Algorithms and Data Structures

Module 2

Lecture 8

Shortest paths, part 1

Adigeev Mikhail Georgievich

mgadigeev@sfedu.ru

Shortest paths problem

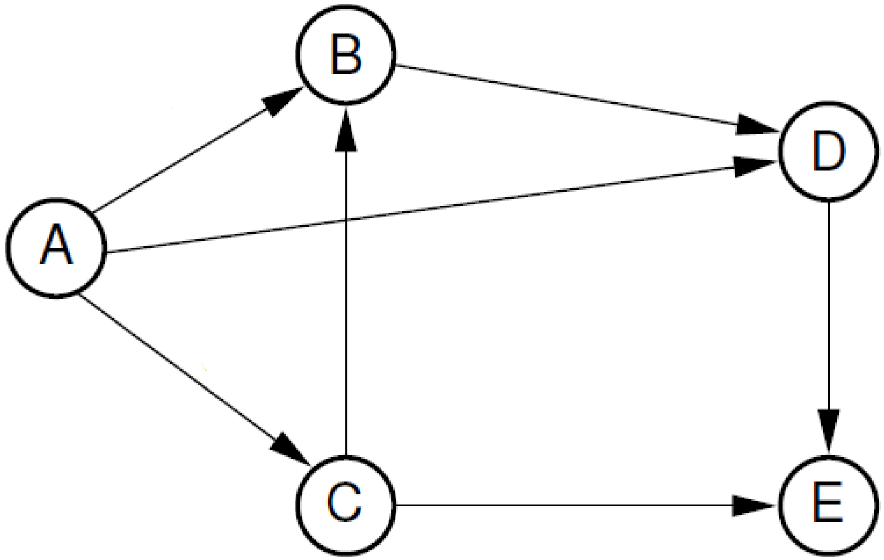
By this time, we have studied two algorithms for calculating distances and building shortest paths on graphs: a BFS-based algorithm (lecture 3) and a topological sort based algorithm for DAGs (lecture 5). Why they are not enough?

BFS: applications (lecture 3)

Graph G=(V,E).

A *distance* between vertices *u* and *v* is the minimum length of the path between *u* and *v*.

dist(A,E) = 2

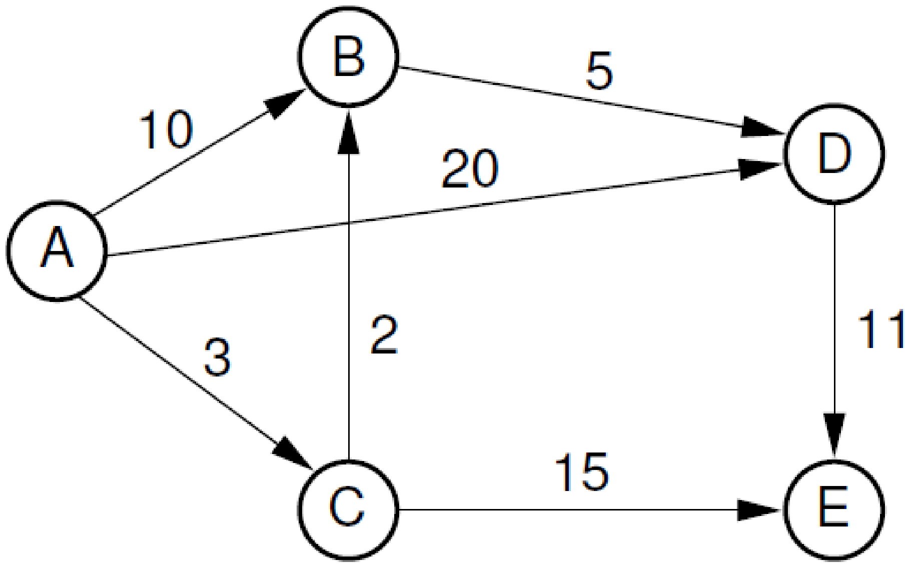


BFS: applications (lecture 3)

Weighted graph G=(V,E), $w: E \to R$

A *distance* between vertices *u* and *v* is the minimum weight (=sum of edges' weights) of the path between *u* and *v*.

