Shortest Paths in Weighted Graphs

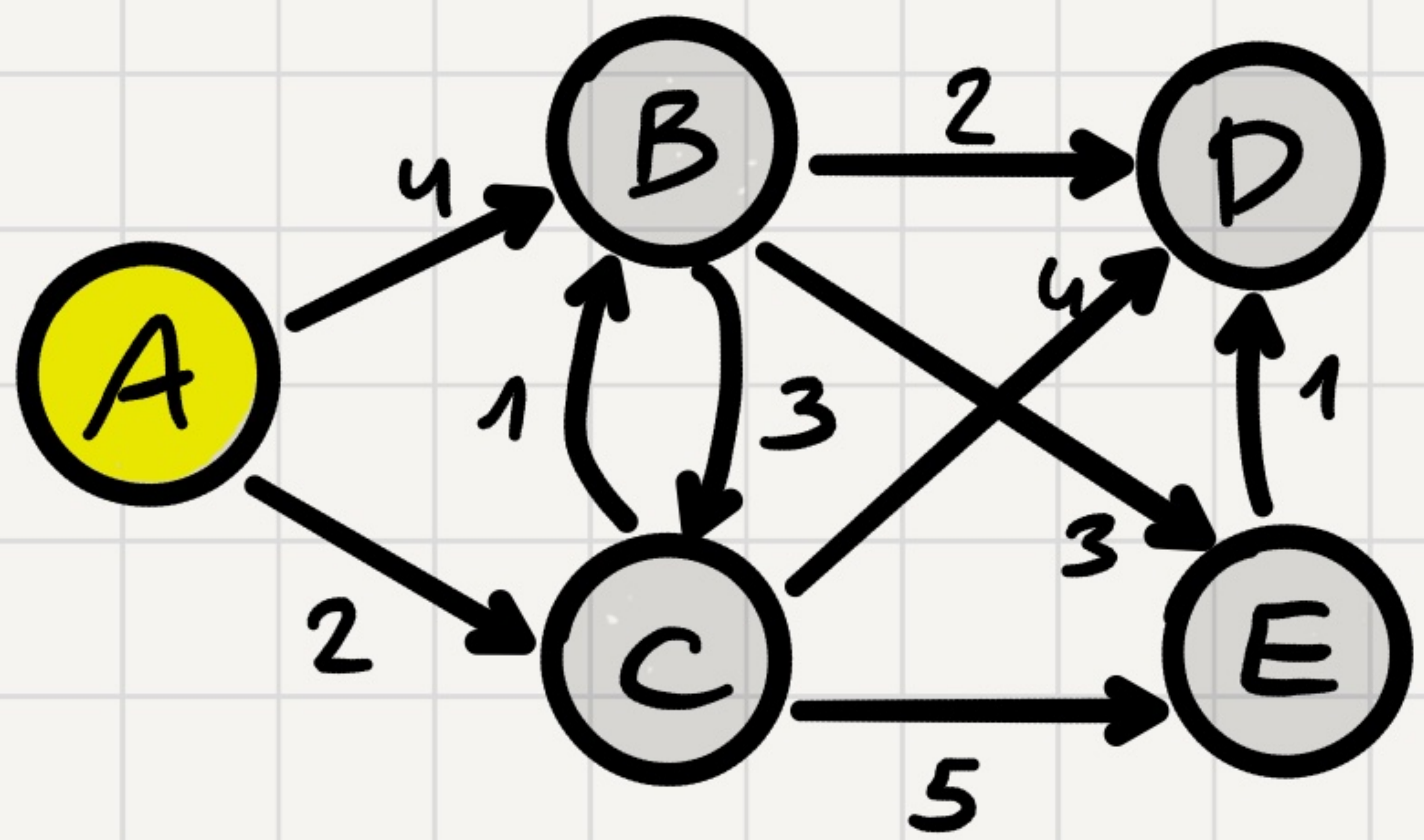
Recall: Last time we showed that BFS finds shortest paths in unweighted graphs.

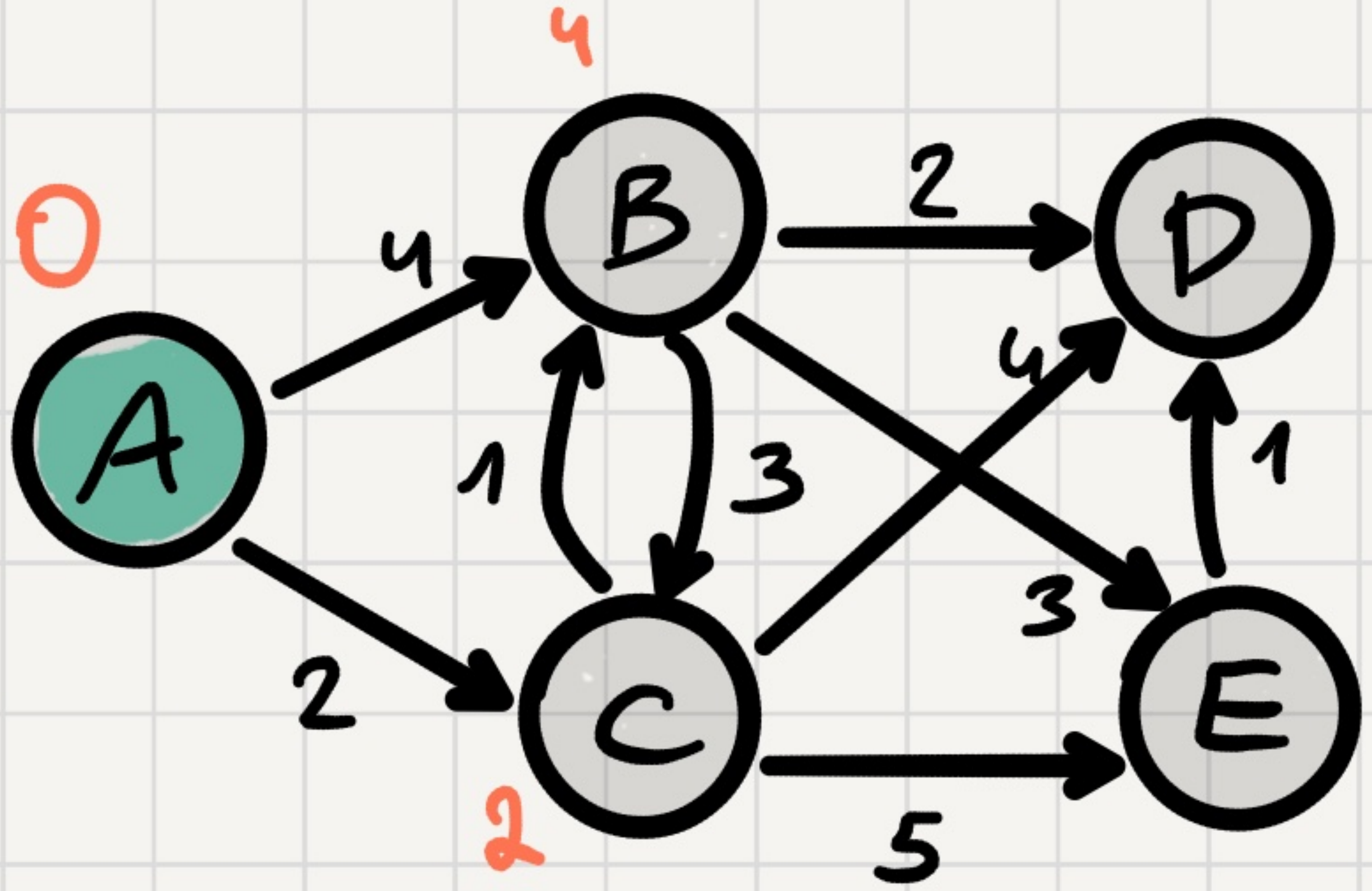
Today:
1. Dijkstra's Algorithm: Positive Weights
2. Bellman-Ford Algorithm: Arbitrary Weights
3. Detecting Negative Cycles.
4. Shortest Paths in DAGs.

