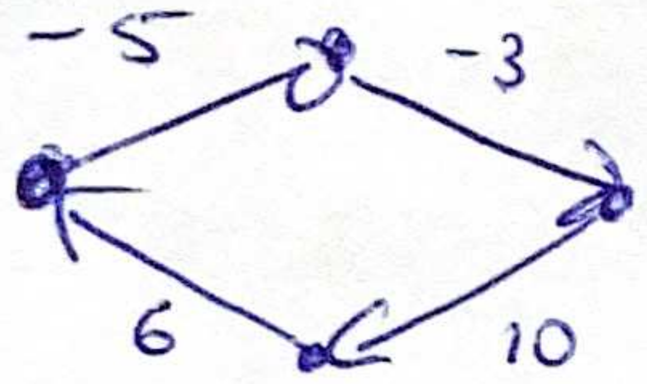
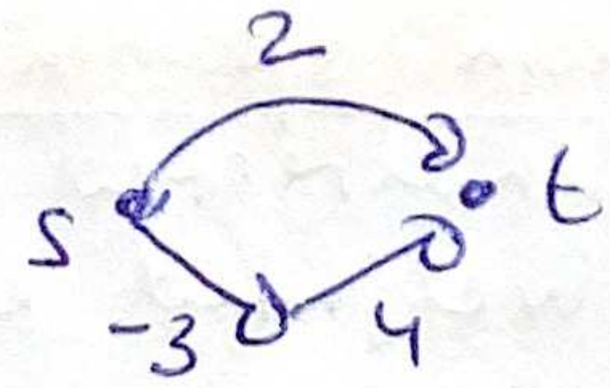
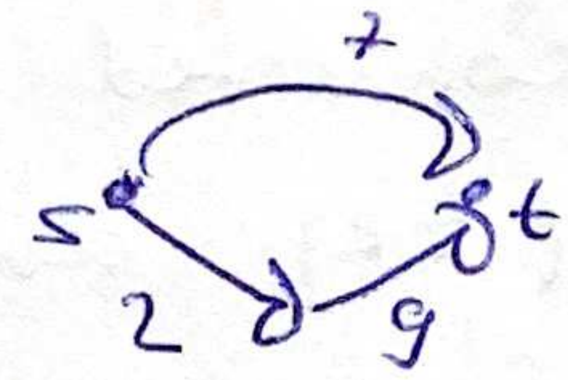
CS 124 Lecture 6 (1



how to handle negative edges

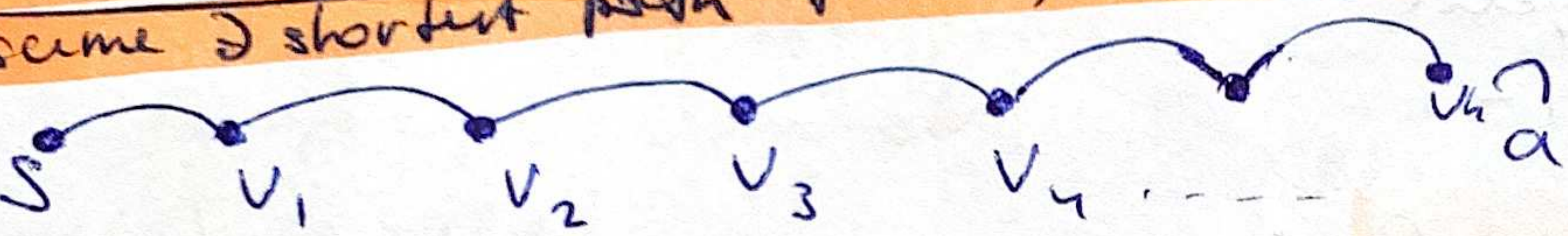
idea: add smallest
- weight to all
→ wrong
- priority queue update won't work





Bellman - Ford

assume ∃ shortest path from S, v_1, v_2 ----- v_k ... a



property: if S → a is shortest path,
then each subpath is shortest
↳ inductively find shortest path