dist(A,E) = 18

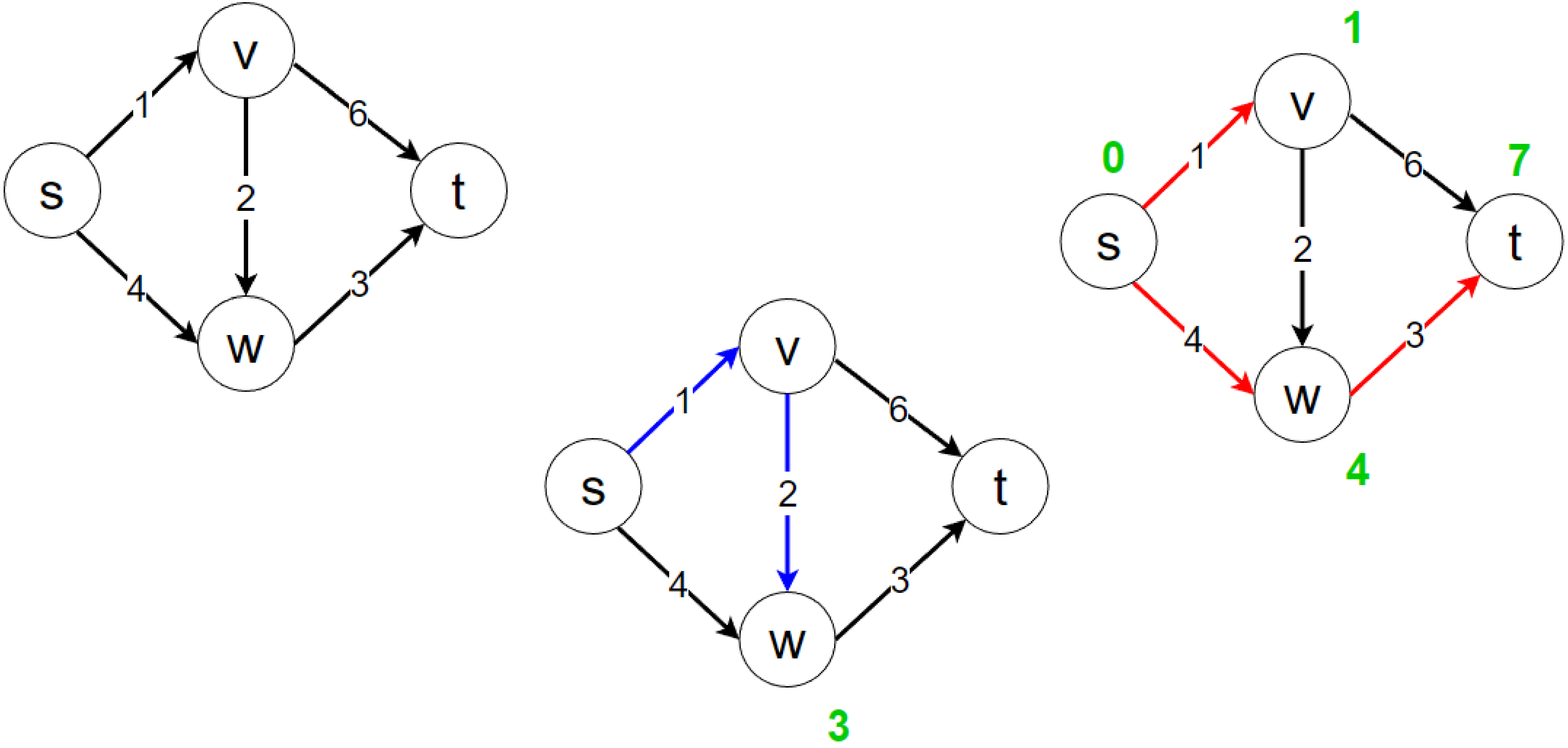


BFS: applications (lecture 3)

For unweighted graphs distances from $s \in V$ to all other vertices can be calculated using BFS.

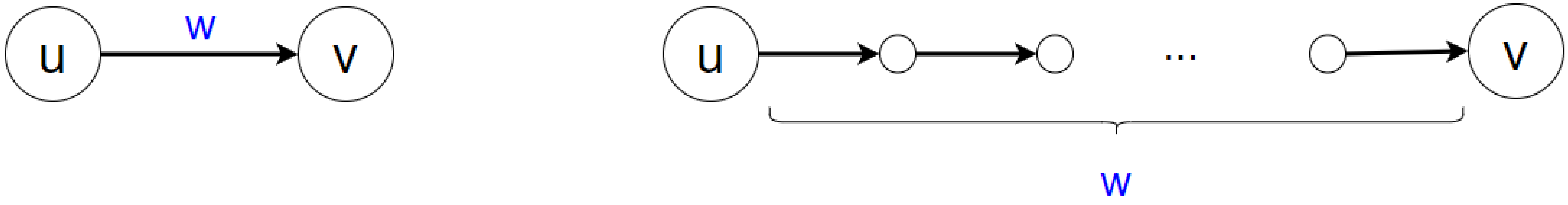
But it works incorrectly for weighted graphs ☹

BFS: applications (lecture 3)



BFS: applications (lecture 3)

We can fix the problem by replacing an edge of weight w with a path of length w.



