CS 580
ALGORITHM DESIGN AND ANALYSIS

Greedy Algorithms 1:
Shortest Paths (cf. KT 4.4, 6.8)

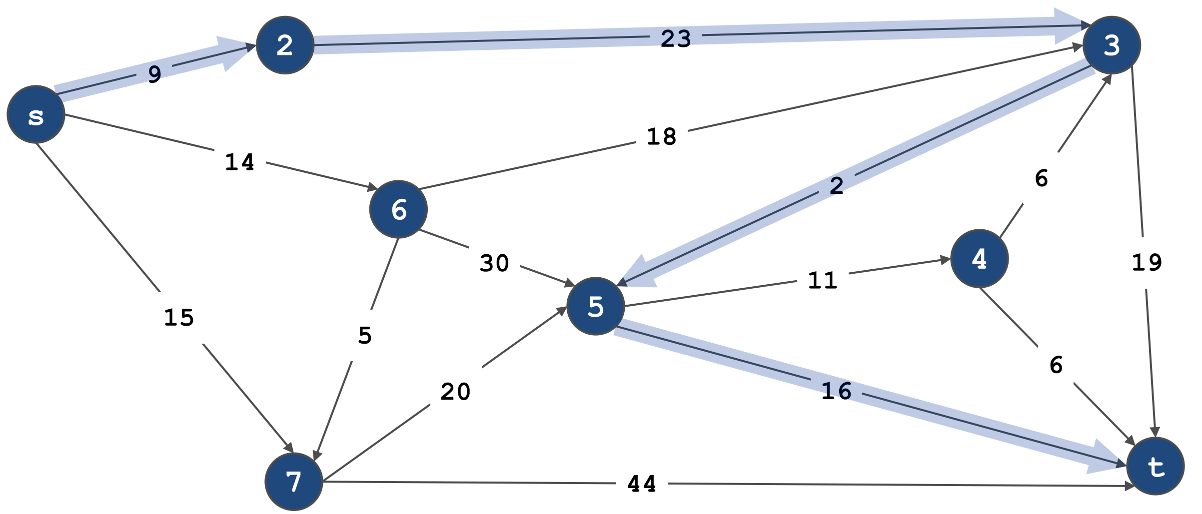
Vassilis Zikas

GREEDY ALGORITHMS

- Build a solution by myopically doing the best thing you can
  - There is a local structure that can be exploited to give global optimality
- Non-trivial problems solved by greedy algorithms are few and far between

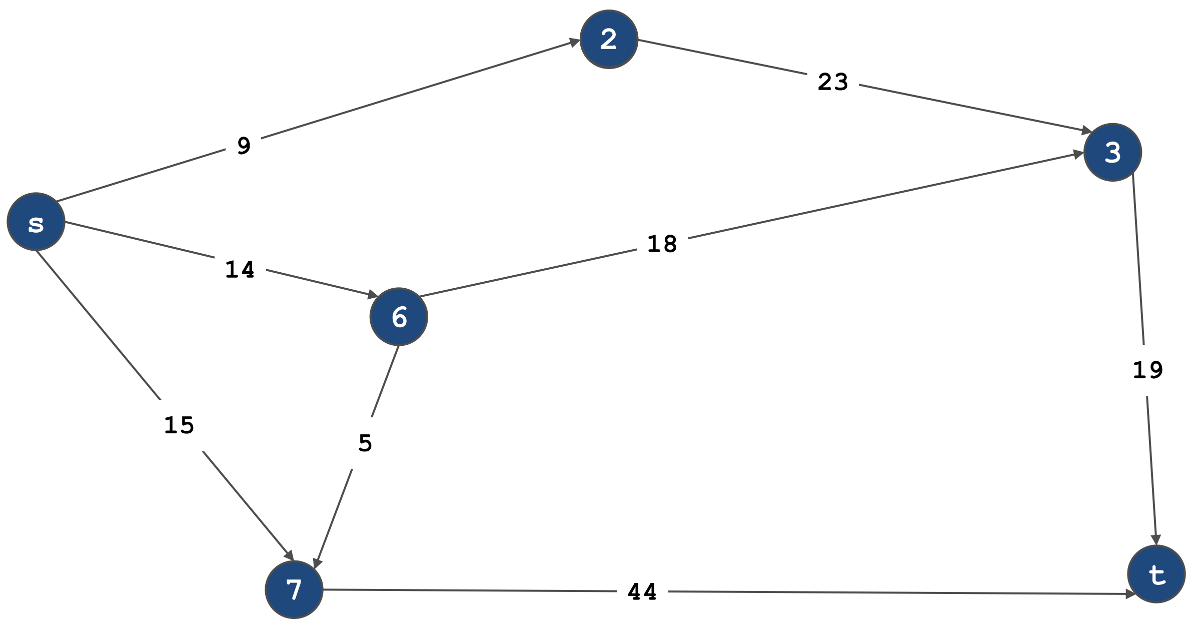
SHORTEST PATH

- *Problem:* Given a directed graph $G = (V, E)$ and two node $s, t,$ what is the length of the shortest path from s to t?
- Length?
- If we're just counting the number of edges, we already know how to do this!
  - Breadth First Search!
  - Would DFS work?
- More general version: each edge e has a length $\ell_e \geq 0$



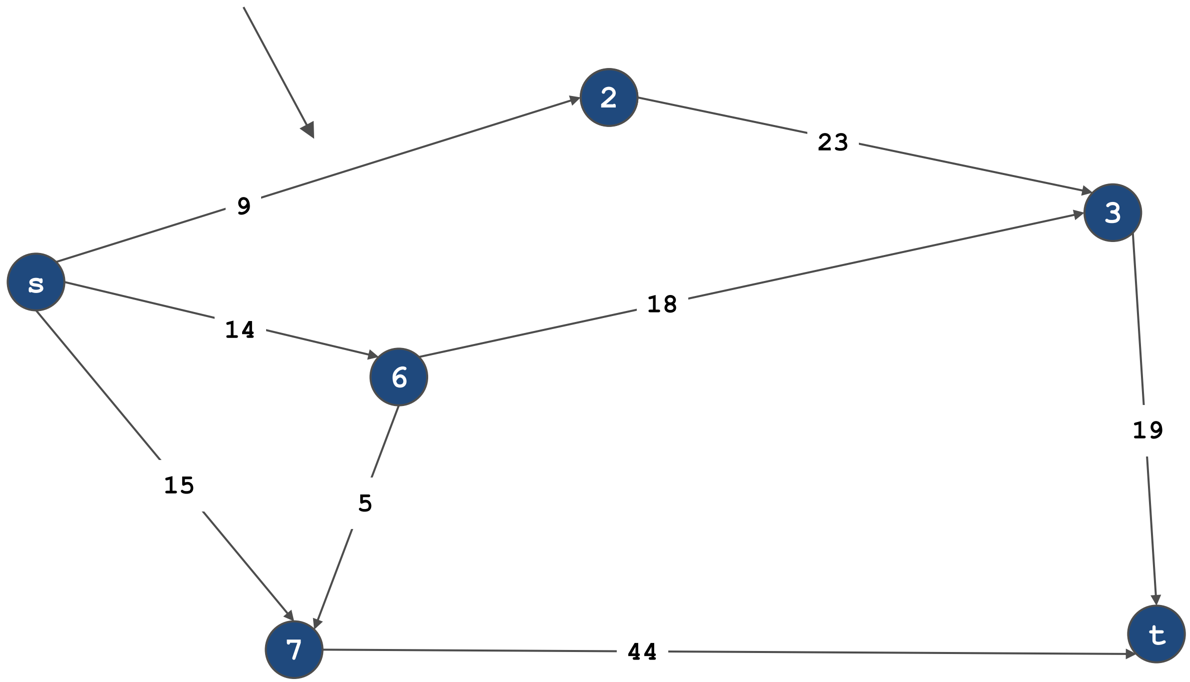
s
2
3
4
5
6
7
t
9
23
14
18
15
30
2
6
11
19
5
20
16
6
44