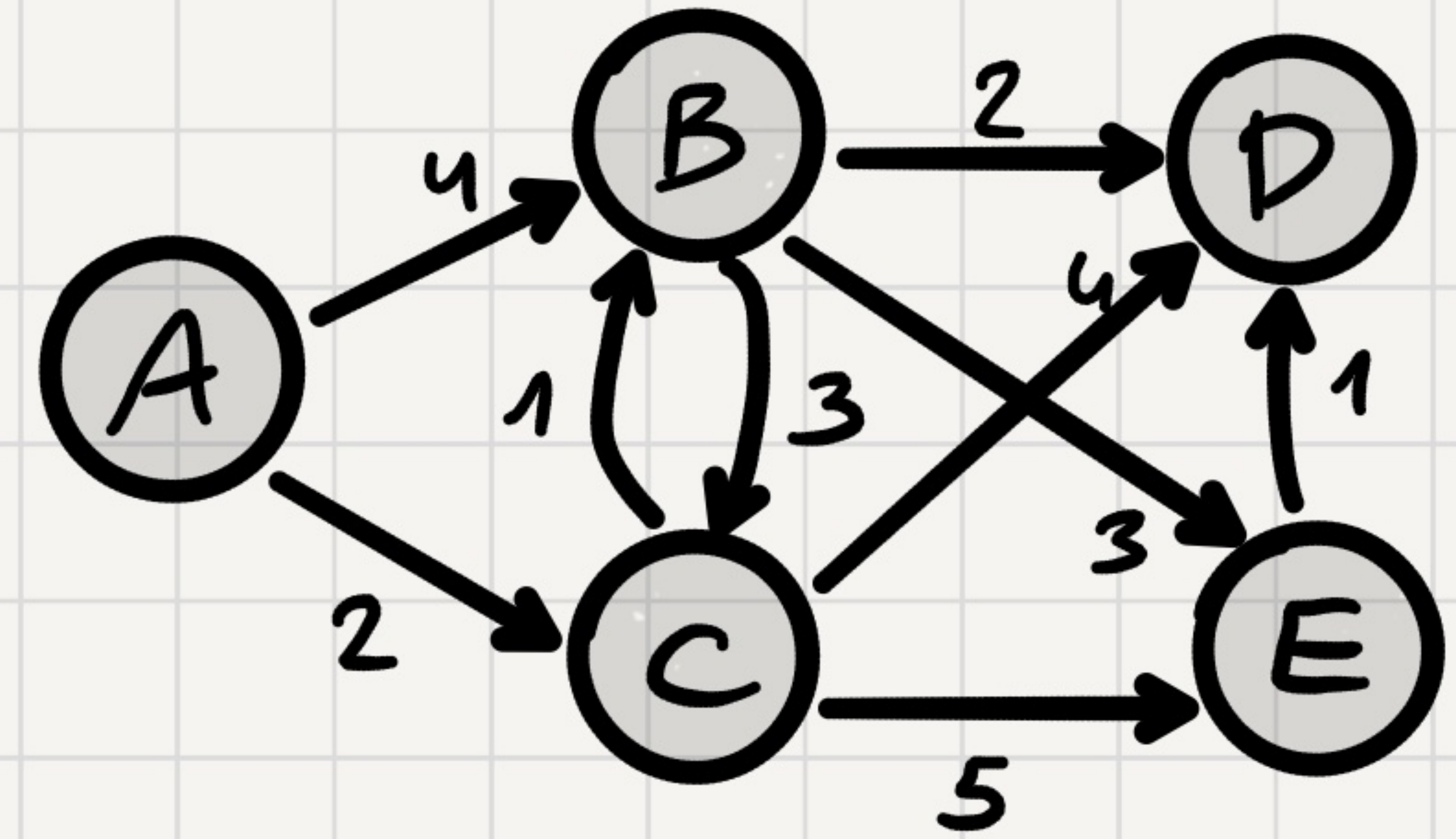
1. $G=(V,E)$ $\quad \underline{\ell}: E \to \mathbb{N}$.

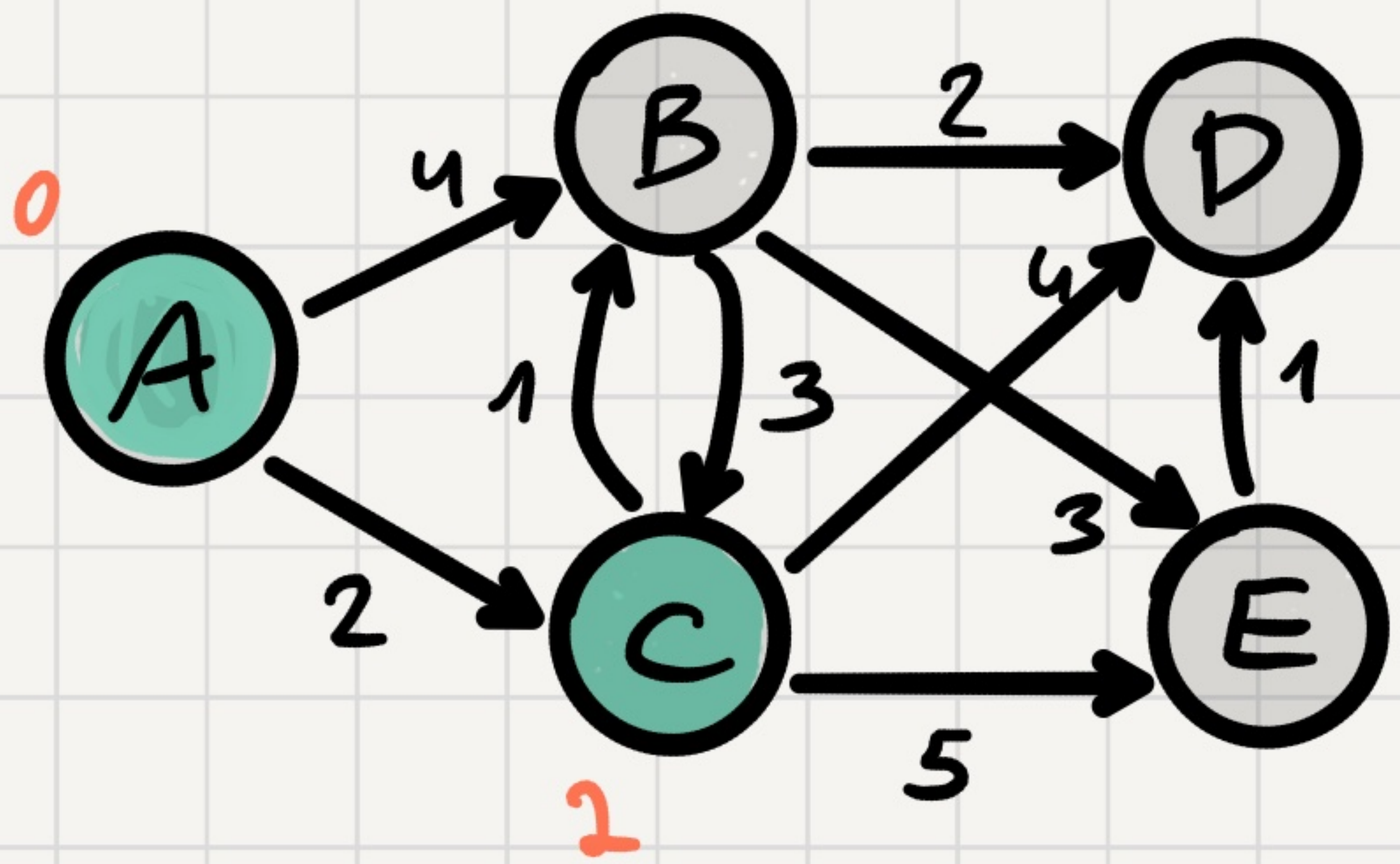
Example:

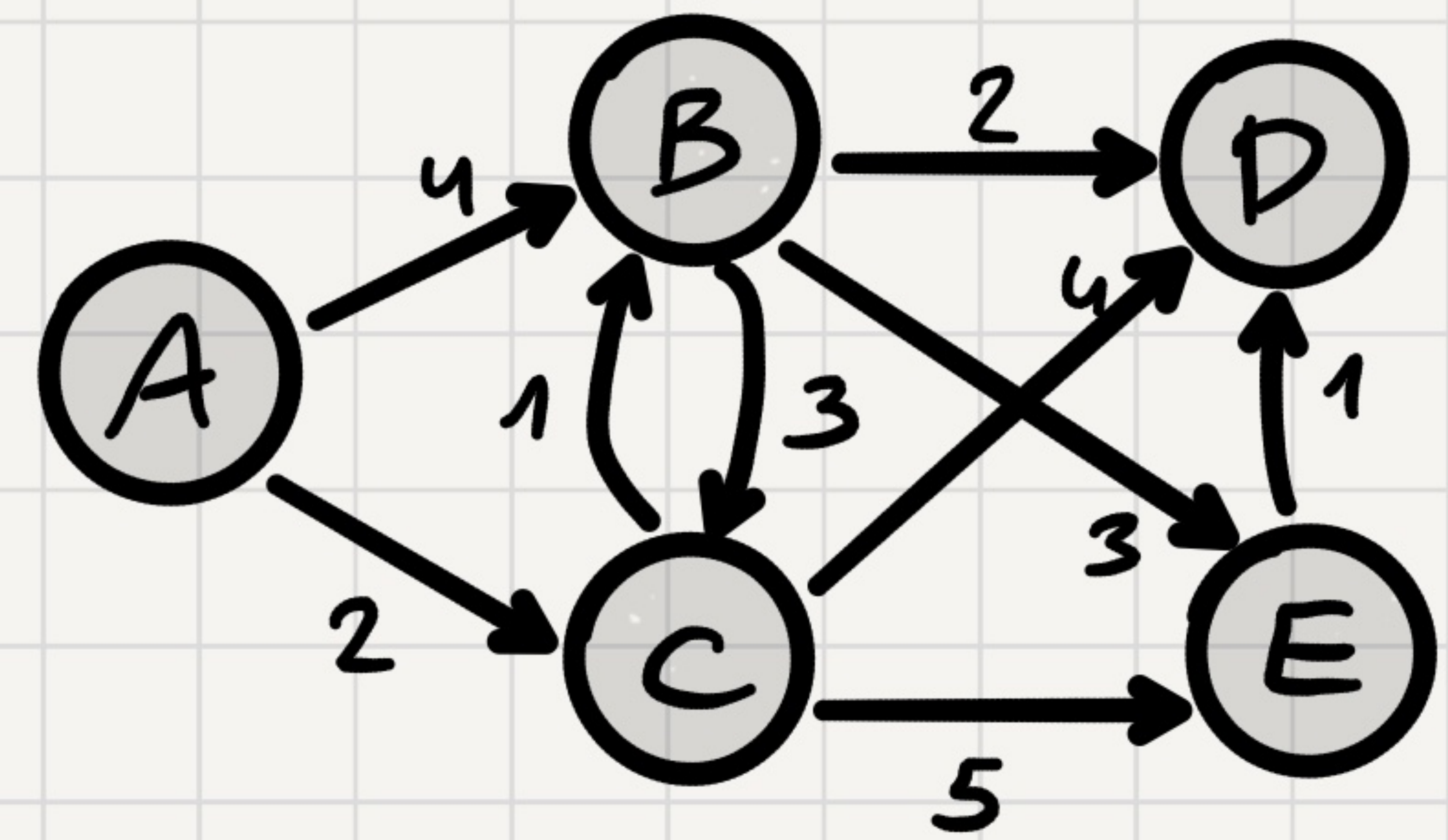
find shortest paths from A.

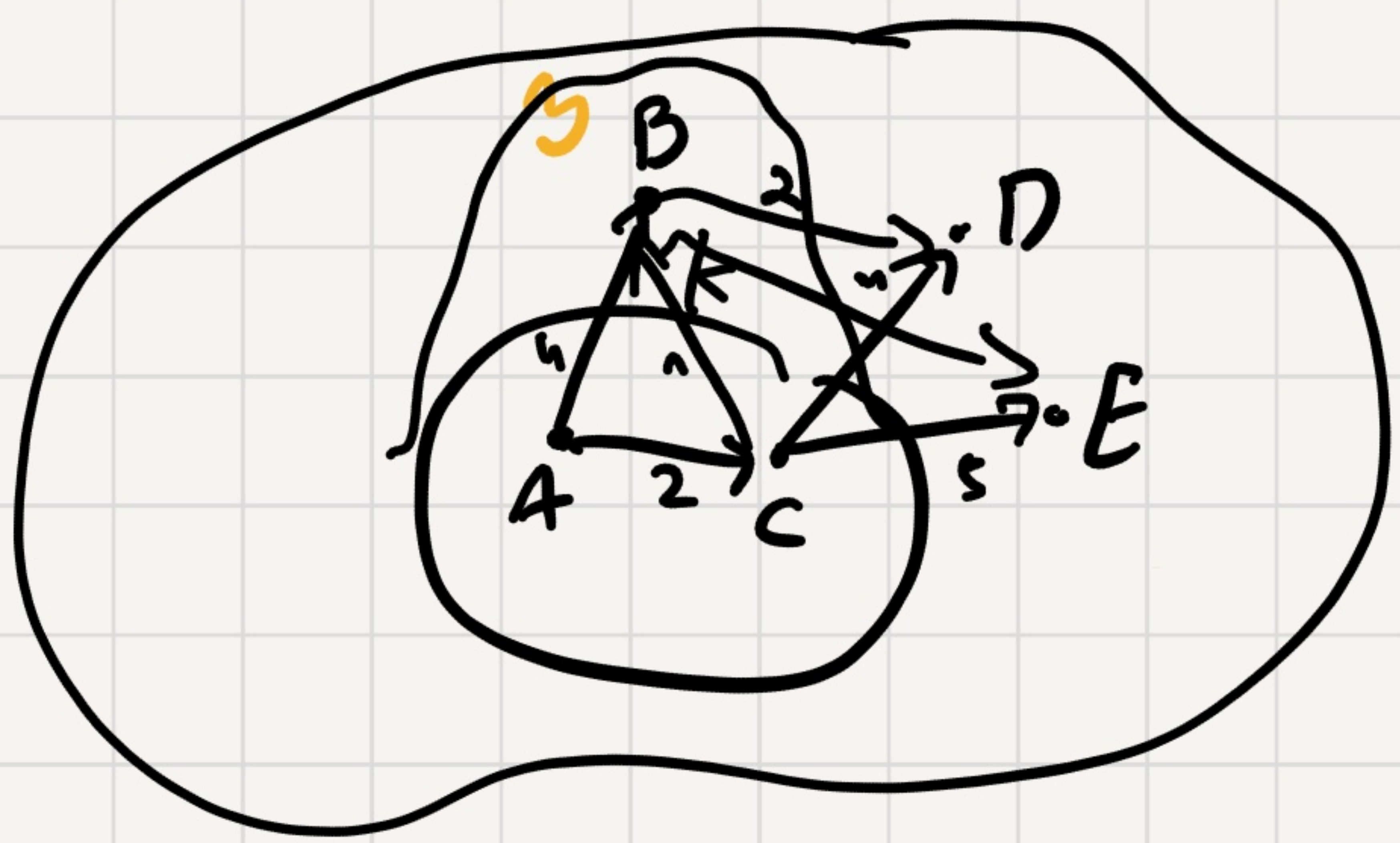












K - the set of vertices for which we know the shortest paths fo

How to extend K?

Dijkstra(G, s):

- dist[s] = 0
- $\forall v \neq s$ dist[v] = ∞.
- U = V   Initialize a Priority Queue $Q \leftarrow V$ with keys = dist.
- while $U \neq \emptyset$.
  - $u \leftarrow$ DeleteMin(Q)

update $\left\{ \begin{array}{l} \bullet \text{ for } (u,v) \in E: \\ \quad \text{dist}[v] = \min(\text{dist}[v], \text{dist}[u] + \ell(u,v)). \end{array} \right.$

DecreaseKey(Q, v, dist[v]).

How to implement?   Binary Heap { DeleteMin, DecreaseKey, Insert } $O(\log |V|)$.

Fibbonaci Heap

Running Time

of Operations:

- MakeQueue: once. $O(|V|)$ or $O(|V|\log|V|)$.
  complicated. by $|V|$ inseration
- DeleteMin: $|V|$.
- Decrease key: at most $|E|$ times.

overall Runtime: $O((|V|+|E|)\log|V|)$. using binary heap.