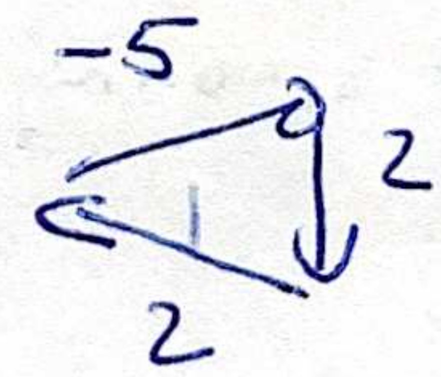
SP2 (1st time loop, 2nd time loop)

for v ∈ V do
  dist[v] := ∞
  prev[v] := nil
dist[s] := 0
for i = 1 ... n-1 O(V)
  for (v, w) ∈ E update (v, w) O(E)

O(E · V)

After k rounds 1st k edges of the shortest path found

↗ fine with negative weights, but does not allow for negative cycles



→ there is no finite # of steps because there is no shortest path

how to deal with negative cycles?

run through E one extra time
[redacted] no neg. cycles ⟺ nothing changes in nth loop
also once nothing changes, it cannot change in the next round
↳ [redacted] stop early

Graph Algs

Minimum spanning trees

* undirected graph
  weighted edges

Tree: connected and acyclic

Lemma (LTR): any 2 of these ⇒ 3rd
1) G is connected
2) G is acyclic
3) $|E| = |V| - 1$

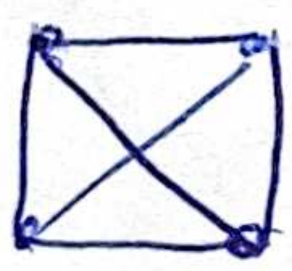
Spanning tree ; given G:

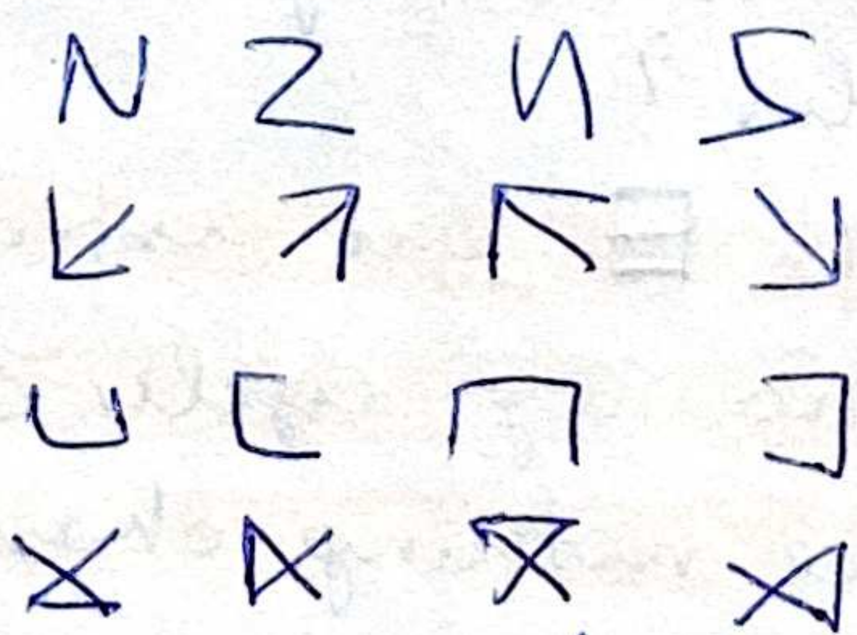
$T \subseteq E$ on all of V (vertices of G)