SHORTEST PATH: A REDUCTION

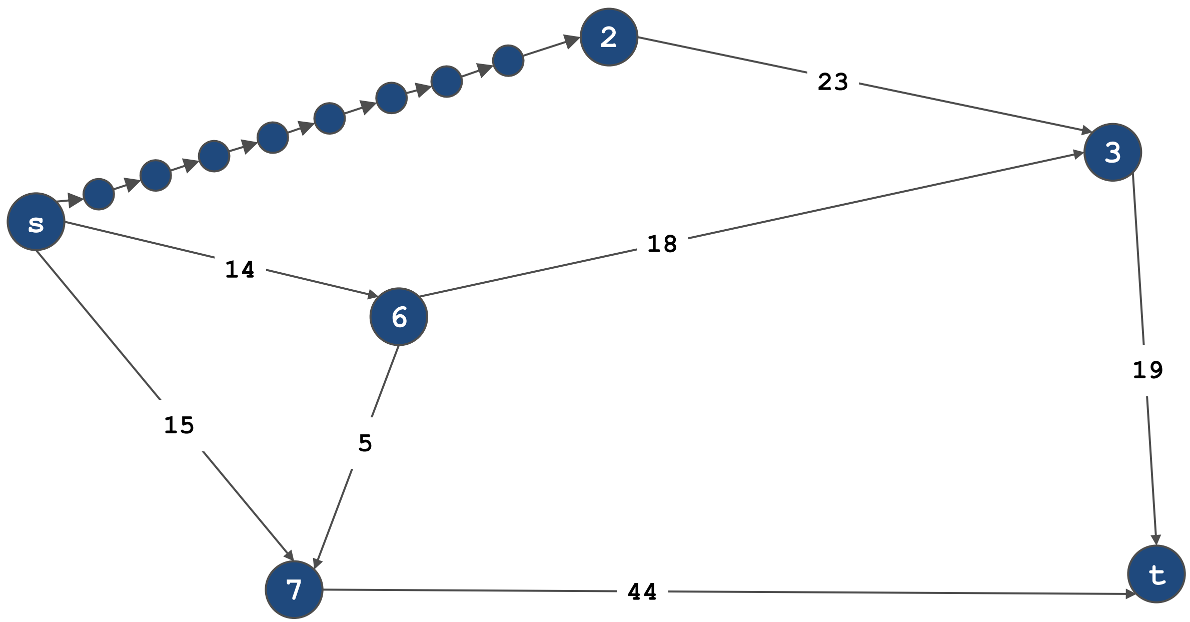


SHORTEST PATH: A REDUCTION

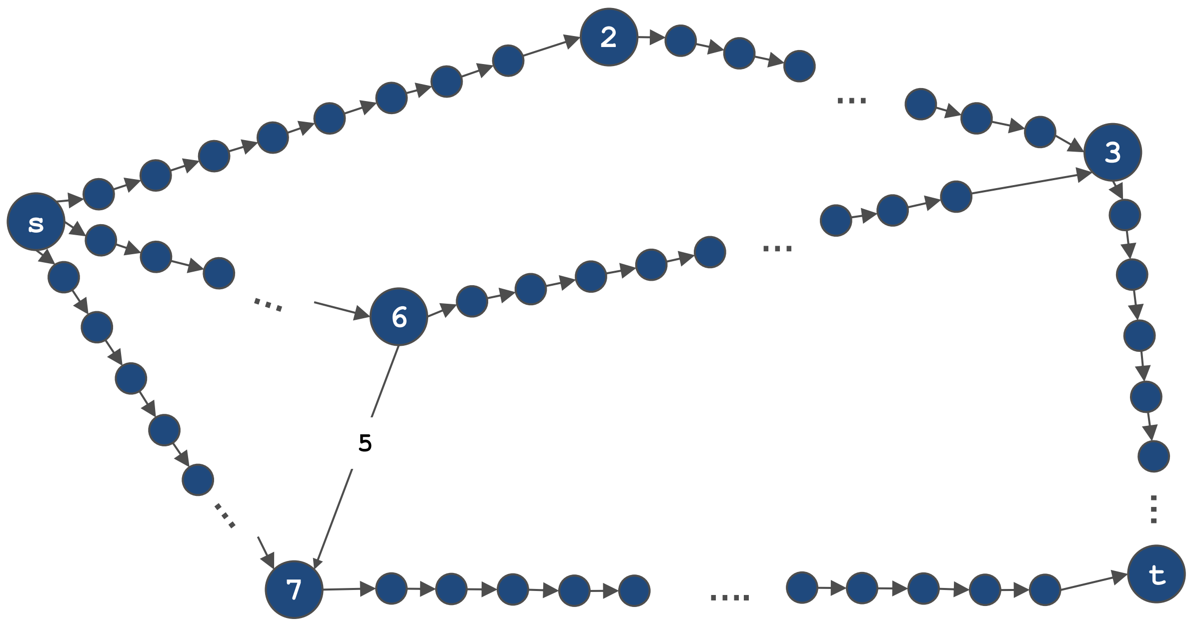
Replace with 8 vertices



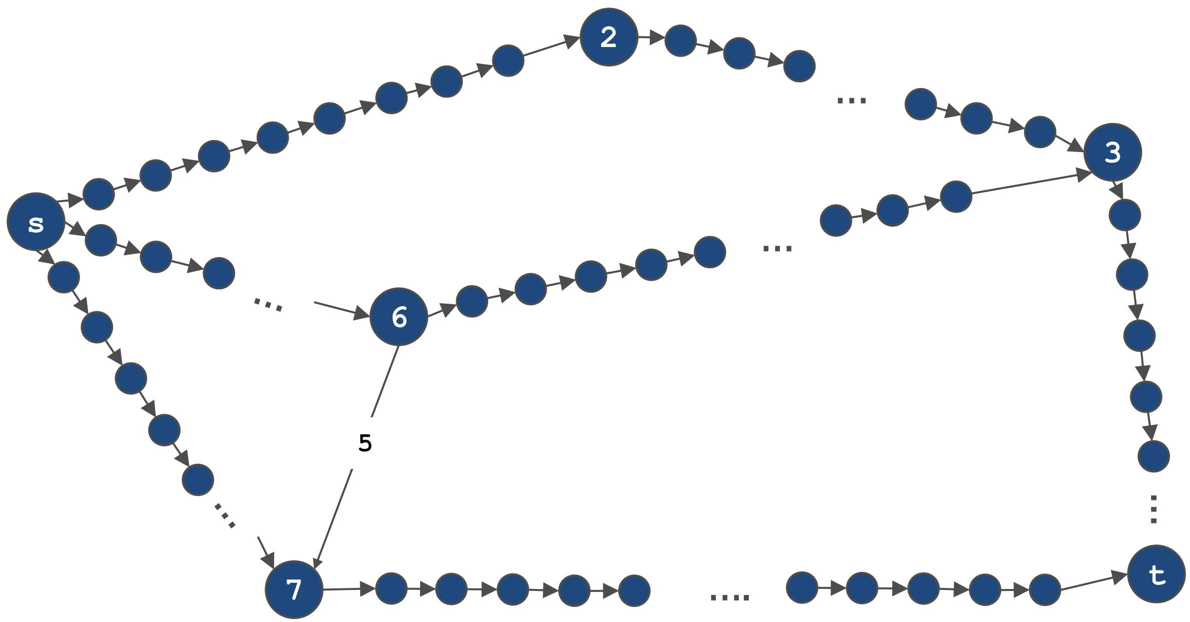
SHORTEST PATH: A REDUCTION



SHORTEST PATH: A REDUCTION

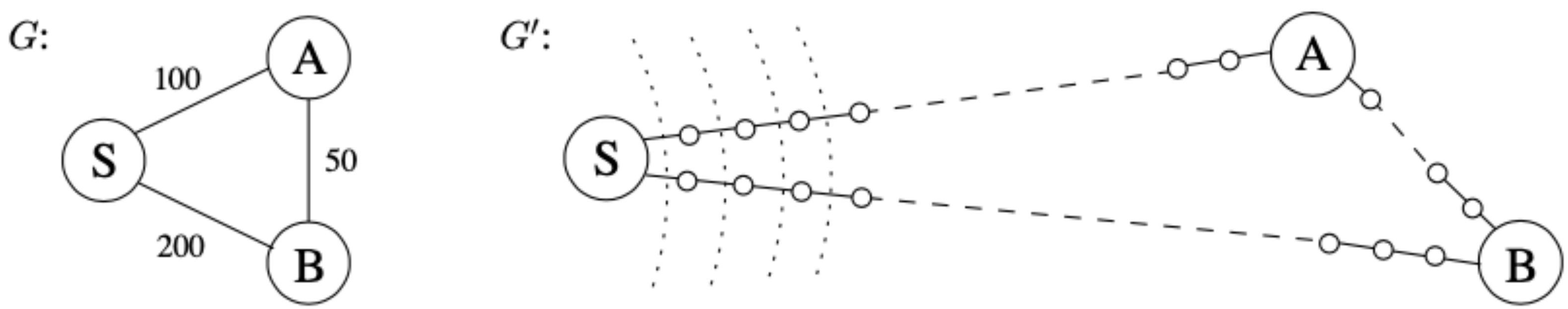


SHORTEST PATH: A REDUCTION



Now run BFS!

SHORTEST PATH



SHORTEST PATH

- Run time?
- The number of nodes in the new graph depends on the length of the edges in the original graph
  - With n-bits we can write the number 2^n!
- Almost there…

SHORTEST PATH

Dijkstra's algorithm

Instructor's favorite quote:
The question of whether computers can think is like the question of whether submarines can swim

SHORTEST PATH: DIJKSTRA'S ALGORITHM

- In the previous reduction, think of an alarm going off every time BFS reaches a "real" vertex
- Nothing interesting can possibly happen between alarms!
- Main idea: simulate alarms (with a shortcut, in order to avoid exponential time)

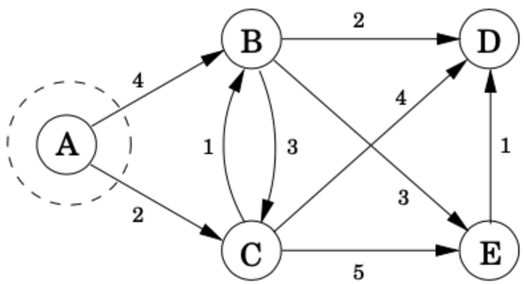
SHORTEST PATH: DIJKSTRA'S ALGORITHM

1. Maintain a set of explored vertices S
   - For each $u \in S$ we have determined the shortest path distance $d(u)$ from s
   - These are the vertices whose "alarm" has gone off
2. Initialize $S = \{s\}$ and $d(s) = 0$
3. Repeatedly choose an unexplored node v that minimizes

$$\pi(v) = \min_{e=(u,v):u\in S} d(u) + \ell_e$$

   - Add v to S and set $d(v) = \pi(v)$

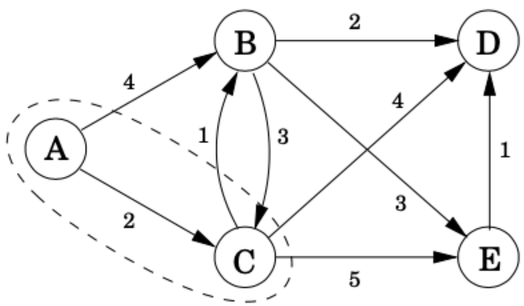
SHORTEST PATH: DIJKSTRA'S ALGORITHM



Current estimated distance $\pi(v)$

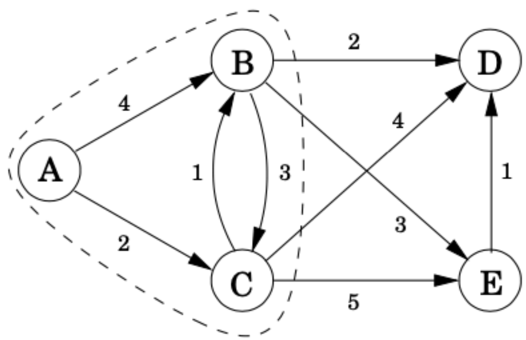
A: 0	D: ∞
B: 4	E: ∞
C: 2	

SHORTEST PATH: DIJKSTRA'S ALGORITHM



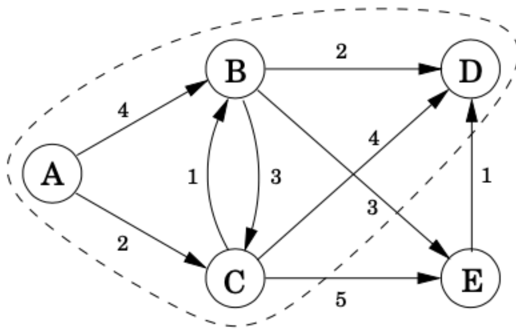
A: 0	D: 6
B: 3	E: 7
C: 2	

SHORTEST PATH: DIJKSTRA'S ALGORITHM



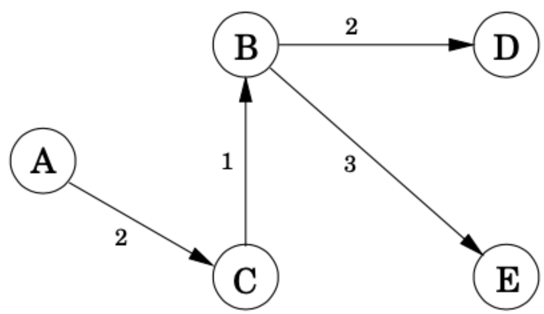
A: 0	D: 5
B: 3	E: 6
C: 2	

SHORTEST PATH: DIJKSTRA'S ALGORITHM



A: 0	D: 5
B: 3	E: 6
C: 2	

SHORTEST PATH: DIJKSTRA'S ALGORITHM



SHORTEST PATH: DIJKSTRA'S ALGORITHM

- TODO:
  - Correctness
  - Implementation and running time

DIJKSTRA'S ALGORITHM: CORRECTNESS

- We want to show that the distance labels are correct
  - Proof by induction
- Base case of $|S| = 1$ is trivial
- Assume true for $|S| = k$
- Let v be the next node the algorithm chooses to add to S, with $e = (u, v)$ the corresponding edge
- Suffices to show that the shortest path is the path to u plus e; consider any other path P from s to v
- Let (x, y) be the first edge on P such that $y \notin S$ and let P' be the subpath to x
  - P is already too long!

$$\ell(P) \geq \ell(P') + \ell_{(x,y)}$$

DIJKSTRA'S ALGORITHM: CORRECTNESS

- We want to show that the distance labels are correct
  - Proof by induction
- Base case of $|S| = 1$ is trivial
- Assume true for $|S| = k$
- Let v be the next node the algorithm chooses to add to S, with $e = (u, v)$ the corresponding edge
- Suffices to show that the shortest path is the path to u plus e; consider any other path P from s to v
- Let (x, y) be the first edge on P such that $y \notin S$ and let P' be the subpath to x
  - P is already too long!

$$\ell(P) \geq \ell(P') + \ell_{(x,y)}$$

Threw out the remaining path from y to v
(non negative weights!)

DIJKSTRA'S ALGORITHM: CORRECTNESS

- We want to show that the distance labels are correct
  - Proof by induction
- Base case of $|S| = 1$ is trivial
- Assume true for $|S| = k$
- Let v be the next node the algorithm chooses to add to S, with $e = (u, v)$ the corresponding edge
- Suffices to show that the shortest path is the path to u plus e; consider any other path P from s to v
- Let (x, y) be the first edge on P such that $y \notin S$ and let P' be the subpath to x
  - P is already too long!

$$\ell(P) \geq \ell(P') + \ell_{(x,y)} \geq d(x) + \ell_{(x,y)}$$

Inductive hypothesis

DIJKSTRA'S ALGORITHM: CORRECTNESS

- We want to show that the distance labels are correct
  - Proof by induction
- Base case of $|S| = 1$ is trivial
- Assume true for $|S| = k$
- Let v be the next node the algorithm chooses to add to S, with $e = (u, v)$ the corresponding edge
- Suffices to show that the shortest path is the path to u plus e; consider any other path P from s to v
- Let (x, y) be the first edge on P such that $y \notin S$ and let P' be the subpath to x
  - P is already too long!

$$\ell(P) \geq \ell(P') + \ell_{(x,y)} \geq d(x) + \ell_{(x,y)} \geq \pi(y)$$

Definition of $\pi(y)$

DIJKSTRA'S ALGORITHM: CORRECTNESS

- We want to show that the distance labels are correct
  - Proof by induction
- Base case of $|S| = 1$ is trivial
- Assume true for $|S| = k$
- Let v be the next node the algorithm chooses to add to S, with $e = (u, v)$ the corresponding edge
- Suffices to show that the shortest path is the path to u plus e; consider any other path P from s to v
- Let (x, y) be the first edge on P such that $y \notin S$ and let P' be the subpath to x
  - P is already too long!

$$\ell(P) \geq \ell(P') + \ell_{(x,y)} \geq d(x) + \ell_{(x,y)} \geq \pi(y) \geq \pi(u)$$

The algorithm picked (the path to) u over (the path to) y to put in S

DIJKSTRA'S ALGORITHM: IMPLEMENTATION

- Bottleneck: $\pi(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$
- First idea: in each iteration, so through all $u \in S$ and all neighbors $v \notin S$, write down the estimated distance and pick the smallest
  - Running time $O(m \cdot n)$, so pretty good already
  - Pretty obvious we can do better
- Data structures!
- We want a data structure that stores estimated distances from nodes in S to nodes not in S
  - We need to be able to insert, remove the smallest and update
- Priority queues!

DIJKSTRA'S ALGORITHM: IMPLEMENTATION

- Put all $v \in V \setminus \{s\}$ in the priority queue with key $\pi(v) = \infty$
- At every iteration we remove the element with the smallest key: time $O(logn)$
- UpdateKey: time $O(logn)$
  - Consider a node $w \notin S$ whose estimate we need to update after adding some node u in S
  - If $(u, w) \notin E$ we don't need to do anything
  - If $(u, w) \in E$ we need to set $\pi(w)$ equal to the minimum of $\pi(w)$ and $d(u) + \ell_{(u,w)}$
    - Find w in the priority queue (remember it's an array)
    - Update key and Heapify-up

DIJKSTRA'S ALGORITHM: RUNNING TIME

- We get a total of n ExtractMin operations and m UpdateKey operations
  - Total running time $O(mlog(n))$
- Running time depends on implementation
- E.g. using a simple array, one can have ExtractMin take $O(n)$ time, but UpdateKey take only $O(1)$ time
  - Total time $O(n^2)$
  - Could be better than $O(mlogn)$ for dense graphs
- Using a Fibonacci heap the running time is $O(nlogn + m)$

SHORTEST PATH WITH NEGATIVE EDGES

- Dijkstra's algorithm crucially uses the fact that edges are not negative
  - As you explore, there is no way that you find a shorter path to someone in S
- With negative edges, for all you know even the distance $d(s) = 0$ is wrong!
  - $e = (s, u)$ with $\ell_e = 1$
  - $e' = (u, s)$ with $\ell_{e'}$=-10
- Is this problem even well defined?

