Date: 28/05/21

Black Board

Design and Analysis of Algorithms

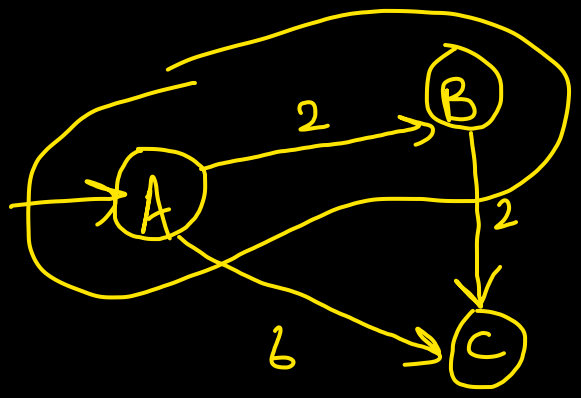
Topics:

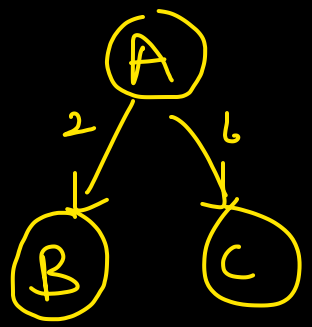
- Dijkstra's Algorithm

Single Sort Shortest Paths for various types of graphs

Directed or Undirected Un-weighted Graph	BFS	O(|V|+|E|)
Directed Acyclic Graph	DP based on Topological Sort	O(|V|+|E|)
Directed or Undirected Weighted Graph (+ve weights)	Greedy Algorithm (Dijkstra's Algorithm)	O((|V|+|E|)lg|V|)
Directed or Undirected Weighted Graph (+ve or –ve weights)	DP Algorithm (Bellman-Ford Algorithm)	O(|V||E|)

The problem with BFS and weighted Graphs





BFS output

correct one

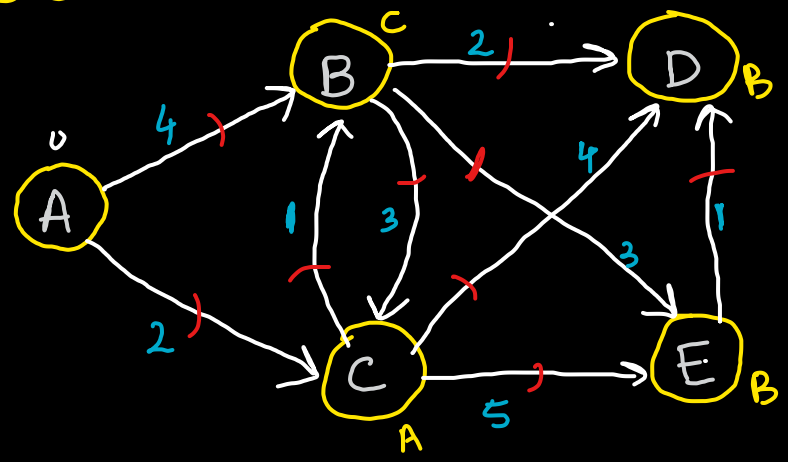
Dijkstra's Algorithm: main idea

⇒ Replace the FIFO Queue in Simple BFS with a priority Queue (min Heap)

{ other updates also }

Dijkstra's Algorithm

Source A



Heap H

init:

~~A 0~~ B ∞ C ∞	D ∞ E ∞

→ step 1

B 4 ~~C 2~~	D ∞ E ∞

step 2

~~B 3~~	D 6 E 7

step 3

~~D 5~~ E 6	

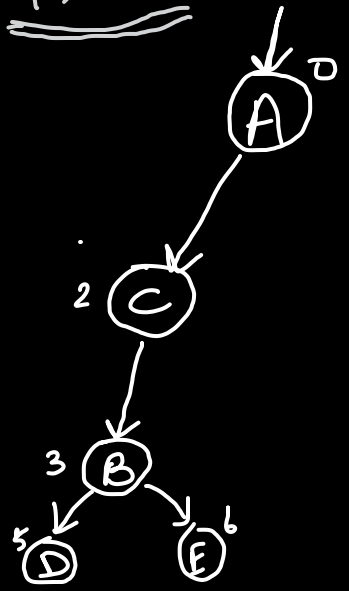
step 4

~~E 6~~	

step 5

STOP

Tree

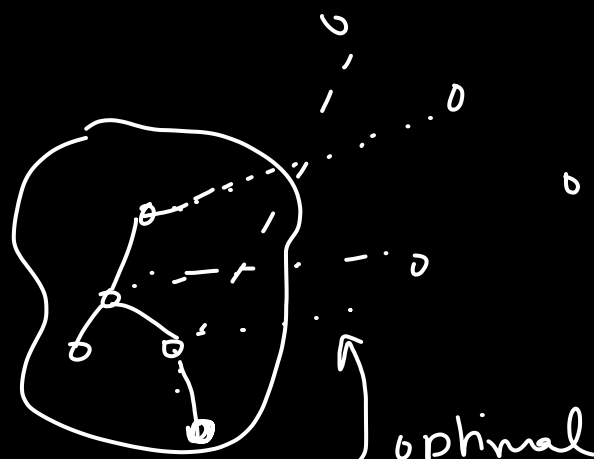


SSSP tree

A, C, B, E

A
C
B
E

In each iteration a vertex is extracted from the heap and added to the SSSP tree



optimal distances from the tree.

grows vertex by vert.

Data:

dist[x]: dist of x from the source.

Par[x]: parent of x in the SSSP tree

w: array of edge weights

$G = (V, E, w)$

H = bin min heap

Relax(x, y, w)

IF dist[x] + w(x,y) < dist(y)

dist[y] = dist[x] + w(x,y)

par[y] = x

return true

return false

O(1) method

Dijkstra's Algorithm Code

① Dijkstra(G=(V,E,w), s)

//Setup

O(|V|) For each $x \in V$:
dist[x] = ∞
par[x] = nil

O(1) dist[s] = 0

O(|V|) Create H
H.buildHeap(V, dist)

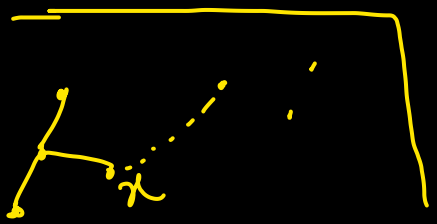
② O(|V|) WHILE !H.empty()
x = H.extractMin()

O(1 V lg|V|) For each $(x,y) \in E$:
IF Relax(x,y,w)
H.decreaseKey(y, dist(x)+w(x,y))

