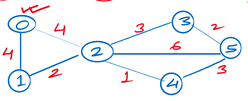
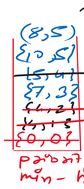
Shortest Distance Algorithms :→

Dijkstra's Algorithm → Source to all nodes.

① Queue ② Set ③ Priority queue

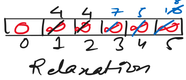


Adjacency List
{ node , weight }
0 → { {1,4} , {2,4} }
1 → { {0,4} , {2,2} }
2 → { {0,4} , {1,2} , {3,7} , {4,6} }
3 → { {2,3} , {5,2} }
4 → { {2,1} , {5,3} }
5 → { {2,2} , {4,3} }



Big O [(E + V) log V]

Distance Array:



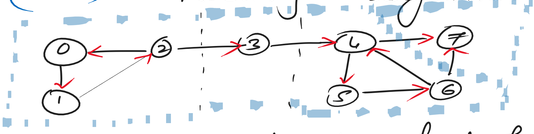
Relaxation

For adj list → TC → Big O [E+V (log V)]
For matrix → (V²)
4x4 for for

*** Strongly connected components:

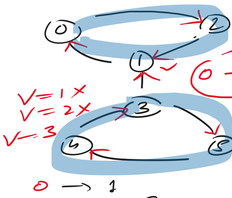
(SCC)

Kosaraju's Algorithm



An scc is a component or part of a graph where you can reach every other node from any node in that component.

Note: Single node is also an SCC.



0 → 1
1 → 2, 3
2 → 0
3 → 4
4 → 5
5 → 3

dfs 1 → Finish Time

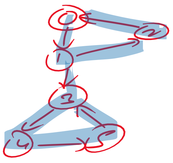
dfs(0)
dfs(1)
dfs(2)
dfs(3)
dfs(4)
dfs(5)

Stack

1
0
2
3
4
5

dfs: Transpose

Stack: → 5 4 3 2 0 1



dfs(0)
dfs(1)
dfs(2)
dfs(3)

"Those who forget the past, are forced to repeat it."

"Dynamic Programming"

A bigger problem can be solved by finding solutions of overlapping subproblems.

TLE
① Recursion
② Tabulation — Store prev val

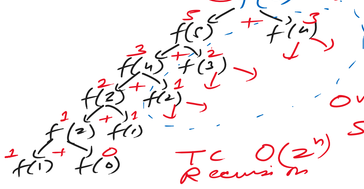
don't calculate prev
③ Memoization
④ Space optimization (vars) O(1)

Fibonacci Series: nth Element: →

0, 1, 1, 2, 3, 5, 8, 13, 21 - - - -
0 1 2 3 4 5 6 7 8

6th Fib Value: → fib(n) = fib(n-1) + fib(n-2)

f(6) → Bigger Problem



Overlapping Subproblems

TC O(2ⁿ) Recursion

Level / height n → 2ⁿ