SHORTEST PATH WITH NEGATIVE EDGES

- A puzzle!
- I have an electronic lock that takes an 8-digit password
- But, it's a very bizarre lock
- If you put a sequence it's just going to ignore all the wrong stuff
- Example:
  - Password is 12345678
  - You enter 1234181556798
  - Door opens!
- What's the shortest sequence you can enter to guarantee opening the door?

SHORTEST PATH WITH NEGATIVE EDGES

- Puzzle solution:
  - $\underbrace{0123456789}_{\text{8 times}}0123456789\ldots$

SHORTEST PATH WITH NEGATIVE EDGES

- Puzzle solution:
  - $\underbrace{0123456789}_{\text{8 times}}0123456789\ldots$
- Sufficient:
  - We'll get the first digit in the first 10, the second digit in the second 10, and so on
- Necessary:
  - My password could be 00000000, so you better have eight zeros
  - My password could be 11111111, so you better have eight ones
  - And so on. Therefore, you need at least 80 digits

SHORTEST PATH WITH NEGATIVE EDGES

- Different view of Dijkstra: the algorithm updates along edges
- Update($e = (u, v)$):
  - $\pi(v) = \min\{\, \pi(v), \pi(u) + \ell_e \,\}$
- Important properties:
  - At all times, $\pi(v)$ is an overestimate of the truth
  - If the true shortest path uses (u, v) as the last edge, and you have a correct value for $\pi(u)$, then you have computed a correct value for $\pi(v)$

SHORTEST PATH WITH NEGATIVE EDGES

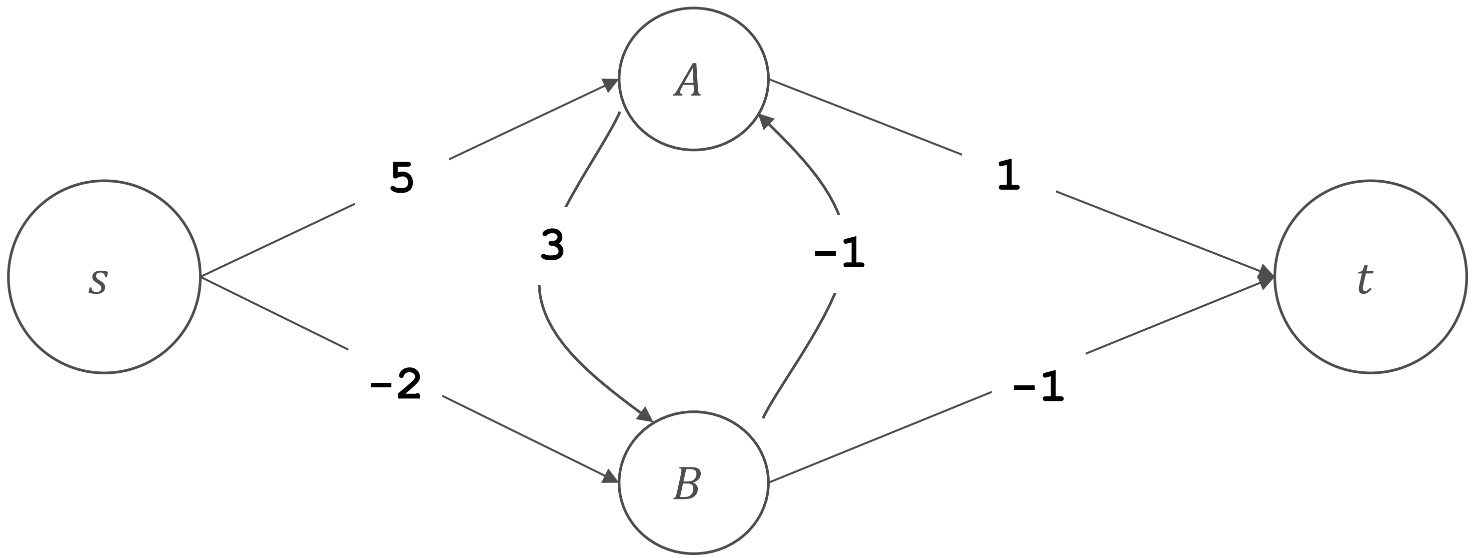
- Say that the true shortest path from s to t is
$$s \rightarrow u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_k \rightarrow t$$
- Look at the edges updated by the algorithm:
  - $e_1, e_2, e_3, \ldots$
- If this sequence includes $(s, u_1), (u_1, u_2), \ldots,$ (u_k, t) in *this order* (but not necessarily consecutively) we have computed the distance from s to t correctly!
- How do we know which edges to update??
- Same answer as the puzzle
  - All of them, $n - 1$ times (the total number of edges in the shortest path is at most $n - 1$)

BELLMAN FORD ALGORITHM

- For all $v \in V$: $\pi(v) = \infty$
- Repeat $n - 1$ times:
  - For all $e = (u, v) \in E$:
    - Update(e): $\pi(v) = \min\{\pi(v), \pi(u) + \ell_e\}$

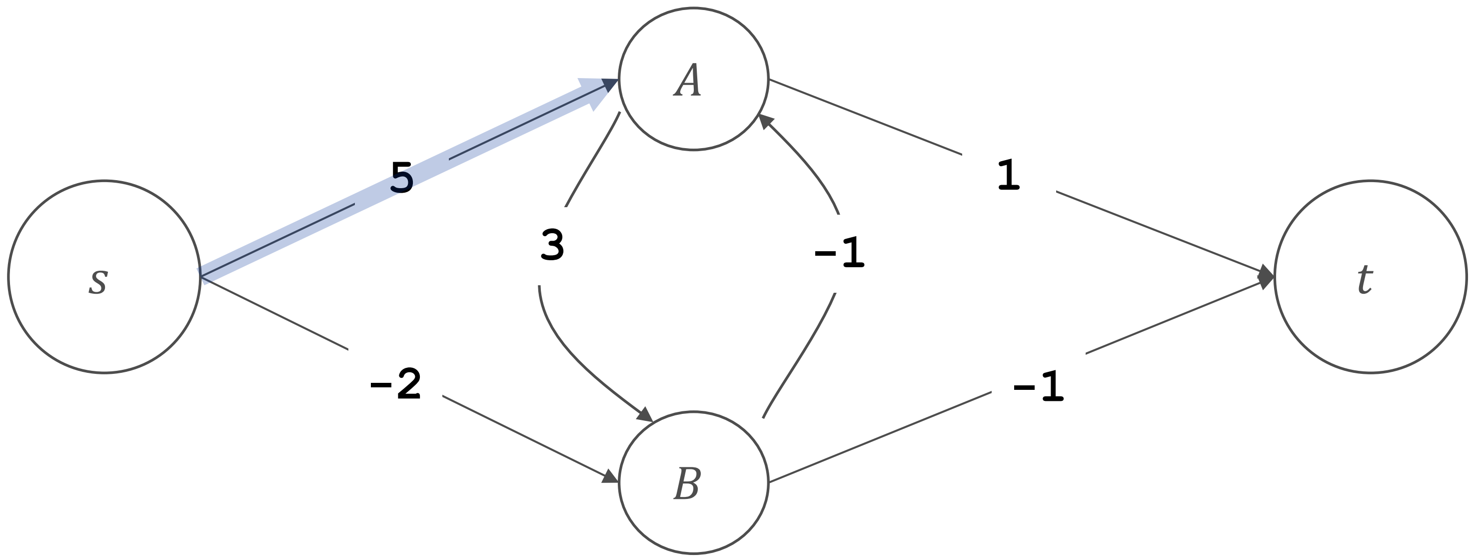
- (Almost…)

BELLMAN FORD ALGORITHM



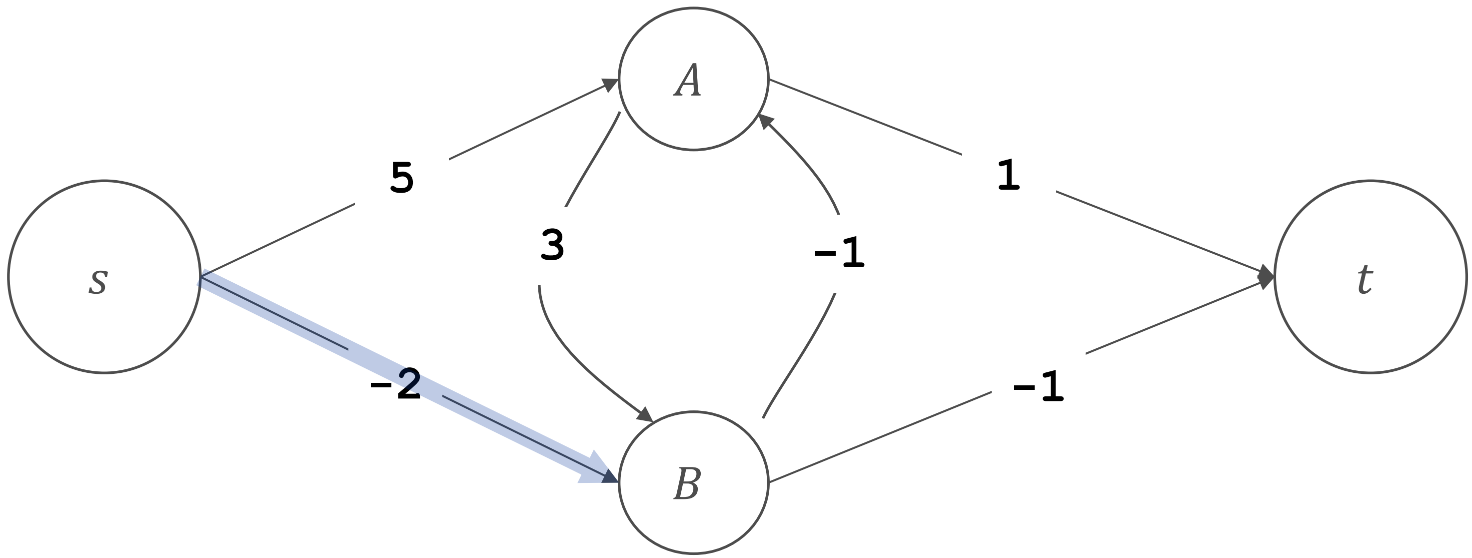
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = \infty$
- $\pi(B) = \infty$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



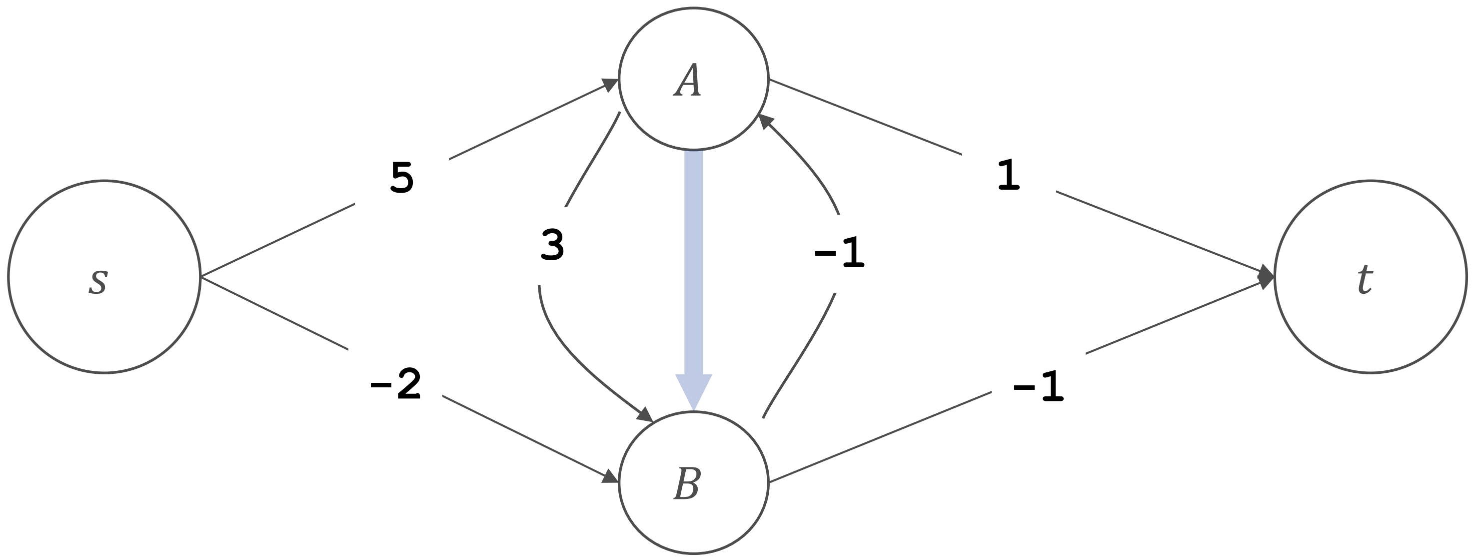
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = \infty$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



