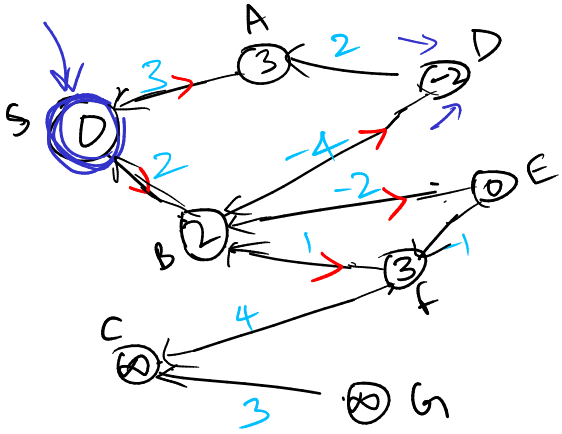
Shortest Paths



S A B D F E ← ?

top. order.

→ O(N+M)



1. Is it a DAG?
   - Yes
2. S.P. from u to O
   $\forall u$
   - Same as prev. solution!

DP

- Set of States (S) $\longleftrightarrow$ V
- Transition function (T) $\longleftrightarrow$ E

} Graph (DAG)

Eg.

S: (day, profit)

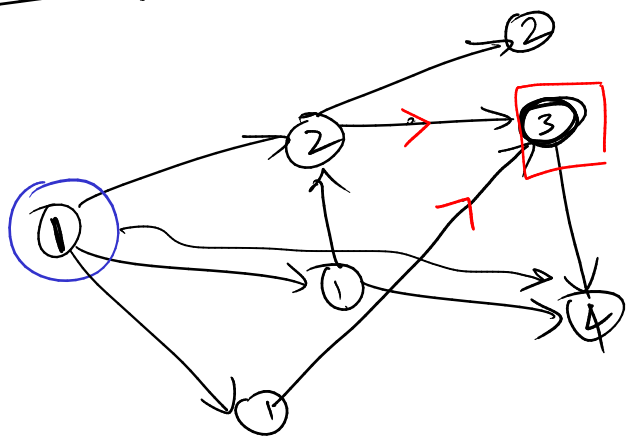
T: (d, p) $\leftarrow$ (d+1, p+f_d)

`int dp[N][10];`

- N: vertices
- 10: distance

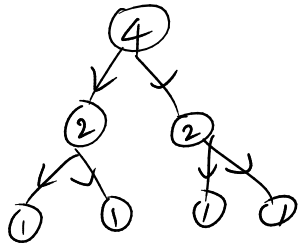
initialize 0 to sources

(P) No. of Paths

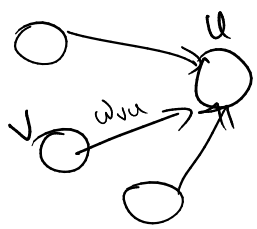


$O(N+M) + O(\text{TOP.})$

Trees



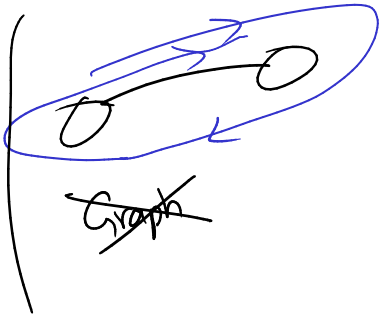
Longest Path



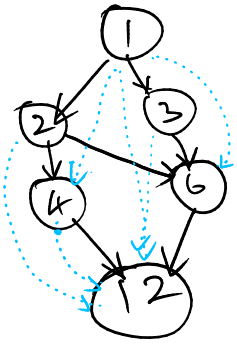
min (SP)

$\max_v (d_v + w_{vu})$

all $w = 1 \Rightarrow$ longest chain



12 : 1, 2, 3, 4, 6, 12

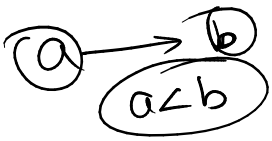


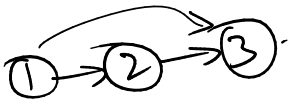
Divisor lattice

$1, 2, 4, 12 = 2 \times 2 \times 3$

1 → → → 12

$S = \{1, 2 \dots 10\}$

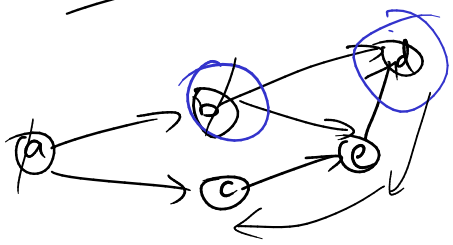




Posets

partially ordered

Top. sort



a b c e d

easier

```
dfs(u){
  if(u is seen) return
  for(v : adj_u)
    dfs(v)
  topo.append(u)
  seen[u] = true
}
```

- lexicographic minimum ✗