$O(|V|\log|V| + |E|)$ using Fib heaps

Claim: At any point in time, $\forall v \in K$ $\quad dist[v] = d(s,v)$.

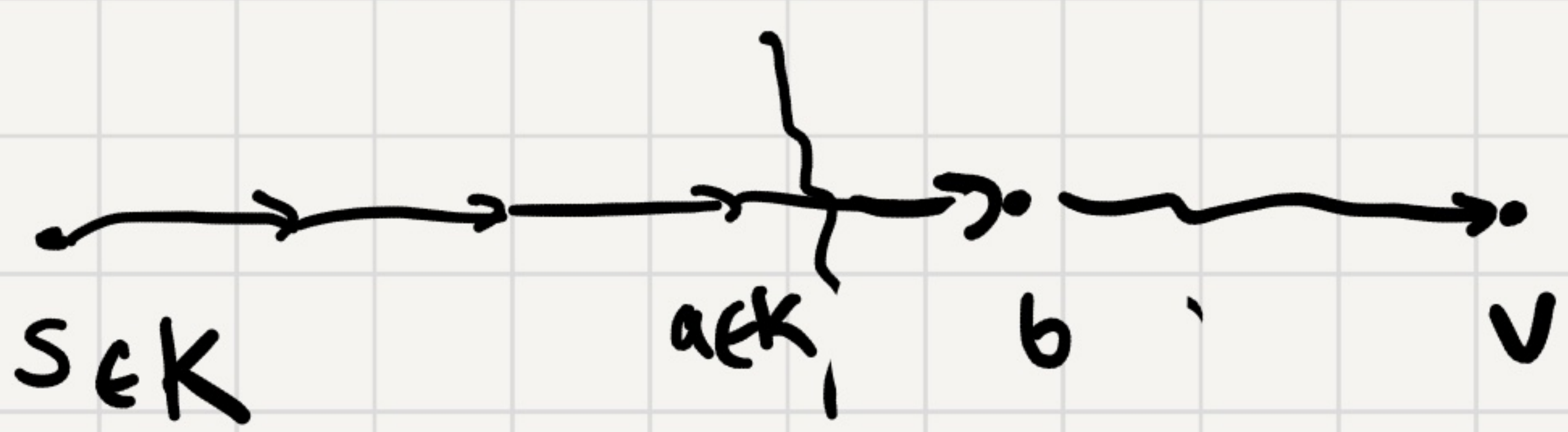
Proof: By Induction.

Base case: Trivial.

In the second step $K = \{s\}$ trivial.

Step: Let v be the vertex with smallest dist number

We show that $dist[v] = d(s,v)$.



$dist[a] = d(s,a)$

$dist[b] \le dist[a] + \ell(a,b)$.

$\le d(s,a) + \ell(a,b)$

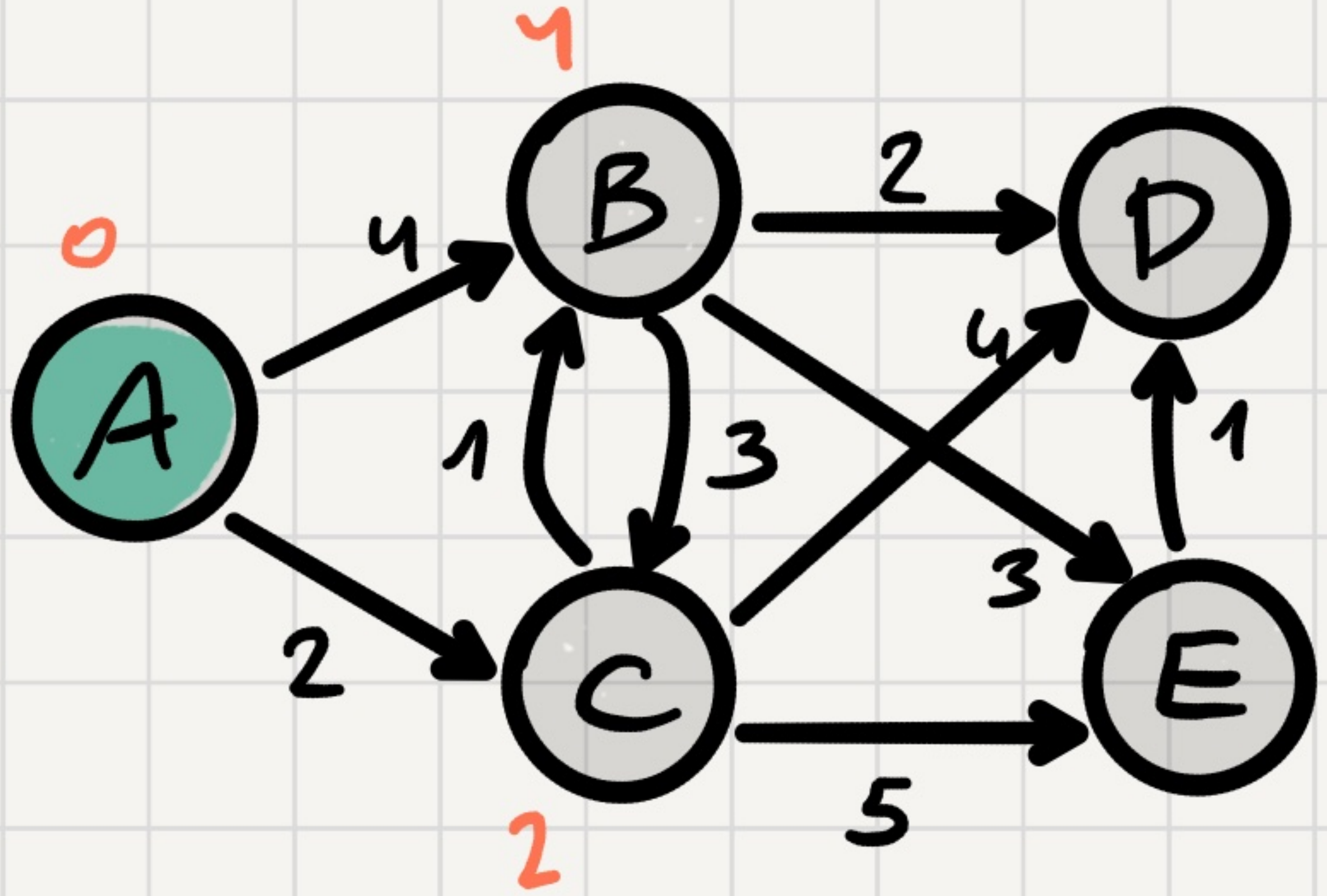
$= d(s,b)$.