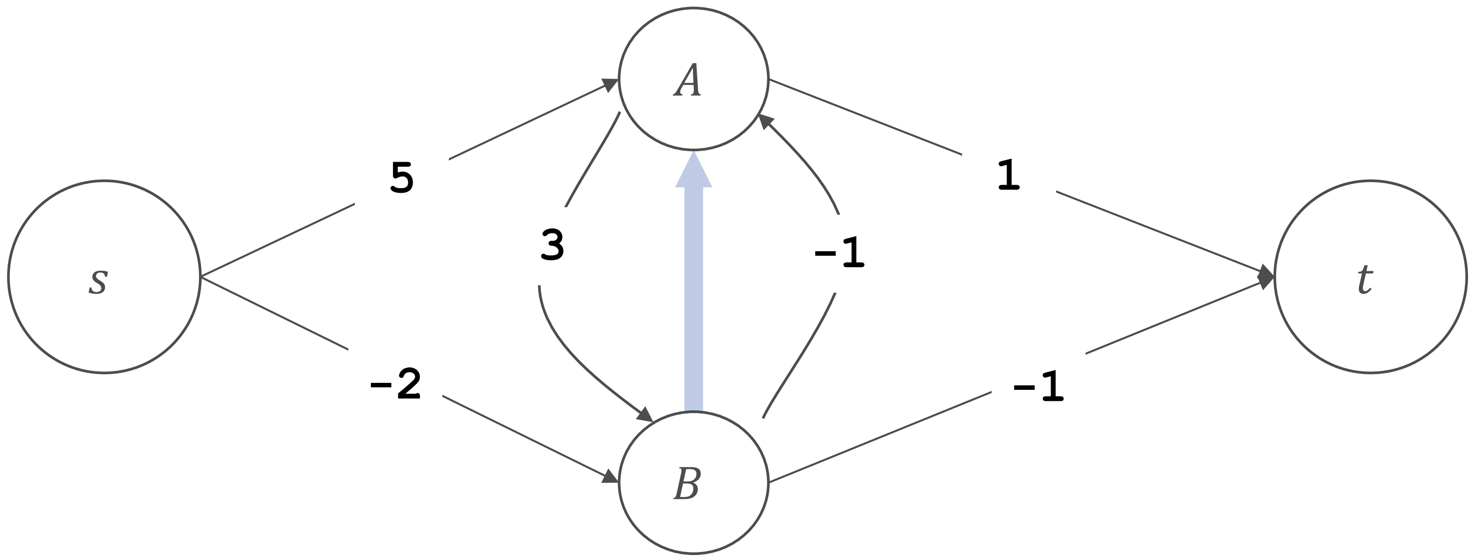
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



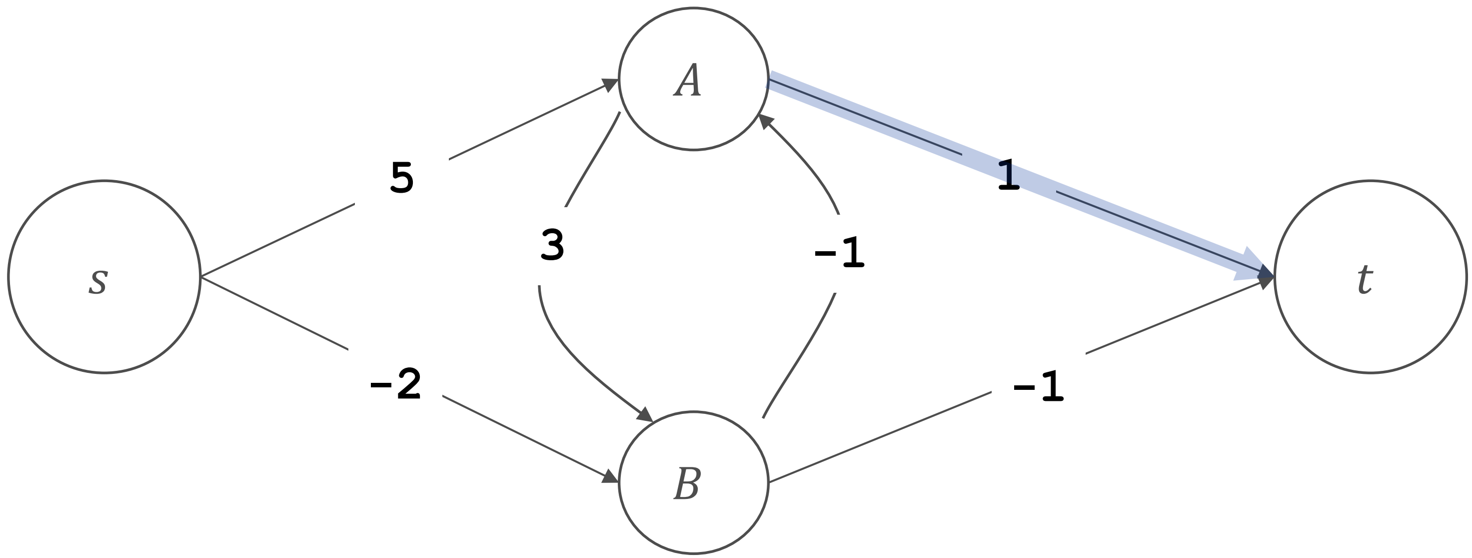
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



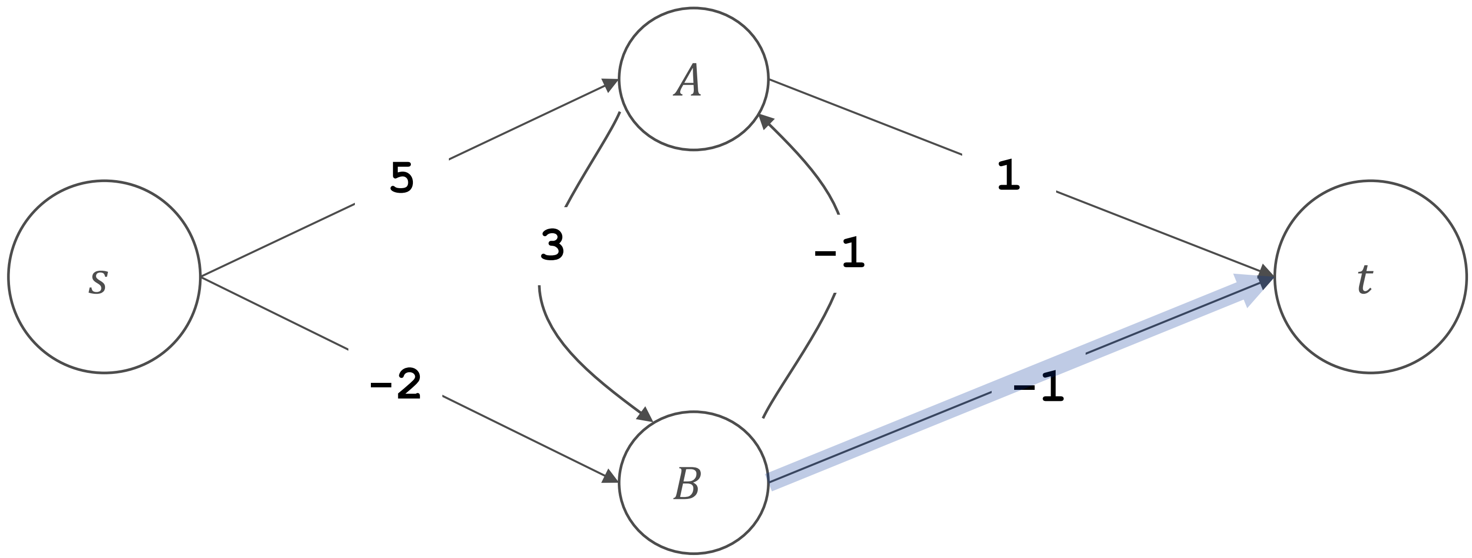
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



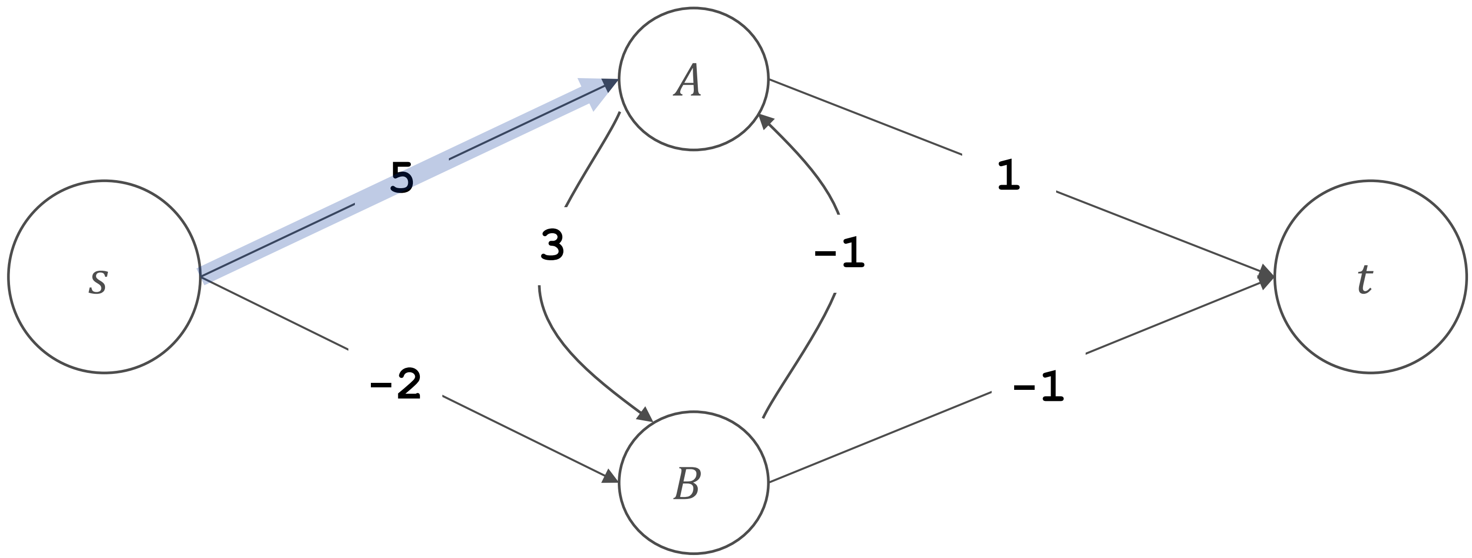
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -2$

BELLMAN FORD ALGORITHM



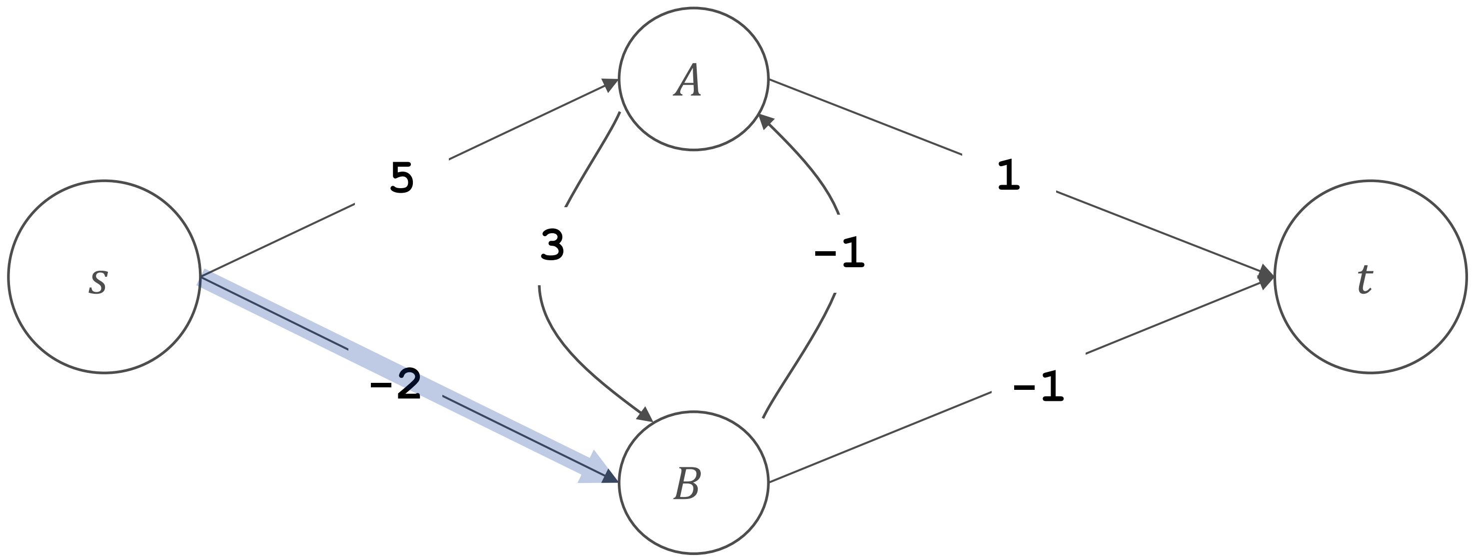
- Order of edges: $(s,A),(s,B),(A,B),(B,A),(A,t),(B,t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