new. enhant in the tree

$O(|E| \lg |V|)$

$T(G=(V,E,w)) = O((|V|+|E|)\lg|V|)$



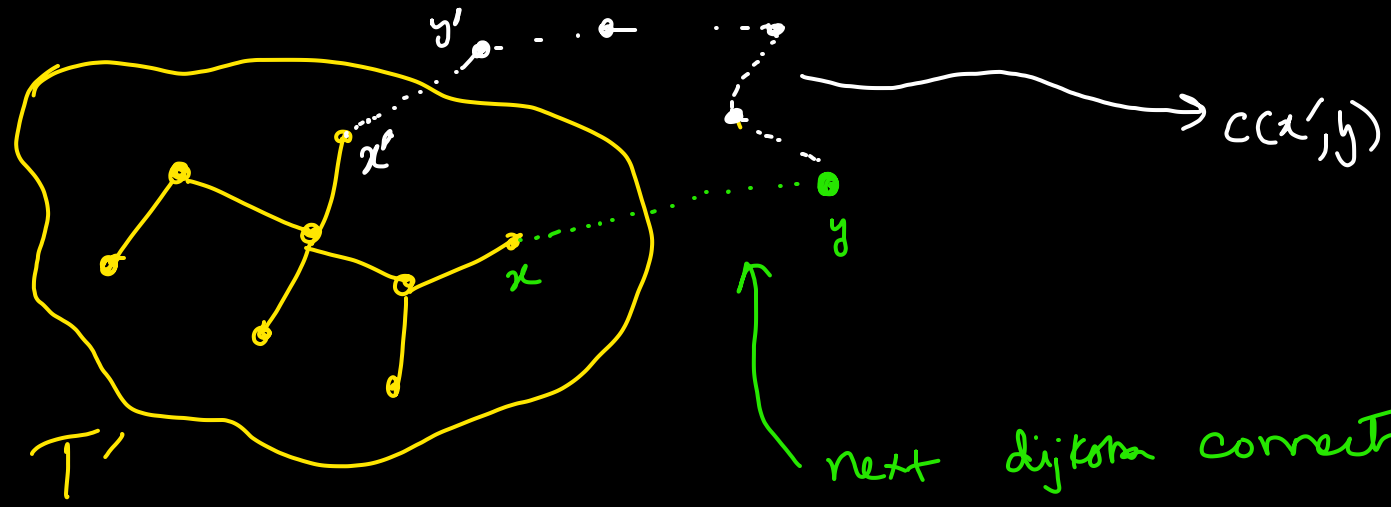
$T(G) = O(|V| \log |V| + |E|)$

Correctness of Dijkstra's Algorithm

Base case: $|V_T| = 1$

Ind. Hypothesis: T' is an optimal tree for k vertices after k iterations of Dijkstra's algo.

Ind. step:



$|V_{T'}| = k$

dist[y] ↳ A — dist[x] + w(x,y) — Dijkstra chose

↳ B — dist[x'] + c(x',y) — almost optimal.

B

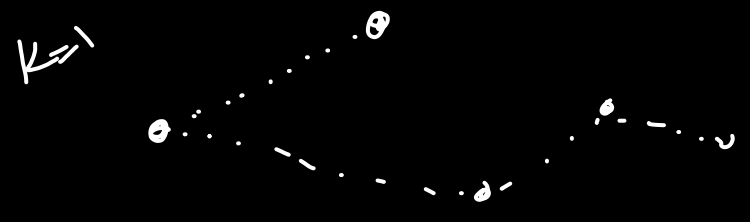
$$dist[x'] + c(x',y) = \underbrace{dist[x'] + w(x',y')}_{\text{in the heap}} + c(y',y)$$

$y' = y$?

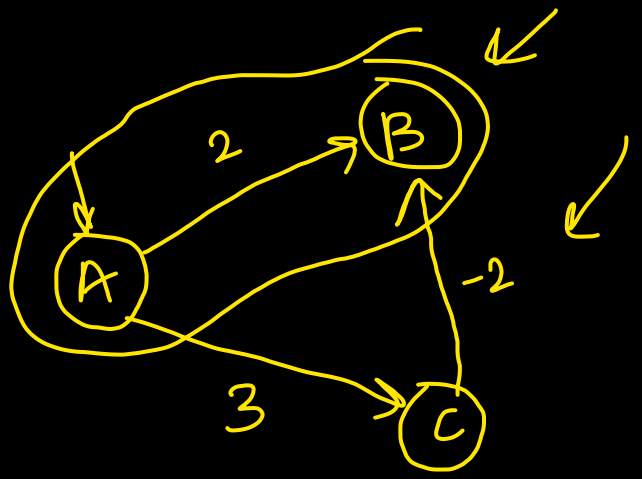
$$dist[x'] + w(x',y') \geq \underbrace{dist(x) + w(x,y)}_{\text{comes from the H}}$$

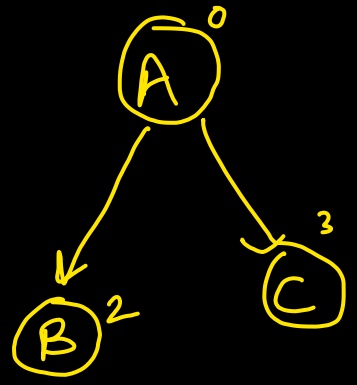
$\Rightarrow$ $dist[x] + w(x,y) \leq dist[x'] + c(x',y)$

$\Rightarrow$ Adding y to T' we get a $k+1$ vertex tree which is optimal.