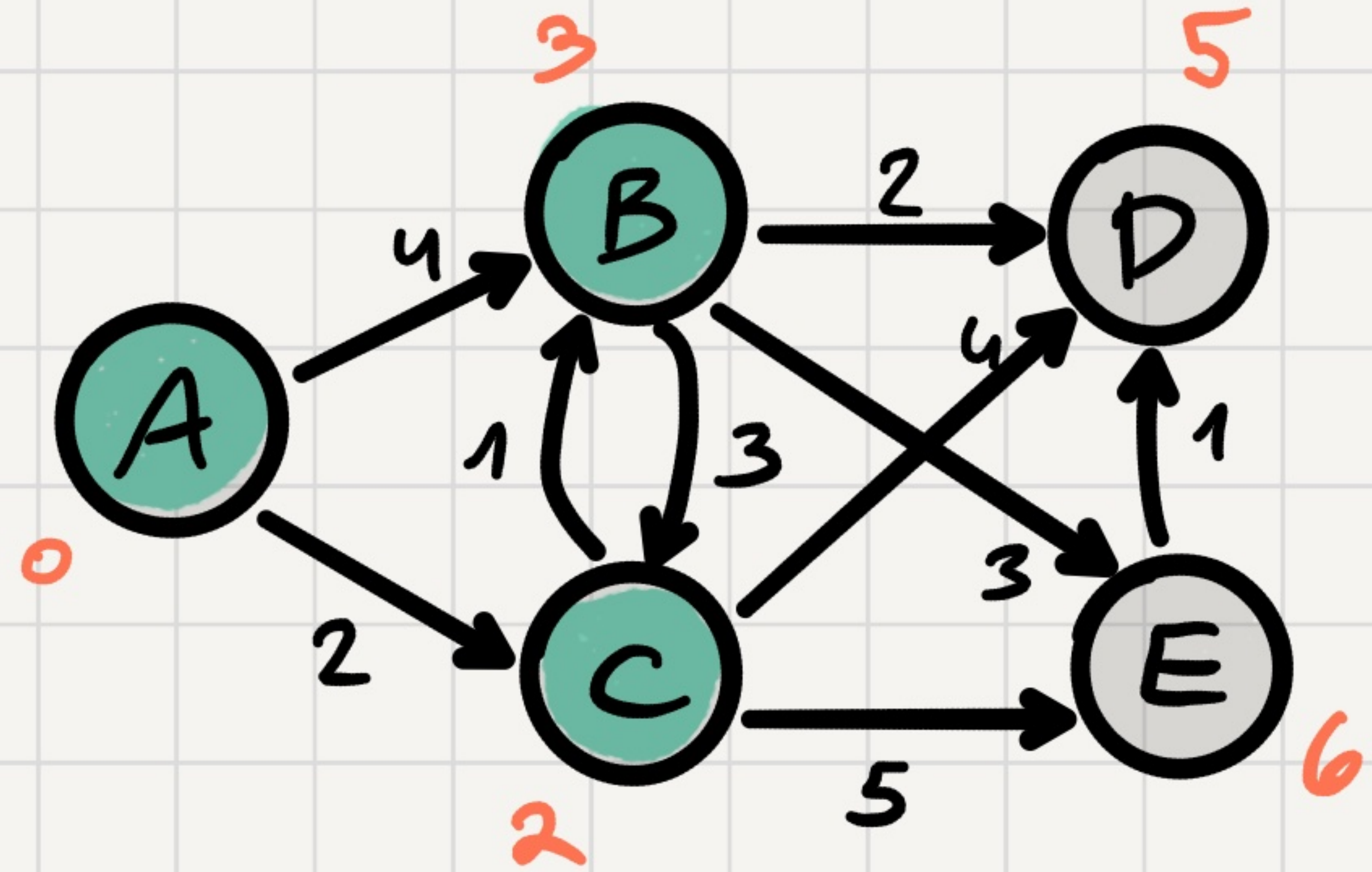
$\forall (a,v') \in E$

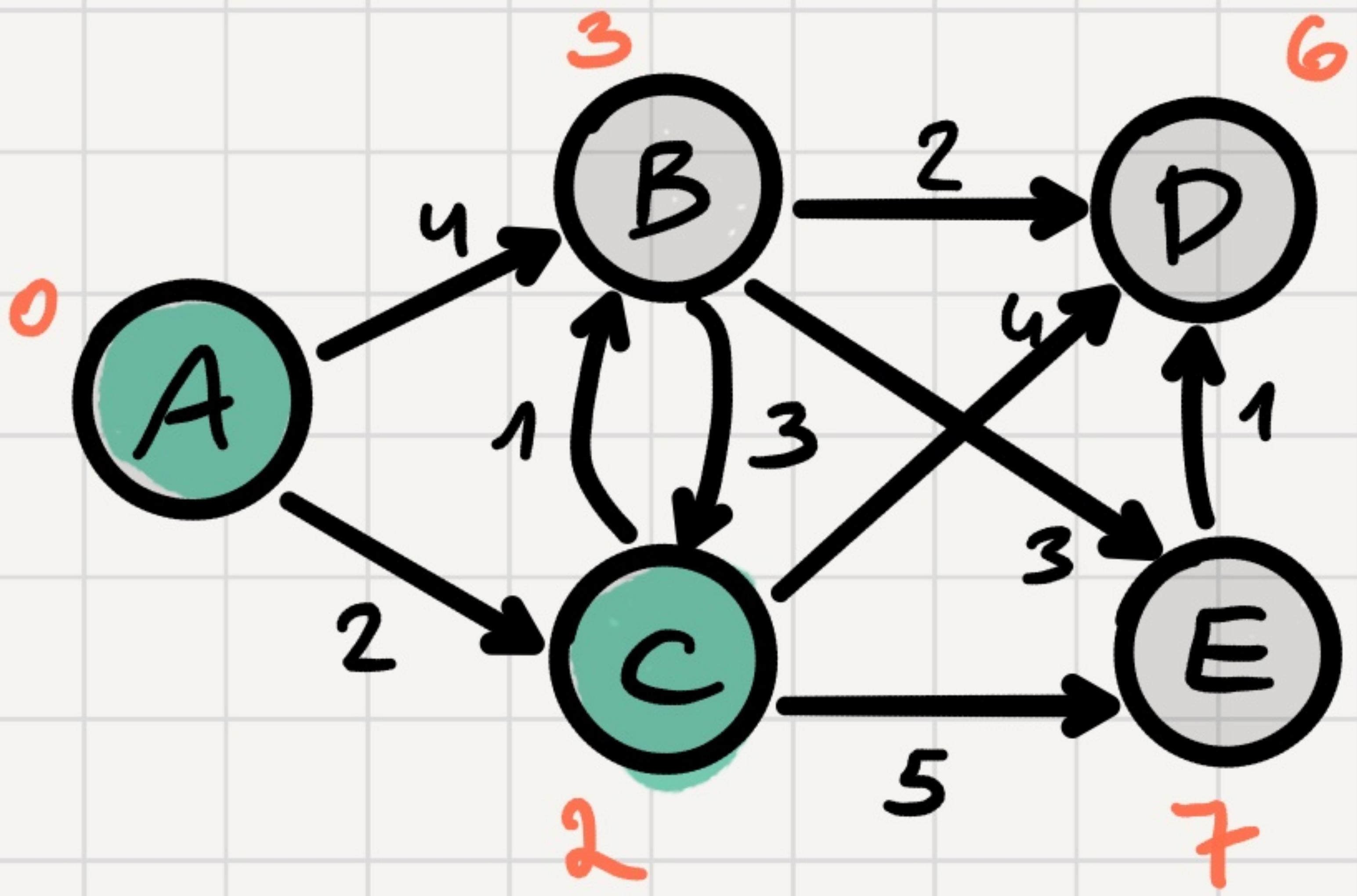
$dist[v'] = \min(dist[v'], dist[a] + \ell(a,v'))$