This algorithm is correct but very inefficient for big weights, since its time complexity is $O(n' + m') = O(\sum w_i)$

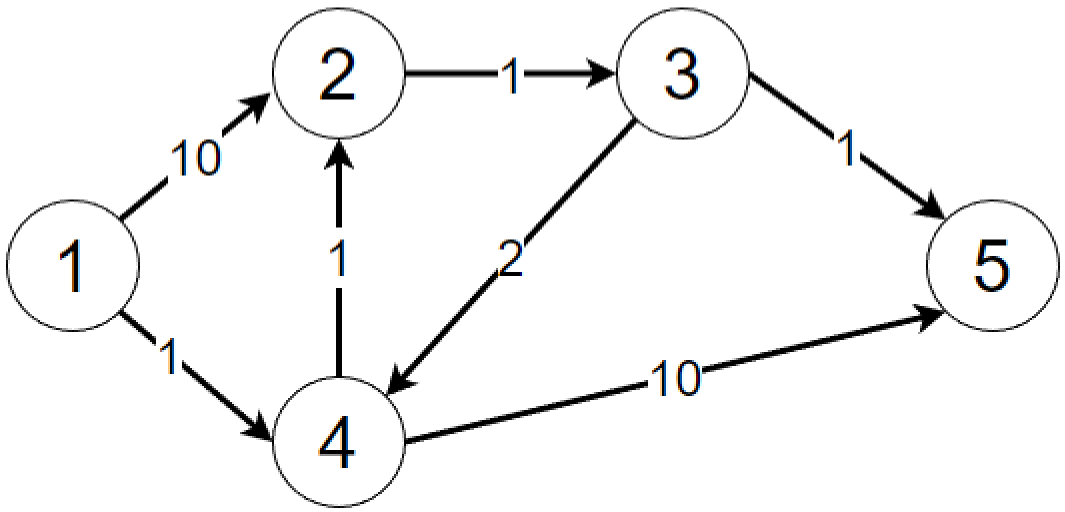
Topological sort based (lecture 5)

The algorithm based on topological sort is correct and efficient for DAGs but is incorrect for graphs with cycles ☹

Let us consider a graph with a directed cycle.

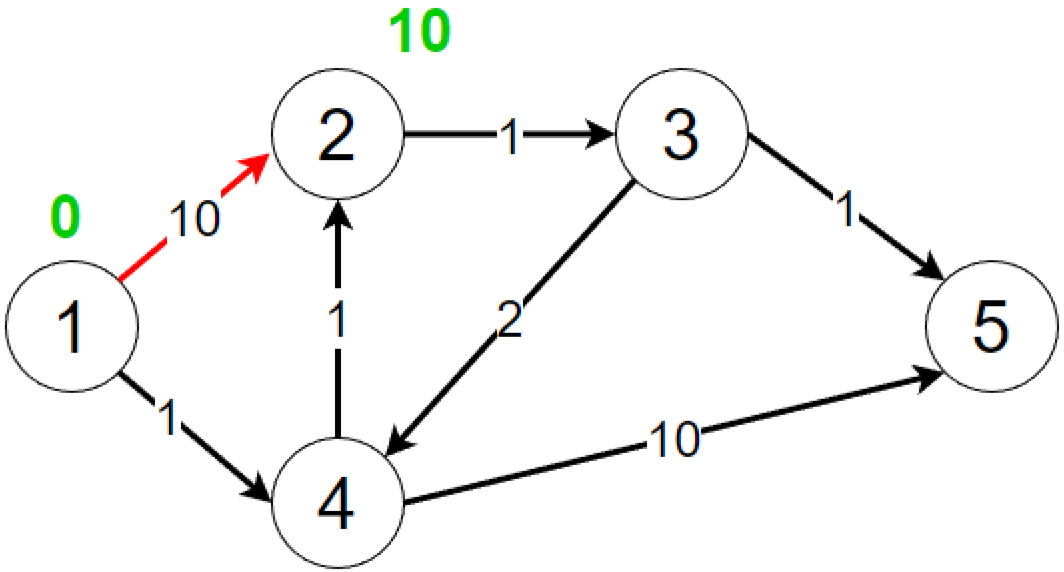
Topological sort based (lecture 5)