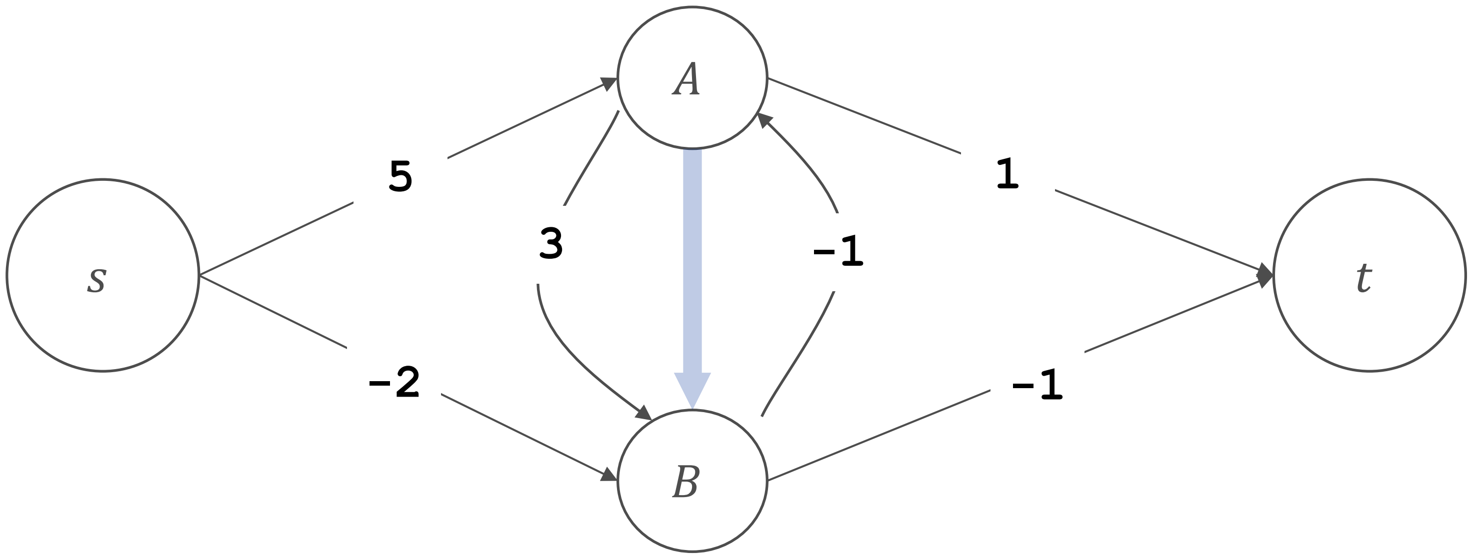
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



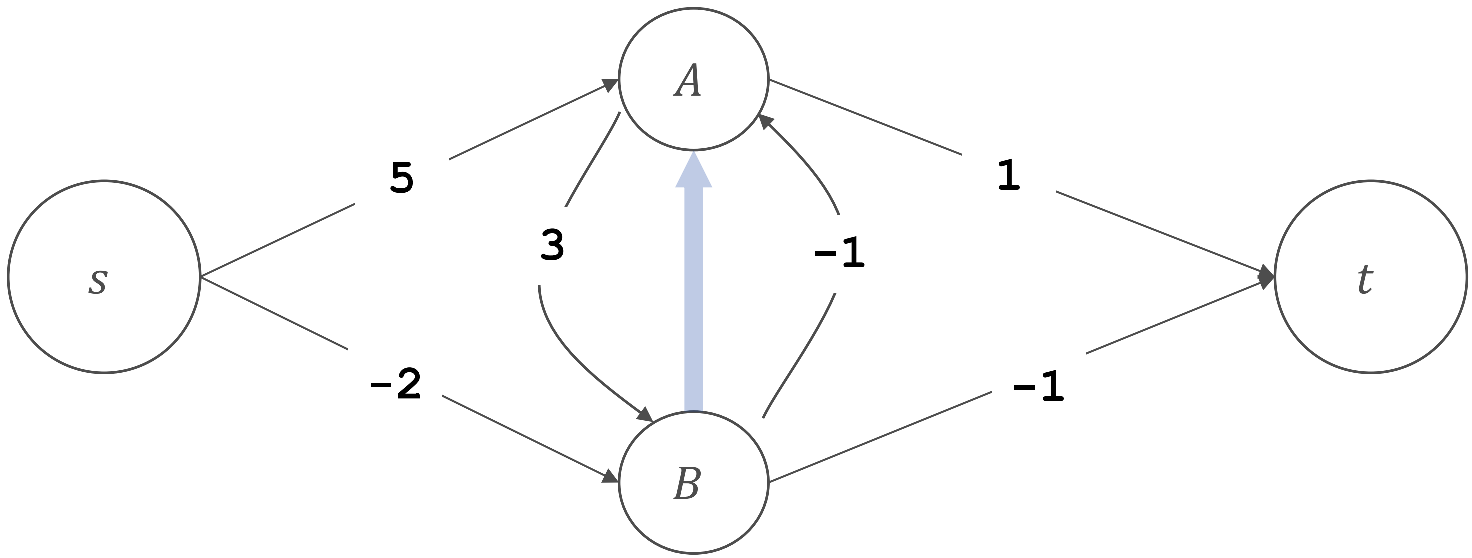
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



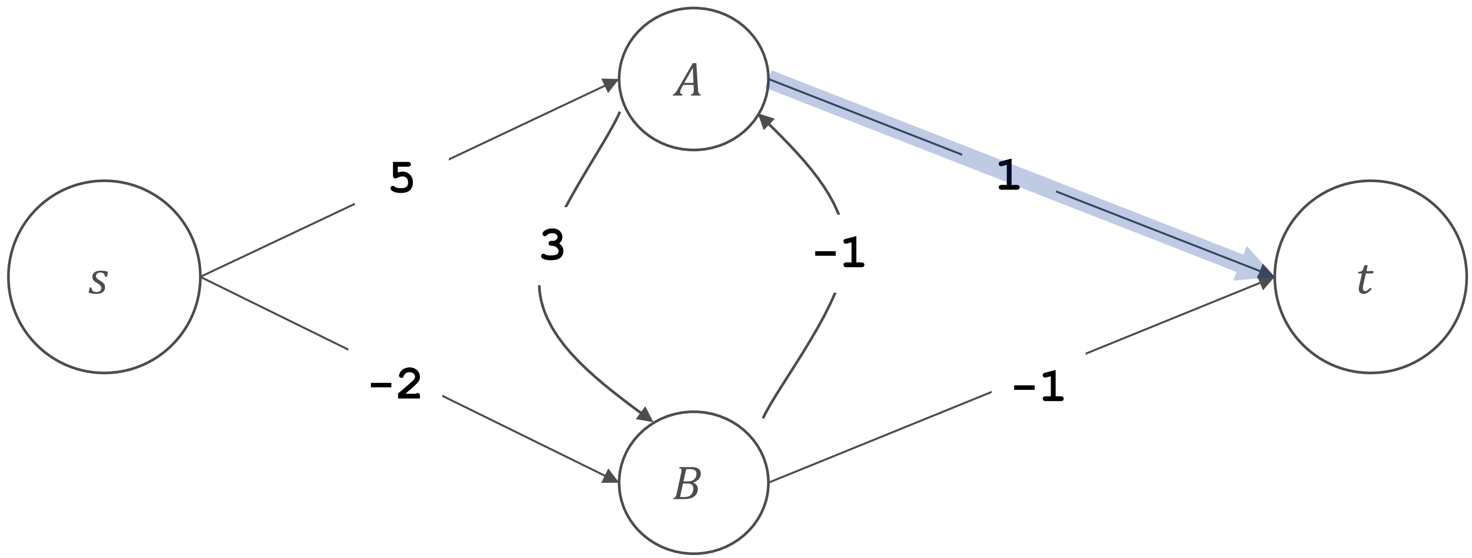
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



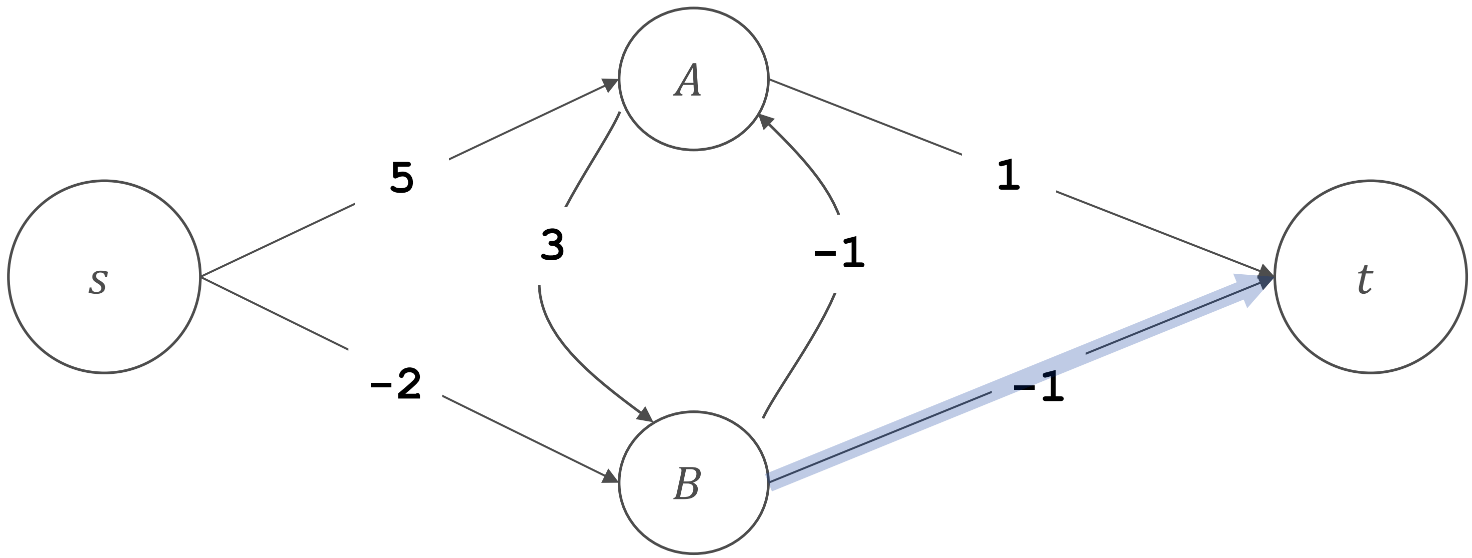
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



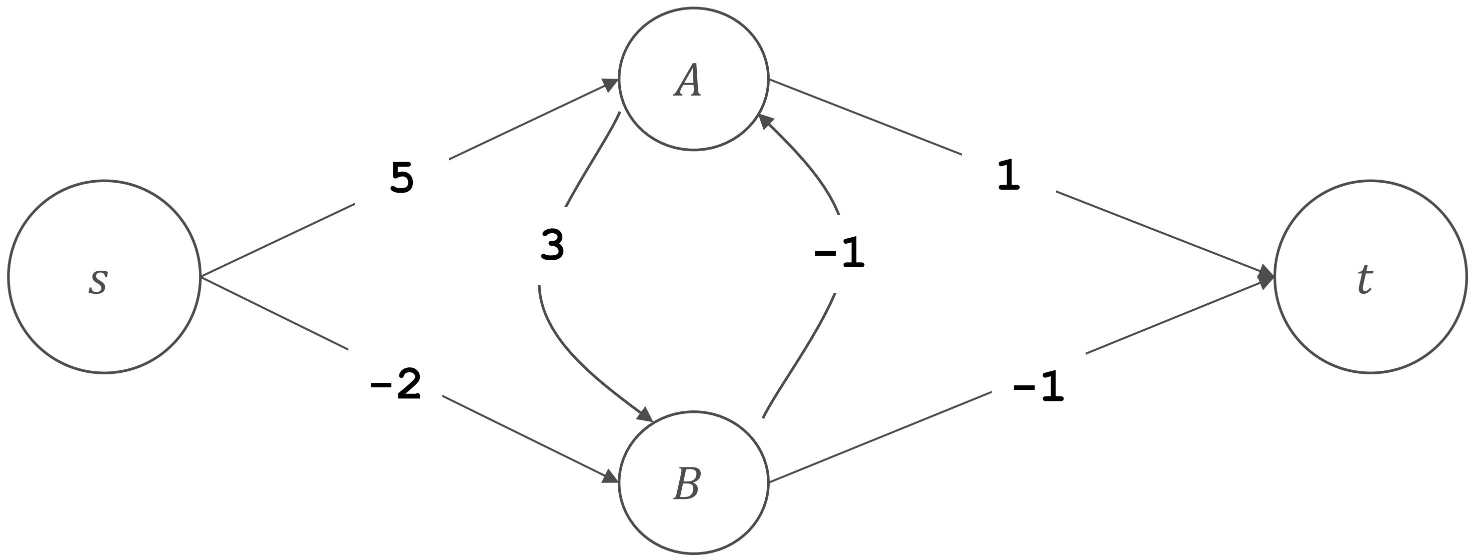
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

BELLMAN FORD ALGORITHM



- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3$
- $\pi(B) = -2$
- $\pi(t) = -3$