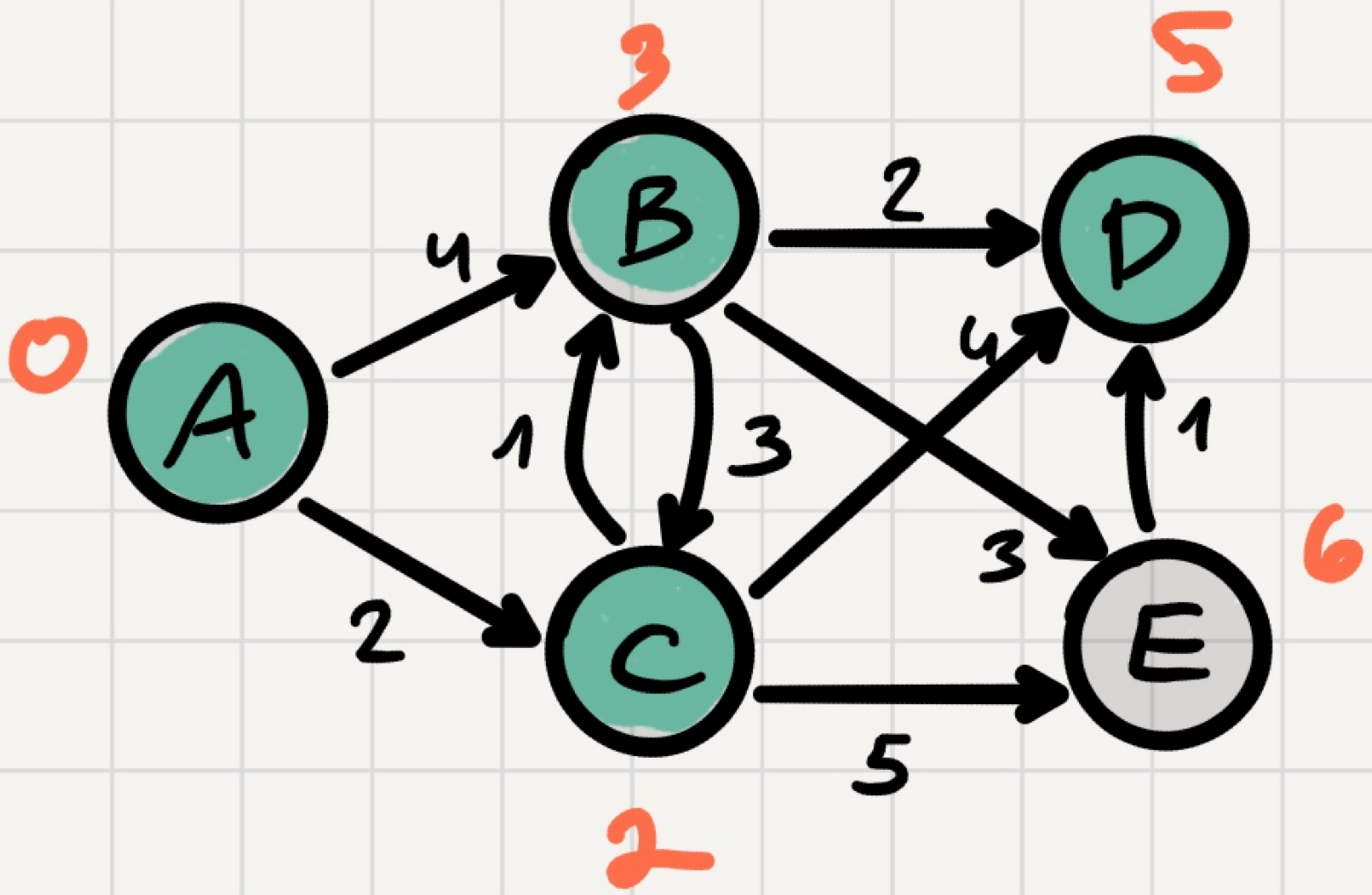
$dist[v'] \le \qquad dist[a] + \ell(a,v')$.

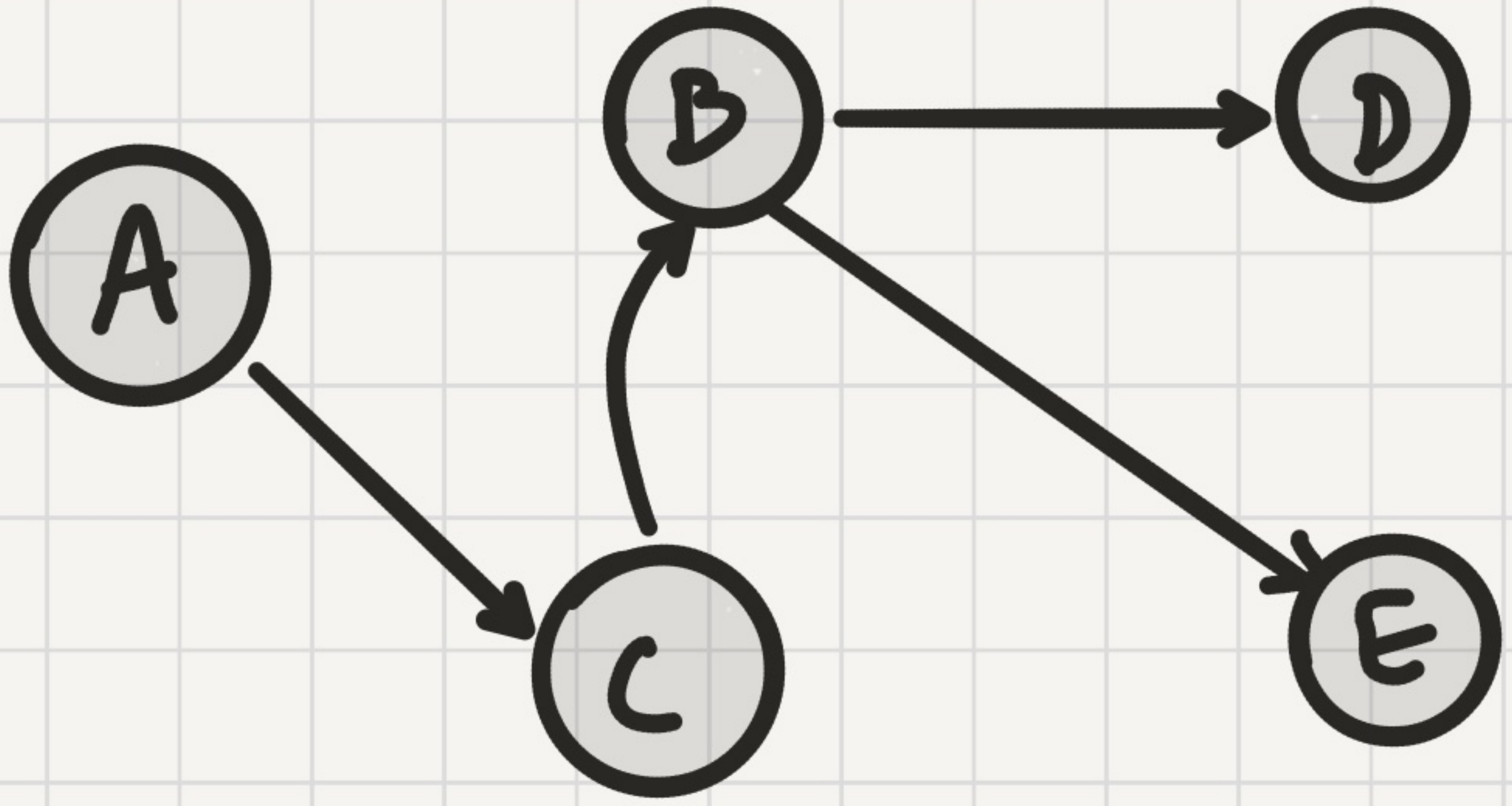
$b \ne v \Rightarrow dist[b] < dist[v]$ contradiction.