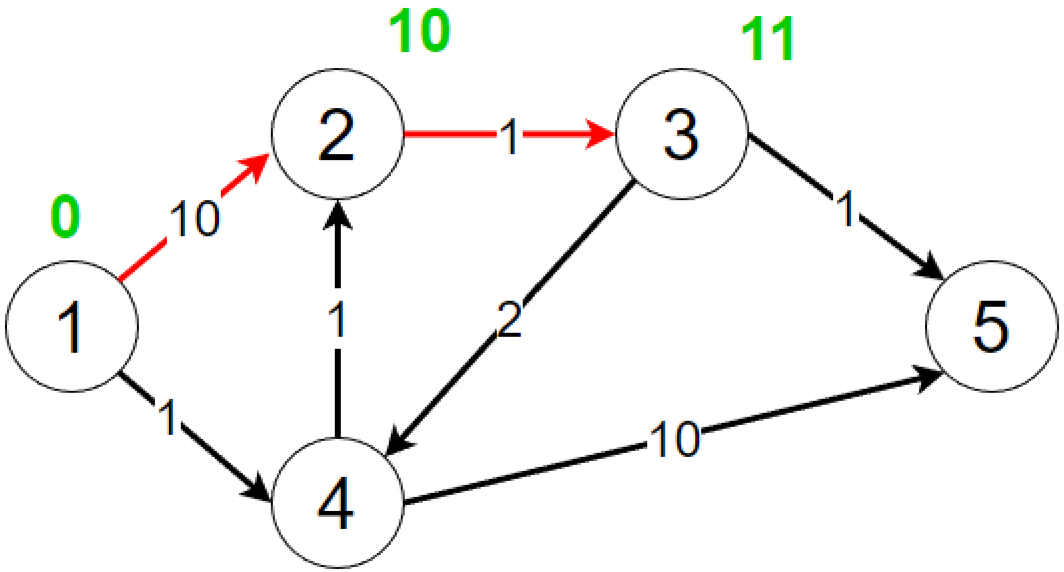
Let us consider a graph with a directed cycle.



Topological sort based (lecture 5)

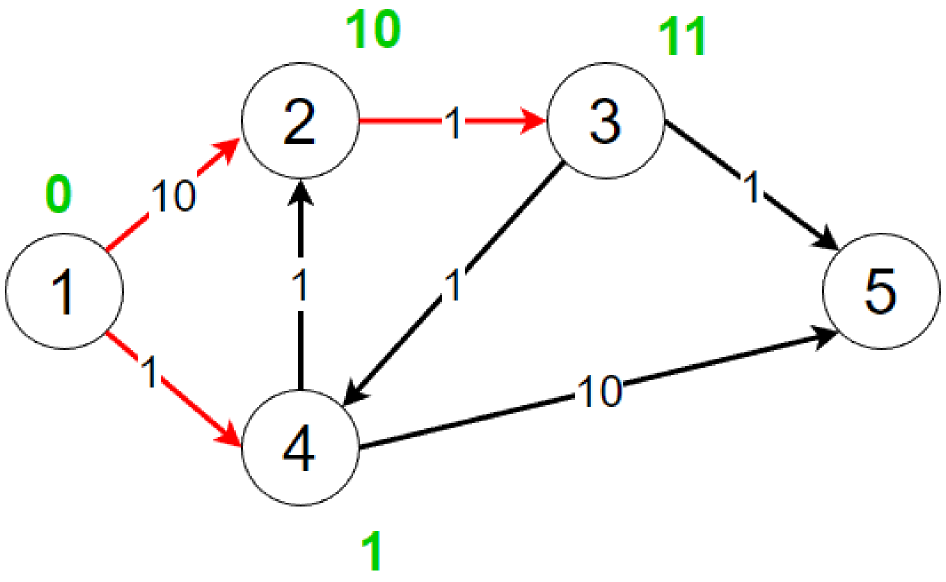
The topological sort based algorithm works as follows