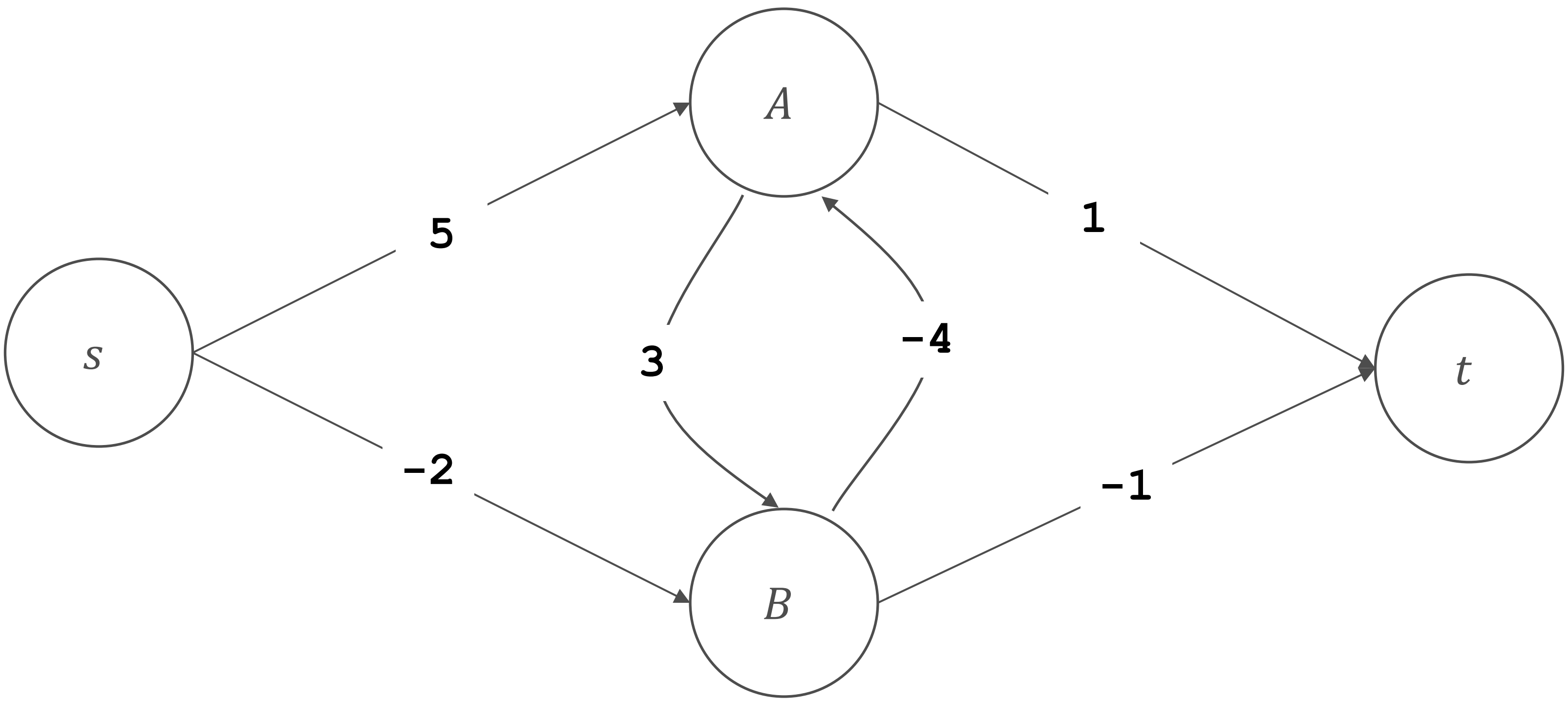
BELLMAN FORD ALGORITHM



- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -3, \pi(B) = -2, \pi(t) = -3$
- Since nothing changed in the last iteration there's no point in repeating

BELLMAN FORD ALGORITHM

- What about the “well defined” business?



- If there’s a negative cycle there is no shortest path!

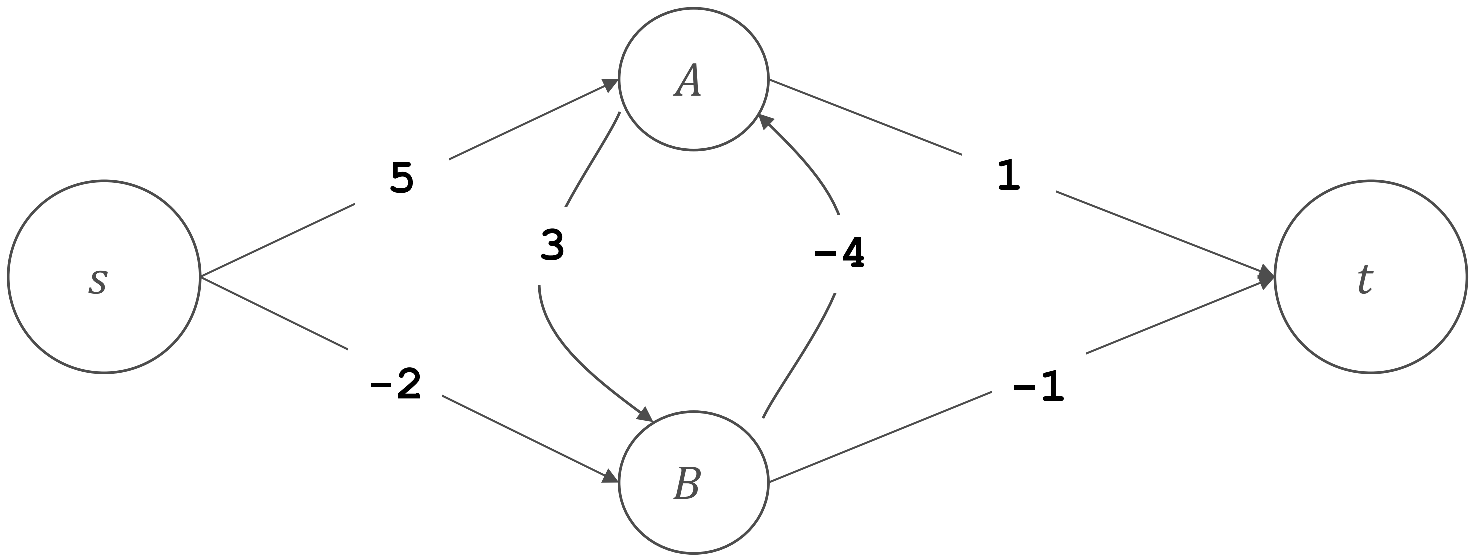
BELLMAN FORD ALGORITHM

- If there are no negative cycles, is there a shortest path?
- It will be implied by the correctness of Bellman-Ford
- But first, how do we find negative cycles?
- Trick:
  - If there's a negative cycle $u_1 \to \cdots \to u_k$ then update(e) should make $\pi(u_i)$ smaller and smaller if we keep repeating the loop
  - Do the loop one more time!

BELLMAN FORD ALGORITHM

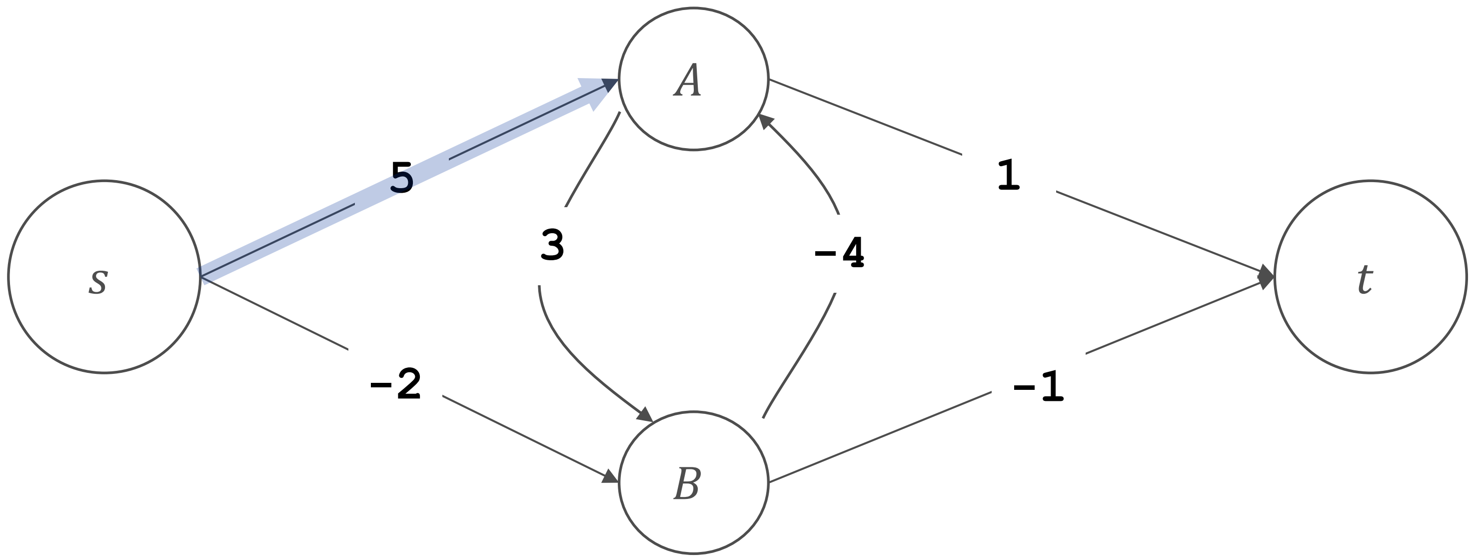
- For all $v \in V: \pi(v) = \infty$
- Repeat $n - 1$ times:
  - For all $e = (u, v) \in E$:
    - Update(e): $\pi(v) = \min\{\pi(v), \pi(u) + \ell_e\}$
- //Check for negative cycles
- For all $e = (u, v) \in E$:
  - If $\pi(v) > \pi(u) + \ell_e$:
    - Exit; negative cycle detected

BELLMAN FORD ALGORITHM



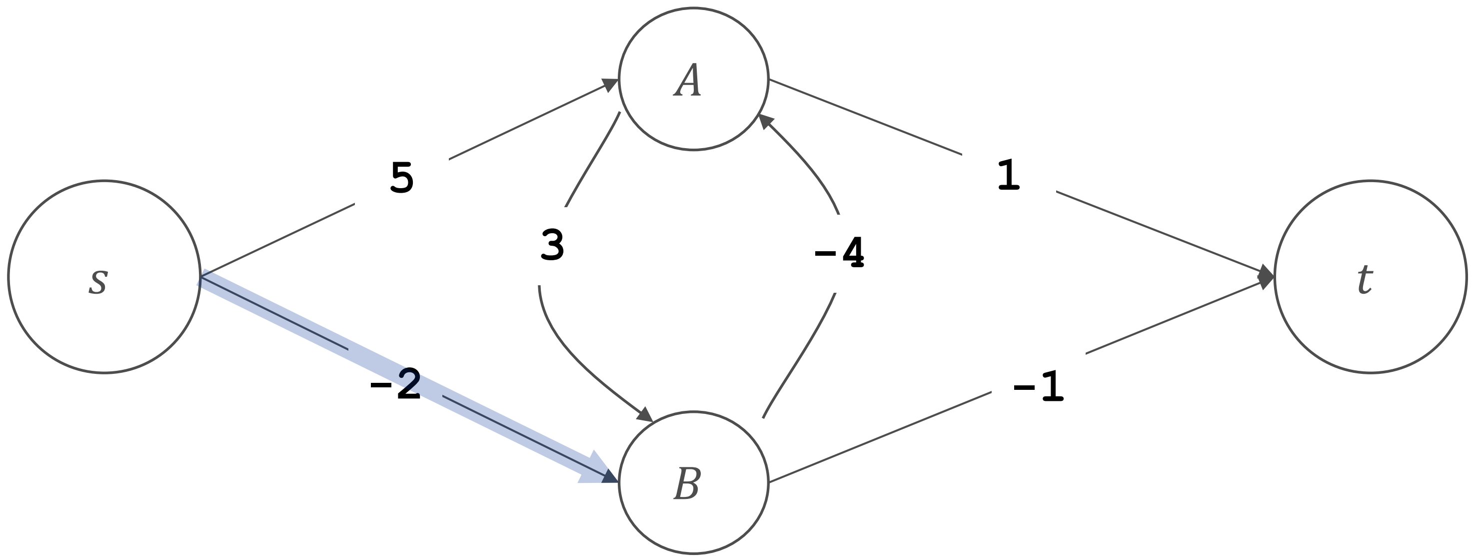
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = \infty$
- $\pi(B) = \infty$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