Min spanning tree:

spanning tree minimizing $\sum_{e \in T} w(e) = w(T)$

Baseline alg:
for each spanning tree calculate its weight, keep the minimum

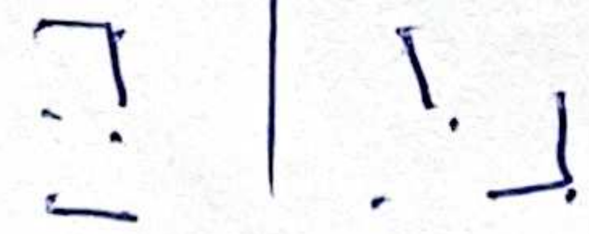
[redacted] n		# spanning trees
2		1
3		3
4		16
5		125
n		n^{n-2}



superexponential growth of the problem size

MST – an edge at a time

S V−S

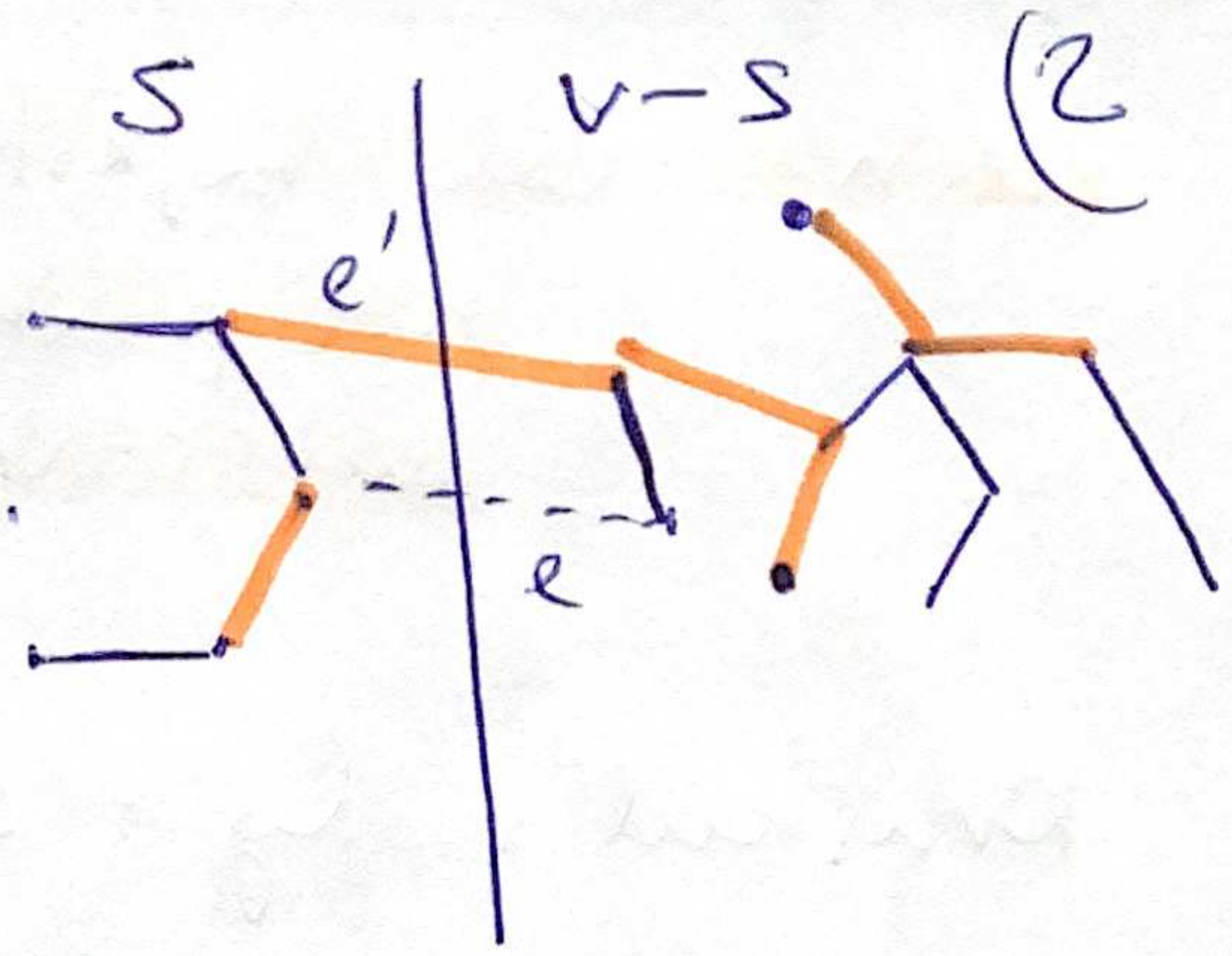


Cut property:
Let $X \subseteq T$, where T is an MST of G
Let $S \subset V$, such that no edge in X crosses from S to V−S
Let e be a min weight edge from S to V−S
Then $X \cup \{e\} \subseteq T'$ for some MST T'

→ can construct MST in a greedy fashion

Proof by contradiction

(2



assume $e \notin T$ [redacted]