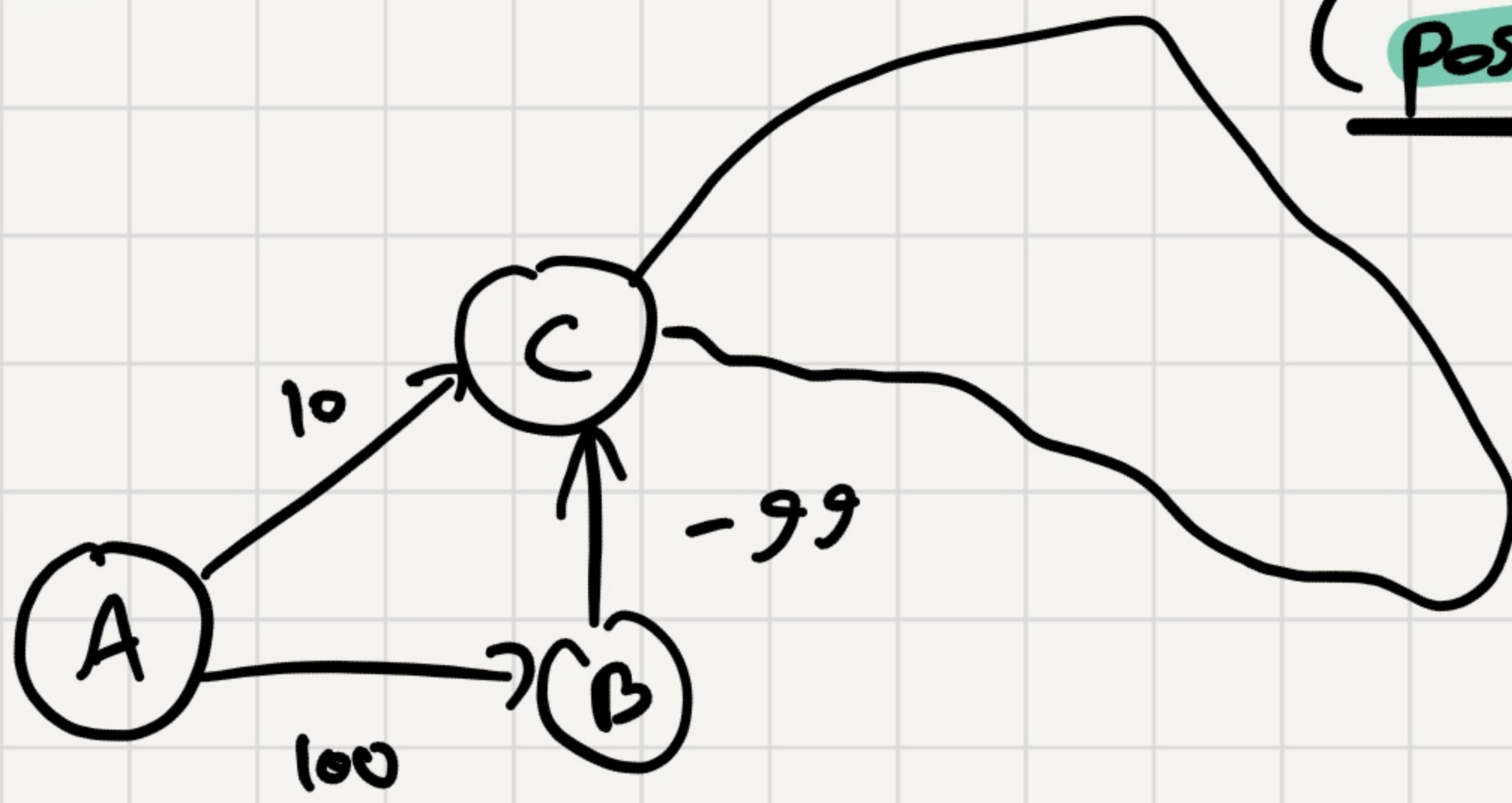




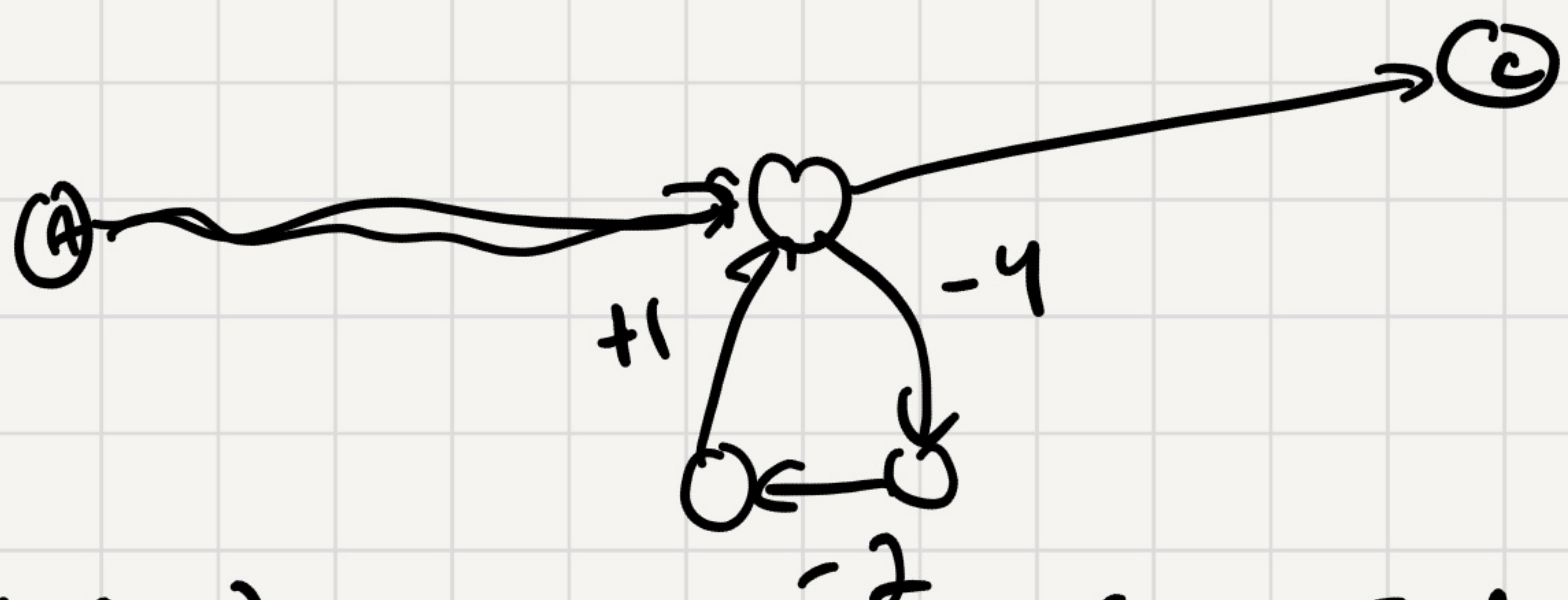
Dijkstra's tree.

Shortest Path with Arbitrary Lengths

(positive or negative)



Does it make sense?



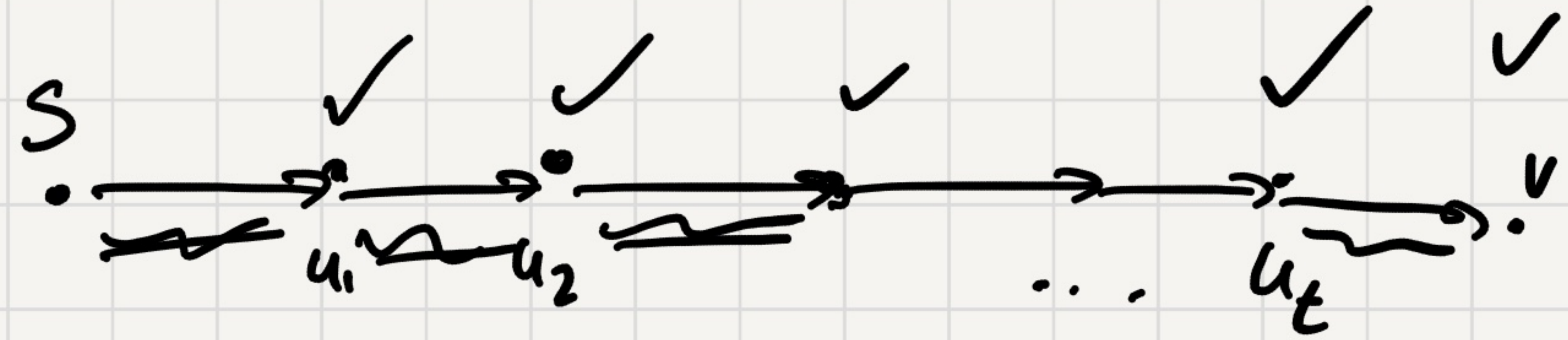
<u>update(u,v)</u>: $\quad dist[v] = \min(dist[v], dist[u] + \ell(u,v))$

① Update is Safe. $\quad \forall v: \; dist[v] \geq d(s,v).$

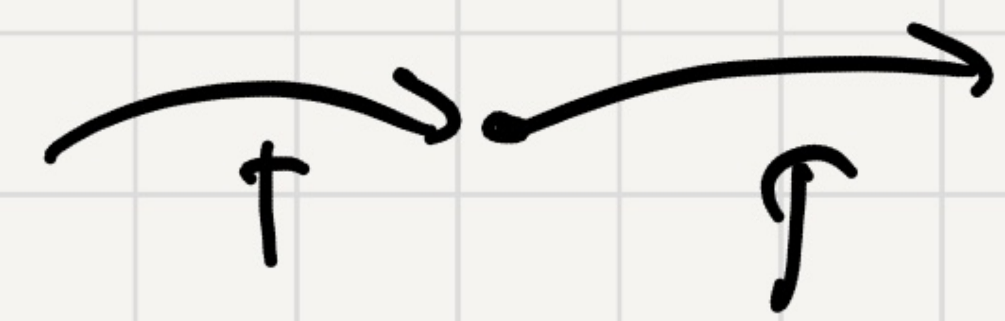
② If shortest path from s to v looks like that:

$$s \longrightarrow \longrightarrow \longrightarrow \longrightarrow u \longrightarrow v$$

dist[u] is correct
update(u,v) $\rightarrow$ dist[v] is correct.