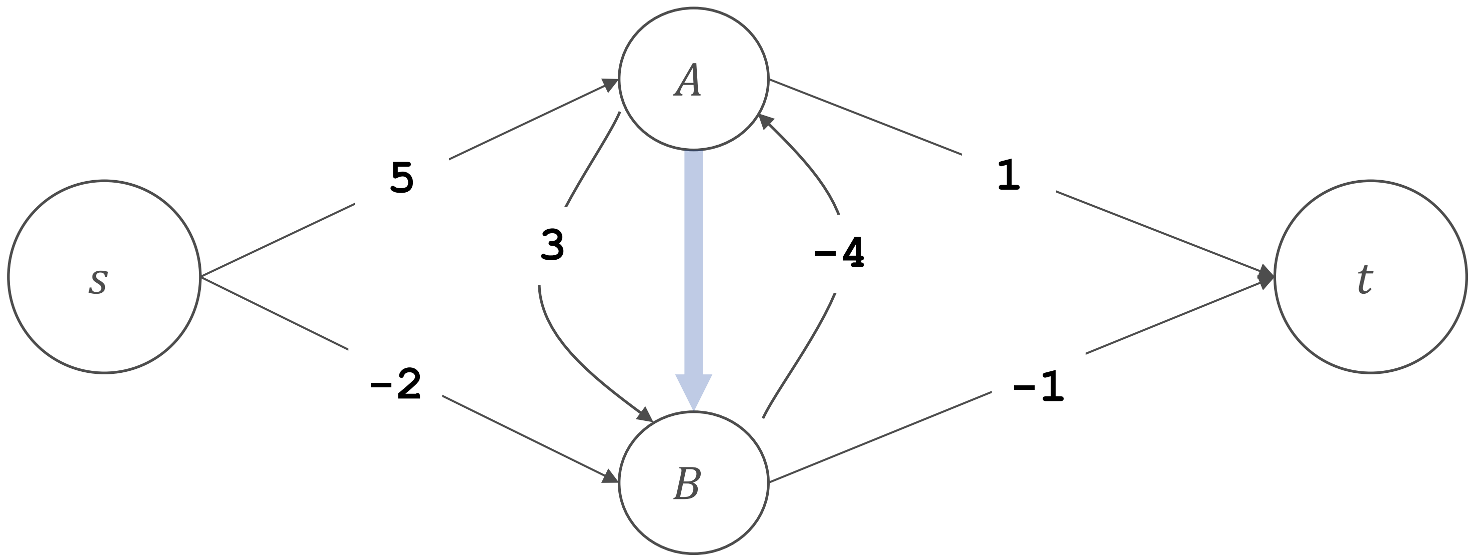
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = \infty$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



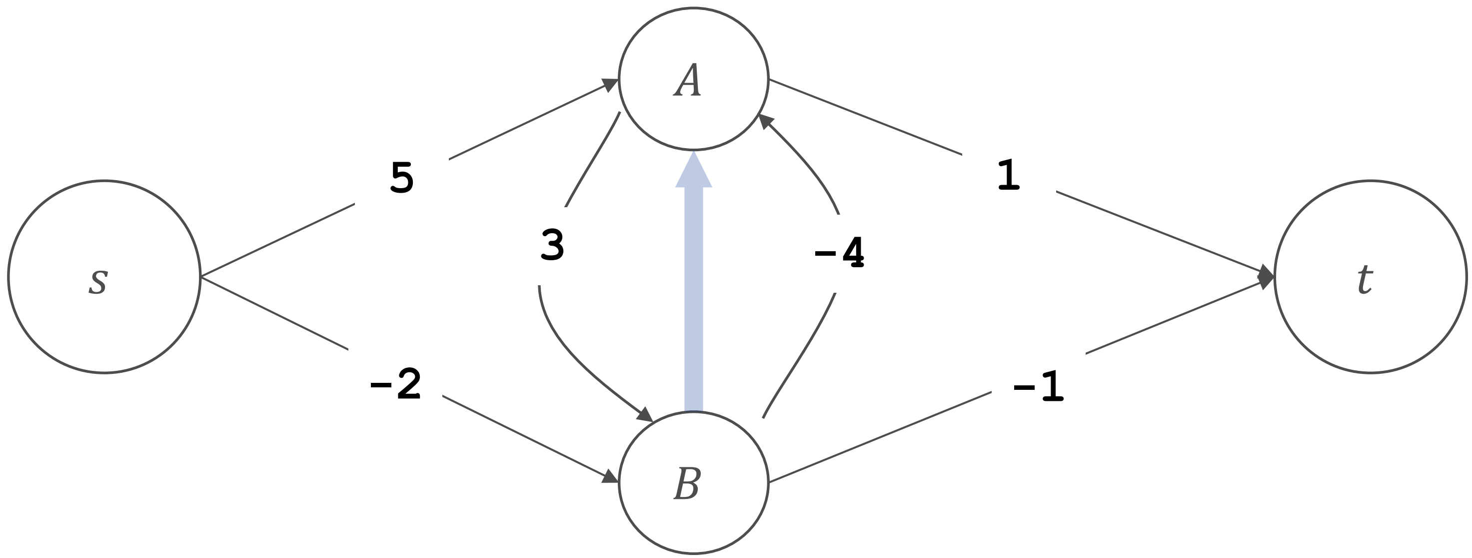
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



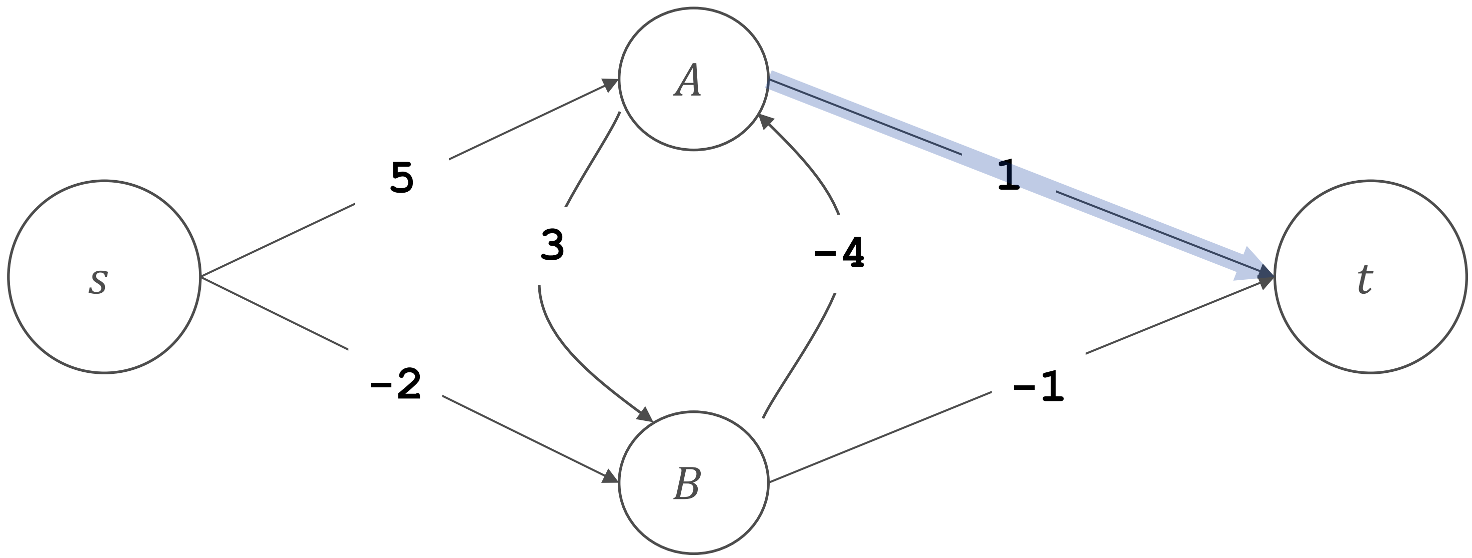
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = 5$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



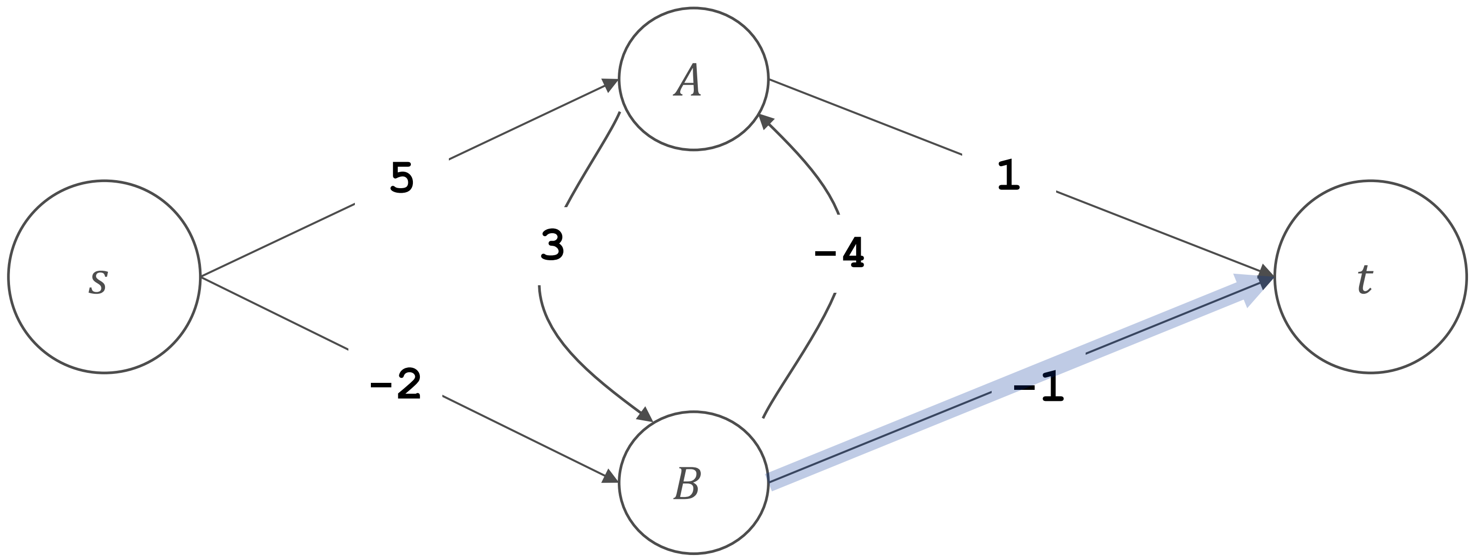
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -2$
- $\pi(t) = \infty$

BELLMAN FORD ALGORITHM



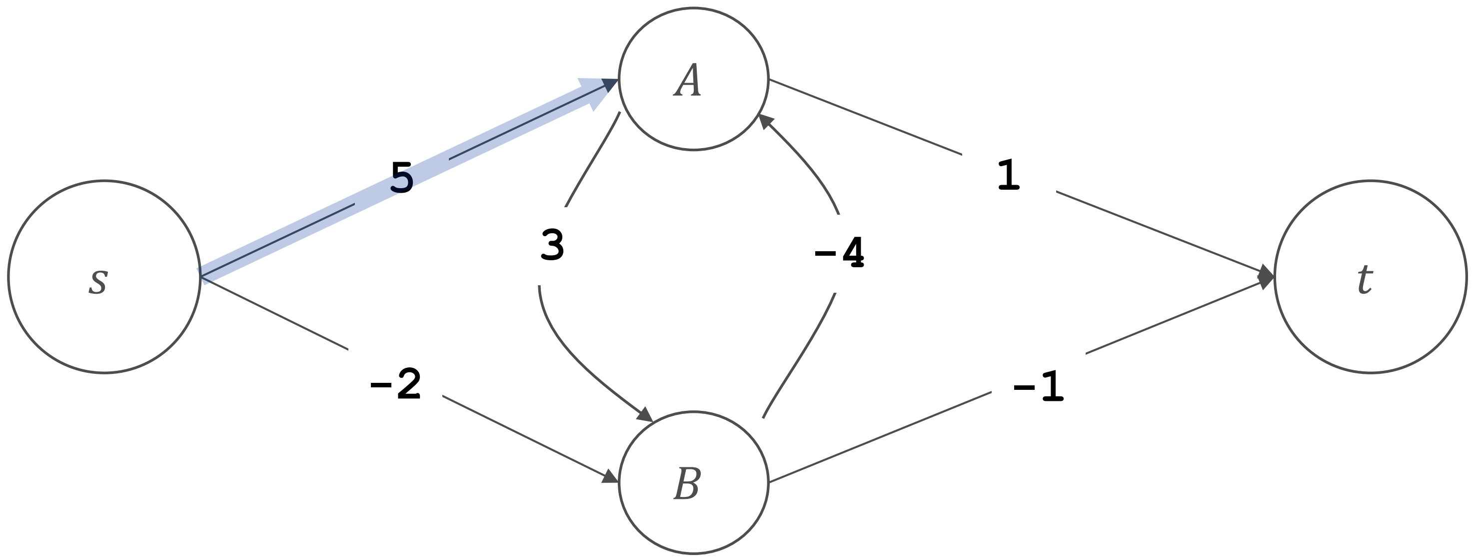
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -2$
- $\pi(t) = -5$