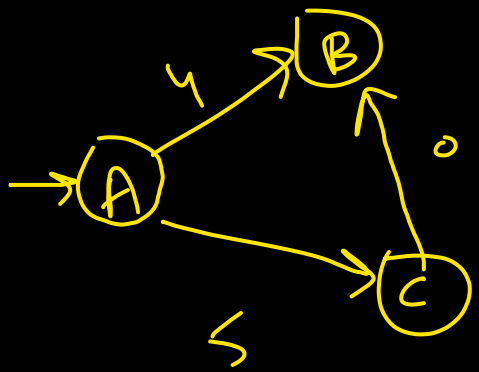
Limitation of Dijkstra's Algorithm

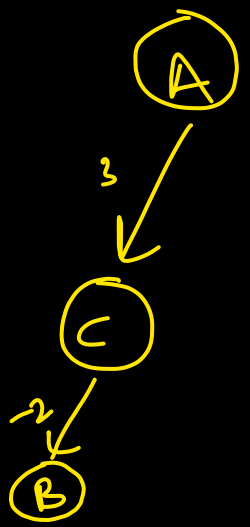




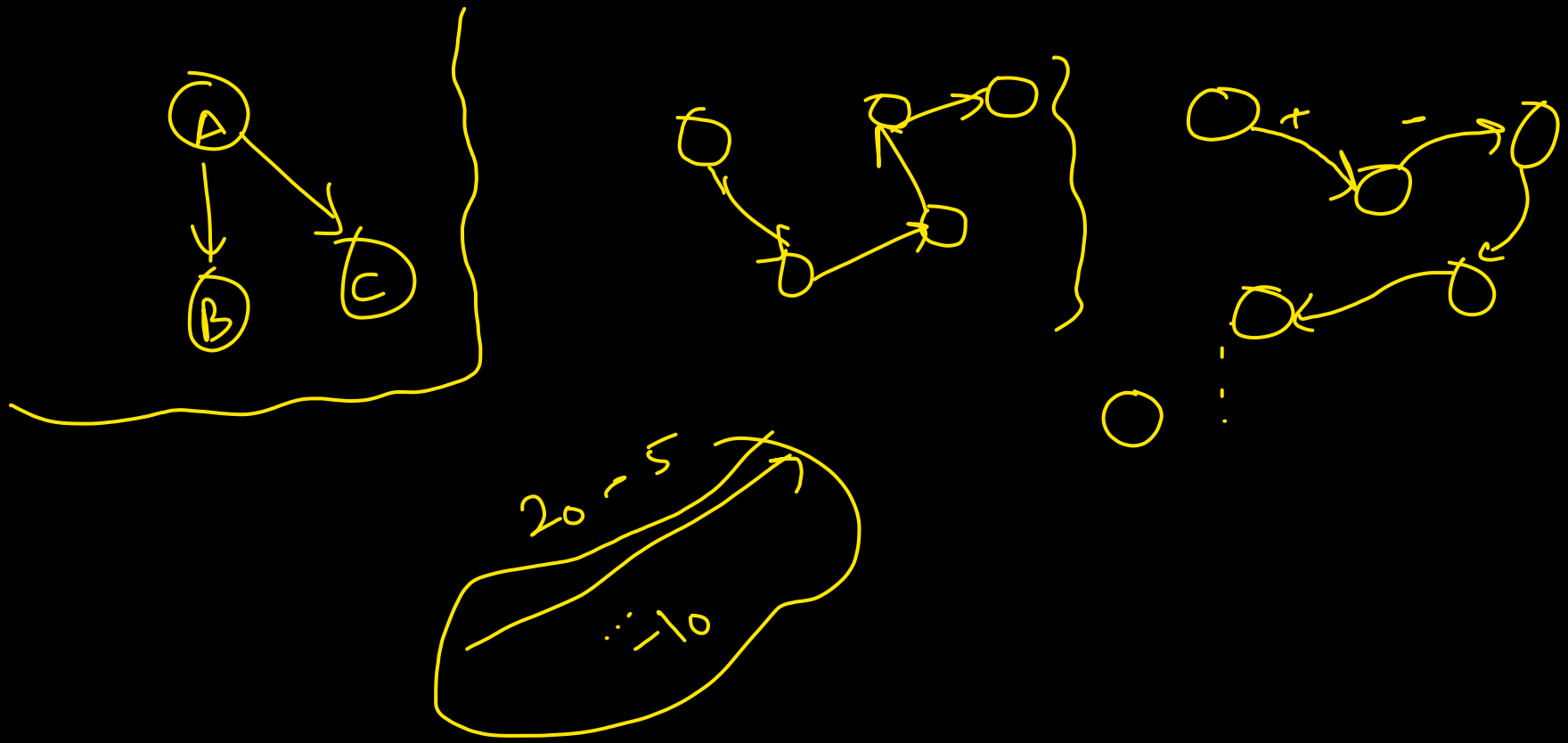
incorrect tree

produced by Dij.





optimal


A
B
C
+
-
20 - 5
= 10