$$\left.\begin{array}{l} \to \text{update}(s, u_1) \\ \to \text{update}(u_1, u_2) \\ \to \text{update}(u_2, u_3) \\ \quad \vdots \\ \text{update}(u_{t-1}, u_t) \\ \text{update}(u_t, v) \end{array}\right\} \Rightarrow \text{dist}[v] \text{ is correct.}$$



Bellman Ford:

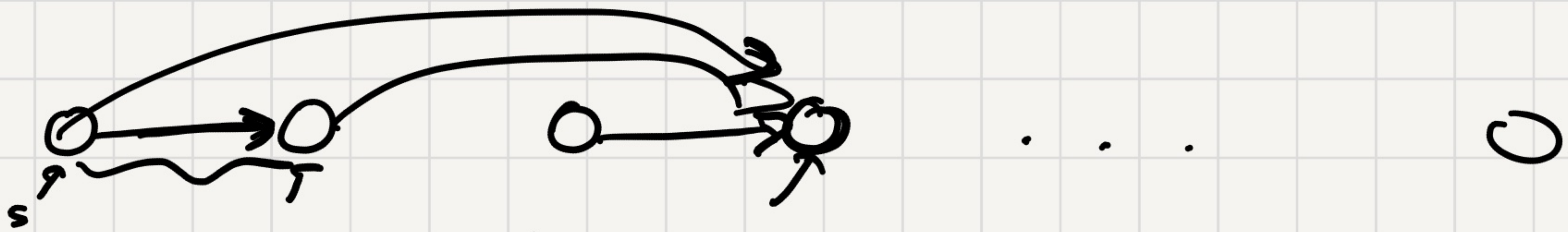
For $i = 1, \ldots, |V|-1$:

- update all edges (for all $(u,v) \in E$ update(u,v))

Running Time: $O(|V| \cdot |E|)$ steps.

Shortest Paths in DAGs.

Can we find shortest paths in DAGs using Bellman-Ford faster?



1. Find topological order on G.
2. For all edges (sorted according to the topological order)

   update(u,v).

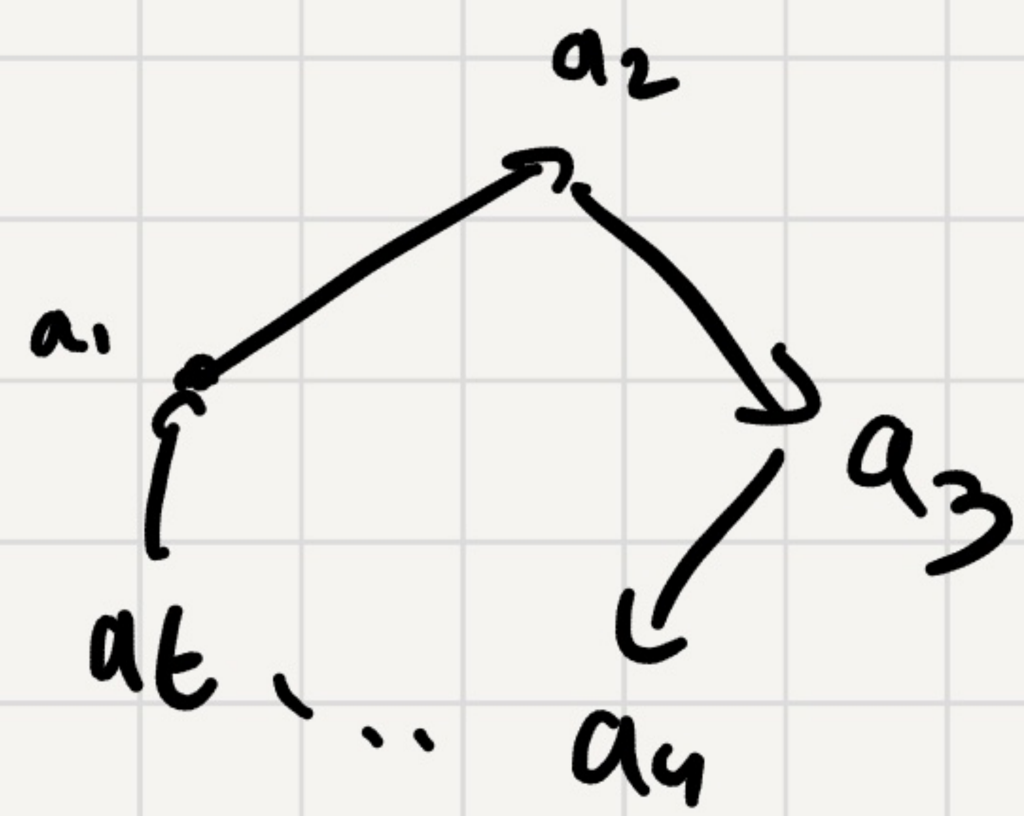
$O(|V|+|E|)$

Detect Negative Cycles?

No negative cycles $\Rightarrow$ running Bellman-Ford for one more iteration would not change any dist.

$\Leftarrow$

Assume that in the last iteration
there were no updates

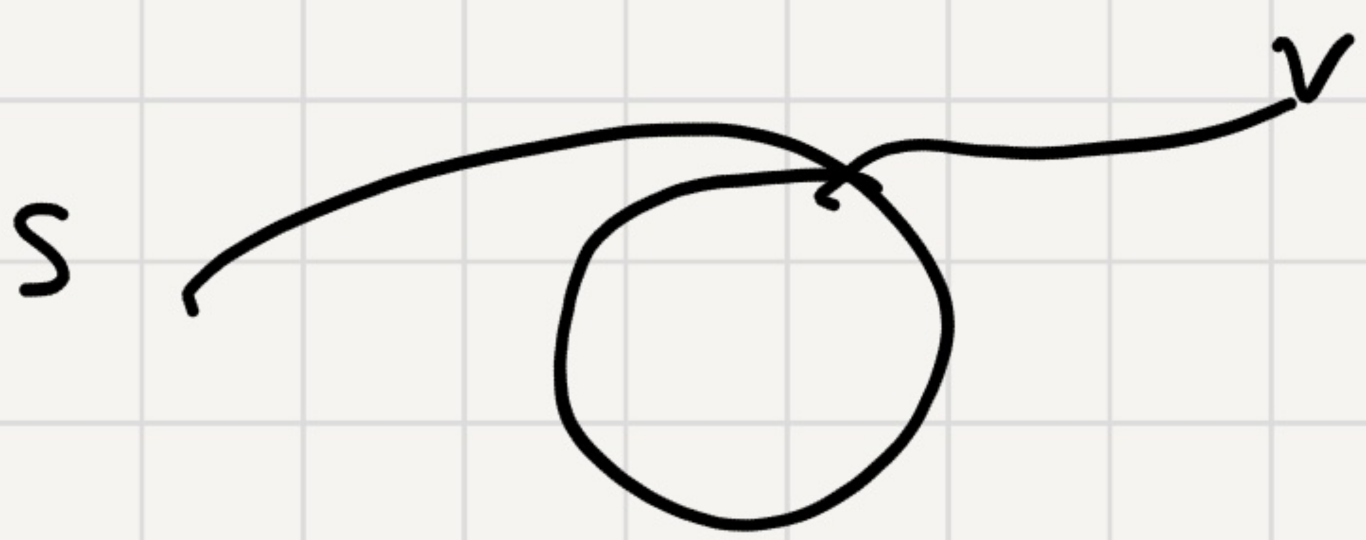


- $dist[a_1] \le dist[a_t] + \ell(a_t, a_1)$
- $dist[a_2] \le dist[a_1] + \ell(a_1, a_2)$

$\vdots$

- $dist[a_t] \le dist[a_{t-1}] + \ell(a_{t-1}, a_t)$

$$dist[a_1] + \cancel{\ldots + dist[a_t]} \le \cancel{dist[a_1] + \ldots + dist[a_t]} + \ell(a_t, a_1) + \ell(a_1, a_2) \ldots + \ell(a_{t-1}, a_t)$$



$\Rightarrow$ every cycle is non-negative.