adding e creates a cycle

from $S \to V-S \to S$, since

$\exists\, e' \in T$, going from $S \to V-S$

let $T' = T \cup \{e\} - \{e'\}$

T' is a spanning tree, since the single cycle eliminated

$|E| = |V|-1$ and acyclic $\Rightarrow$ connected $\Downarrow$ spanning tree

— not yet added

— added

$w(e) \leq w(e')$

$w(T') \leq w(T)$

if $w(e) < w(e')$ contradiction $\Rightarrow e \in T$

if $w(e) = w(e')$, then T' is another MST

$\Rightarrow$ e is in an MST ✓

General algorithm form

$X = \emptyset$

Repeat until $|X| = n-1$

cut $\to$ pick $S \subseteq V$ w/ no edge in X crossing S, V-S

exploit cut property $\to$ get lightest edge e [redacted] between S and V-S

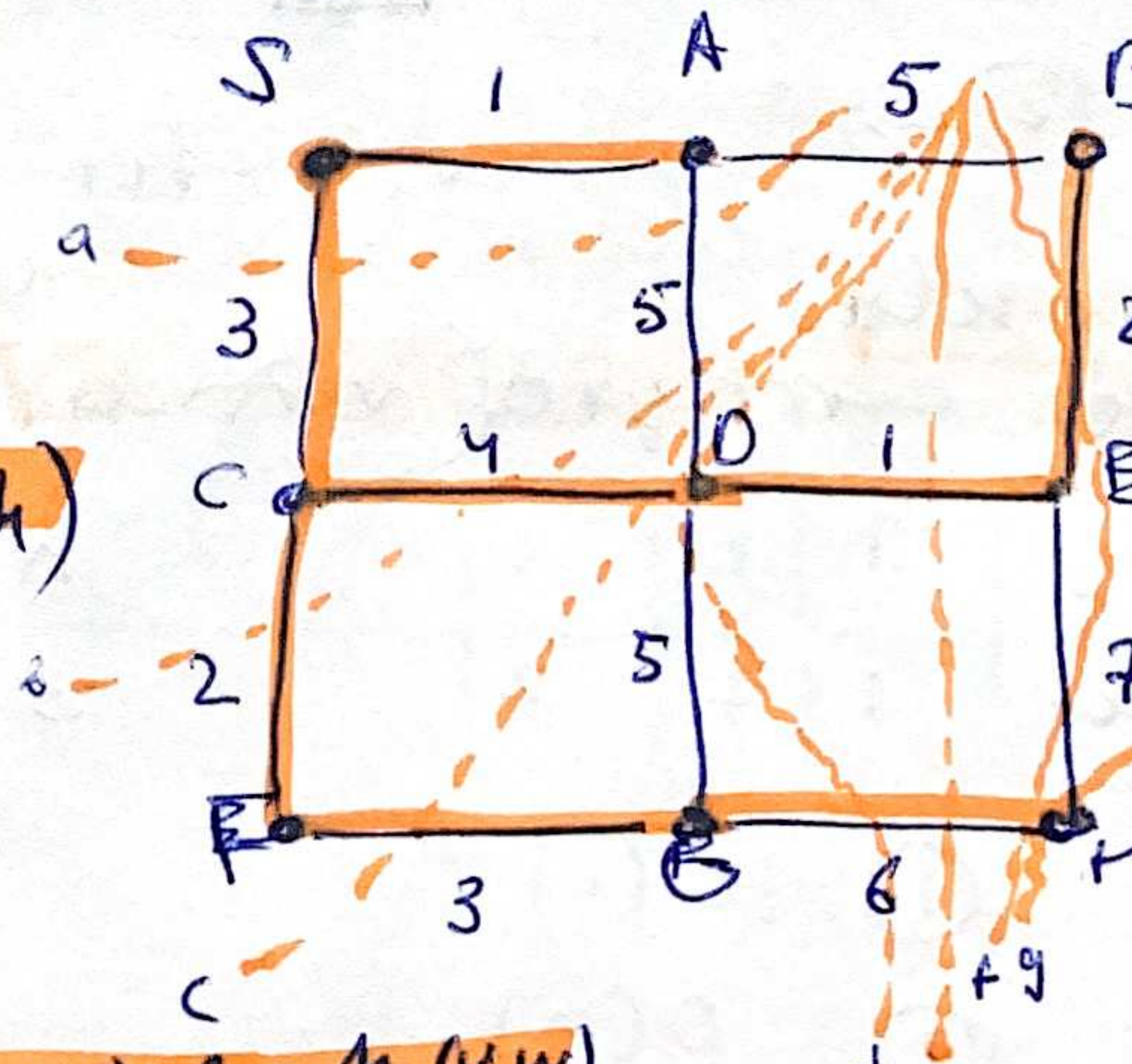
$X = X \cup \{e\}$

One way: force X at each step to be a connected subtree

Prim's

$\to$ Prim's alg.

use priority queue (e.g. heap)



$H := \{s:0\}$

for $v \in V$

  $dist[v] := \infty$, $prev[v] := nil$

$dist[s] := 0$

while $H \neq \emptyset$

  $v :=$ deletemin (H)

  $S := S \cup \{v\}$

  for $(v,w) \in E$ and $v \in V-S$ do

    if $dist[w] >$ length (v,w)

      $dist[w] :=$ ~~dist[w] + length(v,w)~~ length(v,w)

      $prev[w] := v$

      insert $(w, dist[w], H)$

go trough E once

O(E insert + V delete)

heap O(E log V)