BELLMAN FORD ALGORITHM



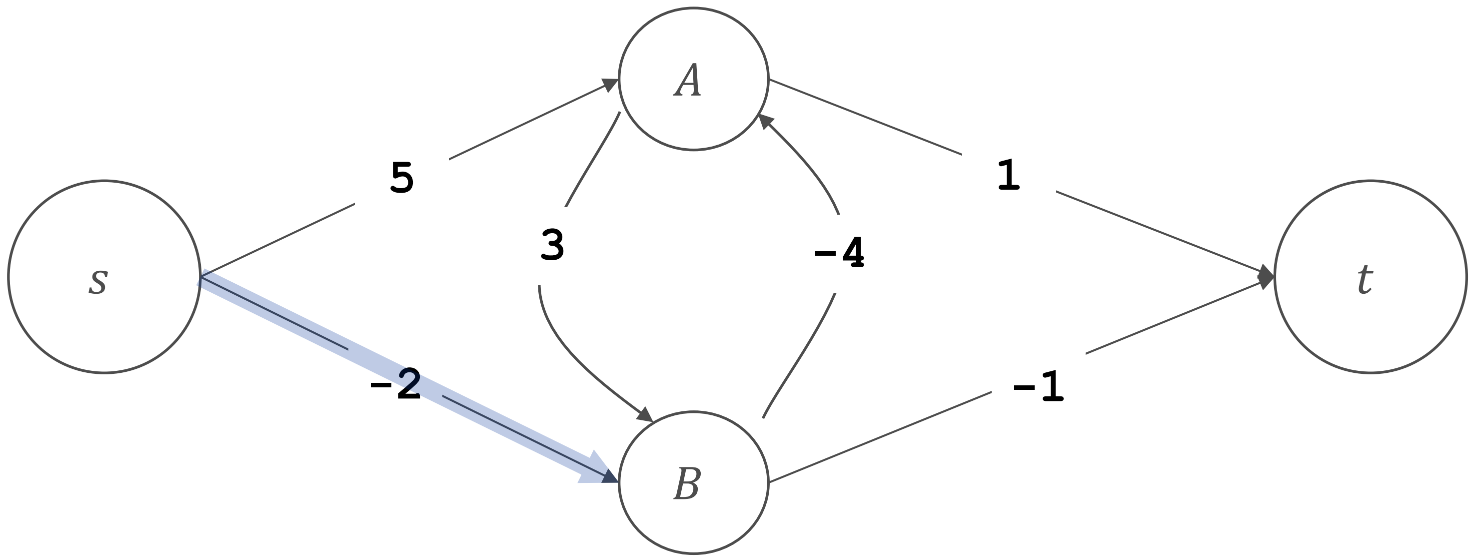
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -2$
- $\pi(t) = -6$

BELLMAN FORD ALGORITHM



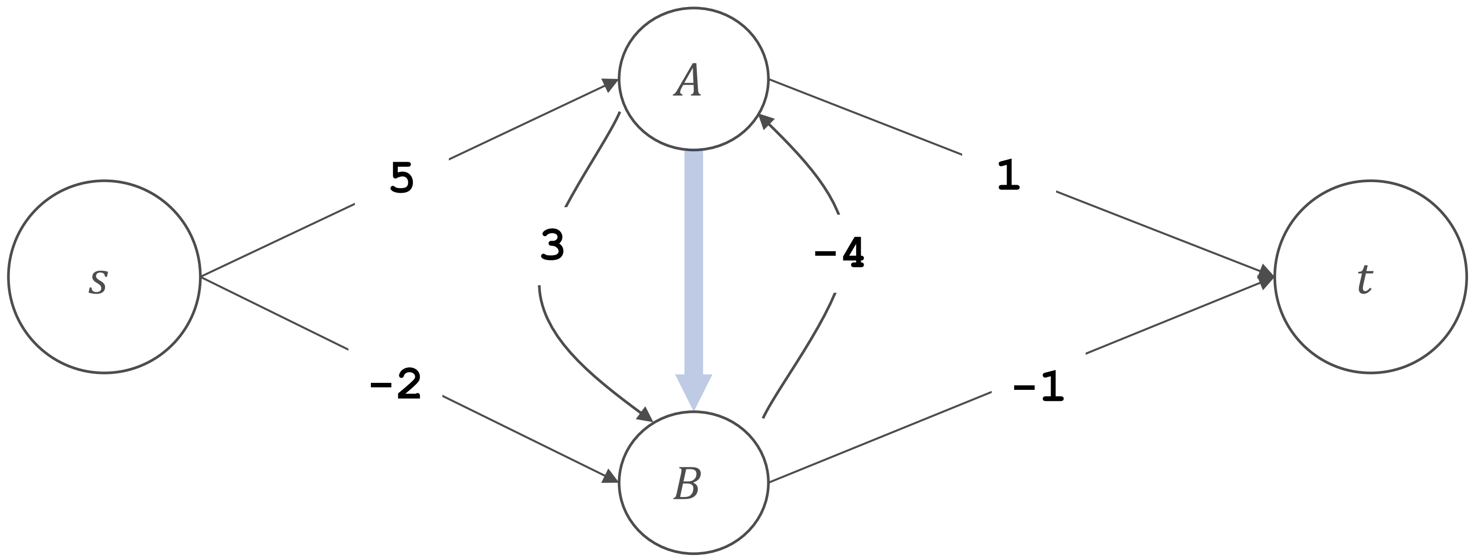
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -2$
- $\pi(t) = -6$

BELLMAN FORD ALGORITHM



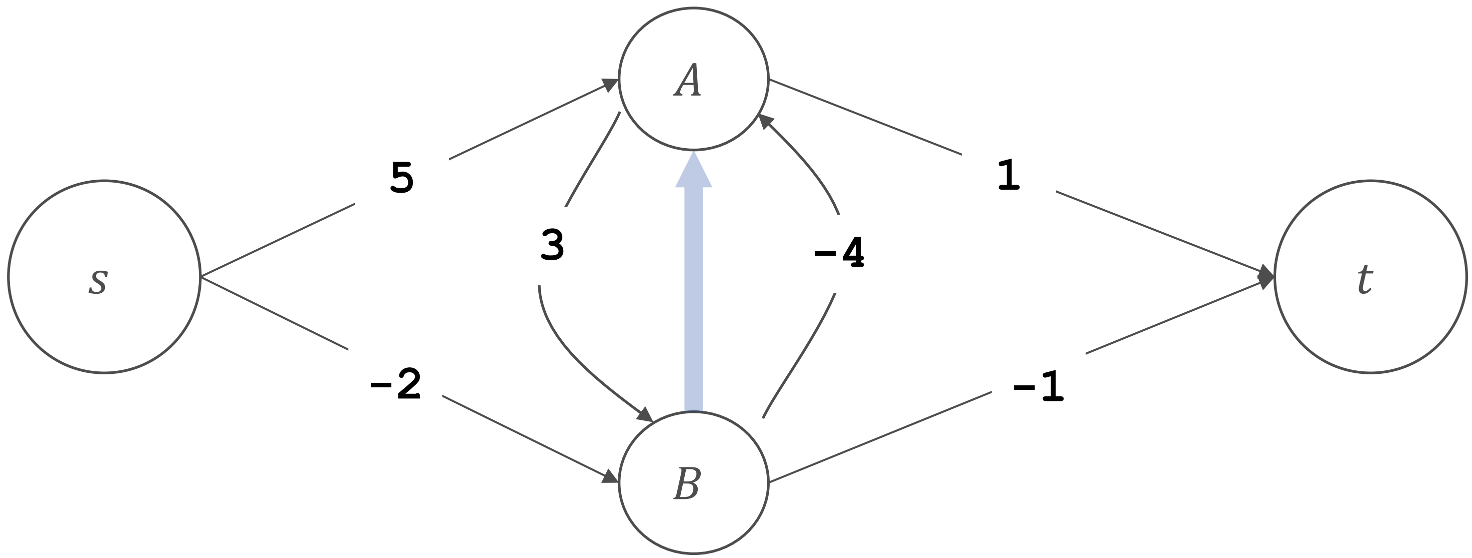
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -2$
- $\pi(t) = -6$

BELLMAN FORD ALGORITHM



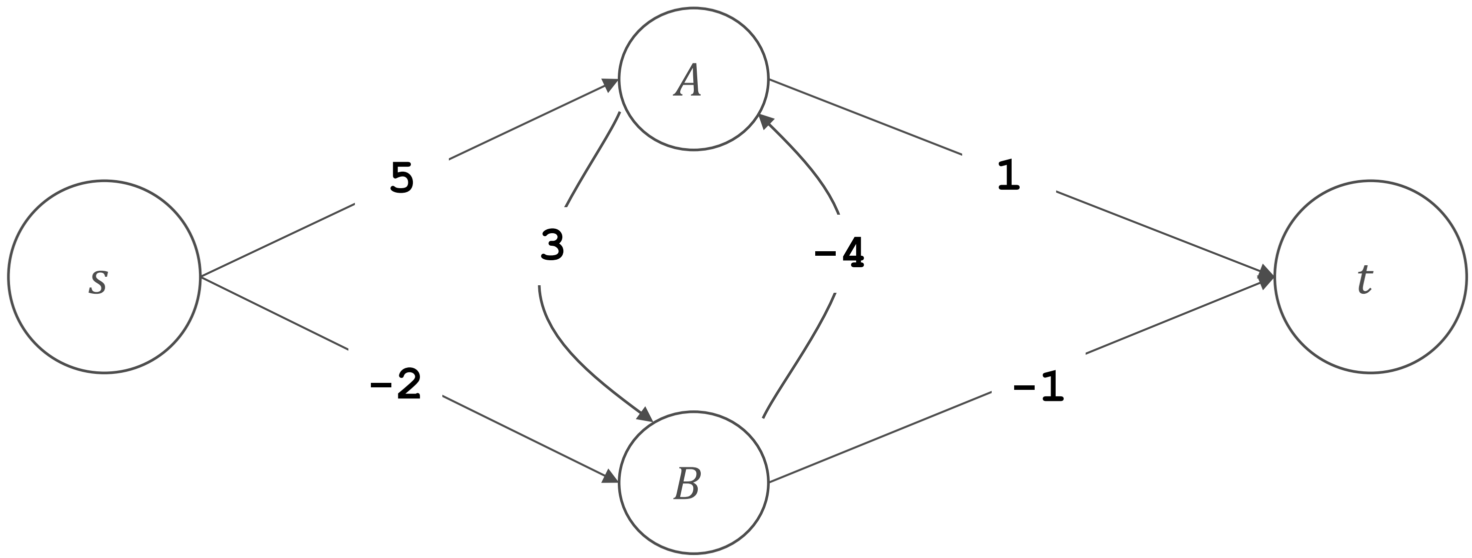
- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -6$
- $\pi(B) = -3$
- $\pi(t) = -6$

BELLMAN FORD ALGORITHM



- Order of edges: $(s, A), (s, B), (A, B), (B, A), (A, t), (B, t)$
- $\pi(A) = -7$
- $\pi(B) = -3$
- $\pi(t) = -6$

BELLMAN FORD ALGORITHM



- $\pi(A), \pi(B)$ and $\pi(t)$ will keep getting smaller and smaller until we stop

BELLMAN FORD ALGORITHM: CORRECTNESS

- Lemma: If there is a negative cycle we will find it
- <u>Proof</u>
- Suppose $v_0 \to v_1 \to \cdots \to v_k$ is a negative cycle, where $v_0 = v_k$
- Suppose that $\pi(v_i) \leq \pi(v_{i-1}) + \ell_{(v_{i-1},v_i)}$ for all i at the last pass of the algorithm
- Summing up over all i we have

$$\sum_{i=1}^{k} \pi(v_i) \leq \sum_{i=1}^{k} \pi(v_{i-1}) + \sum_{i=1}^{k} \ell_{(v_{i-1},v_i)}$$

BELLMAN FORD ALGORITHM: CORRECTNESS

$$\sum_{i=1}^{k} \pi(v_i) \leq \sum_{i=1}^{k} \pi(v_{i-1}) + \sum_{i=1}^{k} \ell_{(v_{i-1},v_i)}$$

- $\pi(v_0) = \pi(v_k)$, since $v_0 = v_k$. Therefore

$$\sum_{i=1}^{k} \pi(v_{i-1}) = \pi(v_0) + \sum_{i=1}^{k-1} \pi(v_i) = \sum_{i=1}^{k} \pi(v_i)$$

- Plugging in above we get

$$\sum_{i=1}^{k} \ell_{(v_{i-1},v_i)} \geq 0$$

- I.e. the cycle is not negative. Contradiction

BELLMAN FORD ALGORITHM: CORRECTNESS

- Lemma: If the graph has no negative cycles, then the shortest paths are computed correctly
- We will show that $\pi_k(v)$, the estimate of the distance at the k-th iteration, is the weight of the minimum weight path from s to v that uses $\leq k$ edges
- Then, for $k = n - 1$, this would imply correctness since $n - 1$ is an upper bound on the length of the shortest path
  - If a vertex is repeated, there must be a cycle
  - The cycle can't be negative by assumption, so removing it makes the path shorter

BELLMAN FORD ALGORITHM: CORRECTNESS

- $k = 0$. Trivially true: no length from s to itself
- Suppose true for $k - 1$
- Let v be some node and P be the shortest path with length k. Let u be the vertex right before v on P
  - $s \to v_1 \to \cdots \to u \to v$
- Following P to go from s to u would give the shortest path to u on $k - 1$ edges
  - Otherwise, we could come up with a shorter path to v
- By the inductive hypothesis, $\pi_{k-1}(u)$ is correct
- At iteration k, at some point we encounter the edge (u, v)
  - We set $\pi_k(v) = \min\{\pi_{k-1}(v),\ \pi_{k-1}(u) + \ell_{(u,v)}\}$
  - Therefore $\pi_k(v)$ is at most the weight of P (which is equal to $\pi_{k-1}(u) + \ell_{(u,v)}$)

SUMMARY

- Shortest path algorithms
  - Non-negative weights: Dijkstra
  - Negative weights: Bellman-Ford