list $O(V^2)$

like Dij but without adding ~~length~~ to previous sum

$\to$ S:0

~~A:1~~, C:3

~~C:3~~, B:5, D:5

~~F:2~~, D:4, B:5

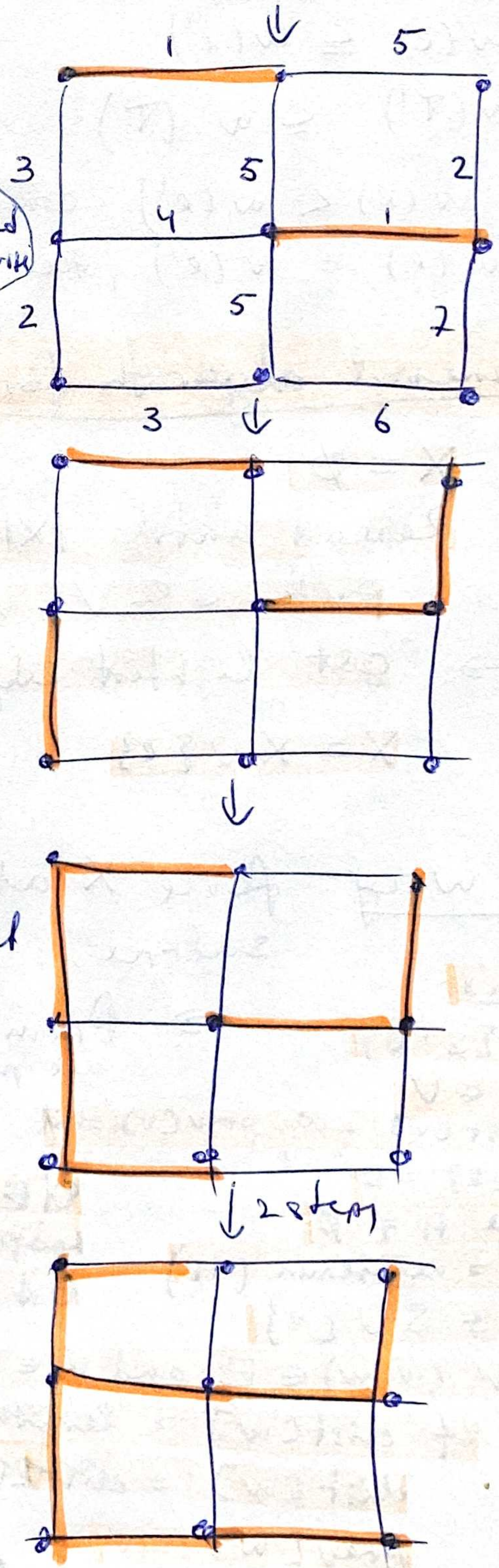
~~G:3~~, D:4, B:5

~~D:4~~, B:5, H:6

Prim: like Dijkstra, distance notion is different
not path length (sum of previous edges)
but connection length to the subtree

Kruskal's alg: second way

sort edges
in increasing order of edges
if the edge does not add a cycle } implicit cut
add it



Proof

[redacted] since it does not create a cycle

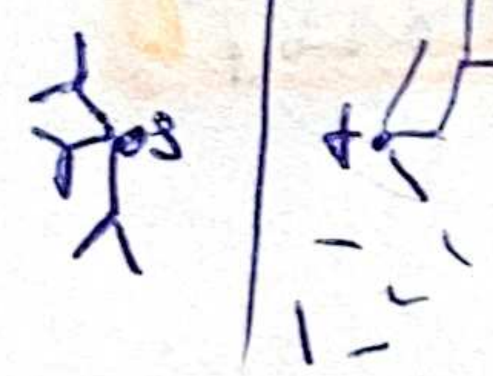
no cycle and $|V|-1 > |E| \Rightarrow$ not connected G

it crosses a cut

connect disconnected parts, otherwise cycle

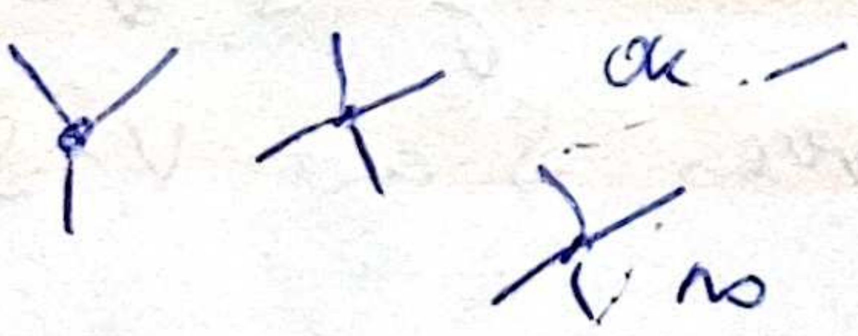
and

it is [redacted] a smallest edge
to cross this cut



$\Rightarrow$ cut property satisfied

checking if no cycle:



disjoint set data structures

sets: components of vertices

① → are 2 [redacted] elts in the same set

② → replace 2 sets by their union

baseline with arrays of vertices:

v	1	2	3	—	n
set	1	2	2		3

① $O(1)$

② $O(n)$ ← bad for Kruskal