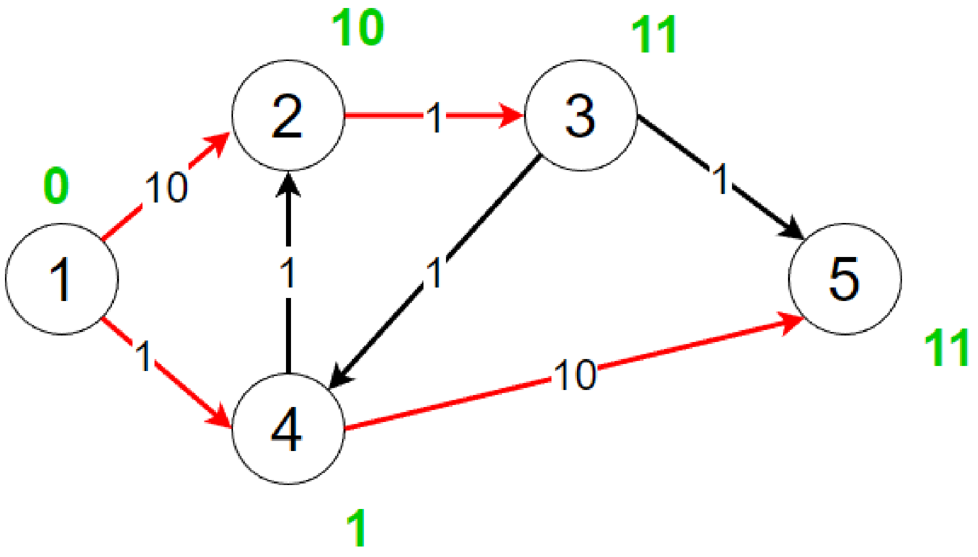




Topological sort based (lecture 5)

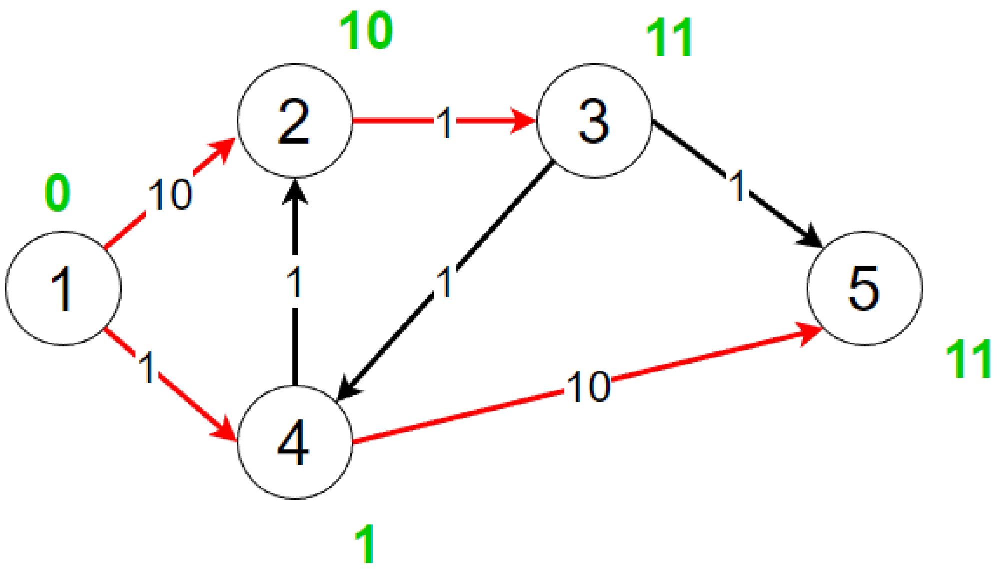
The topological sort based algorithm works as follows



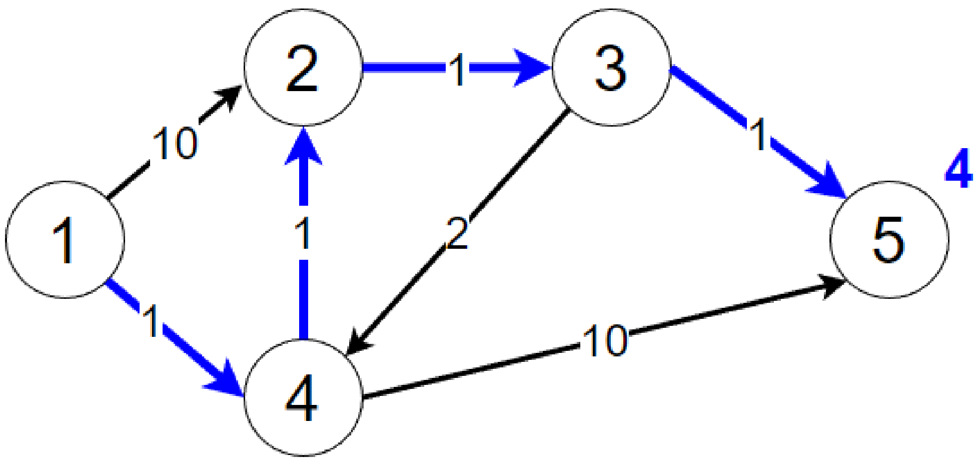


Topological sort based (lecture 5)

… and builds this solution:



But the optimal solution is this:



Problem definition

Given: a weighted graph $G(V, E)$, edge weights $w: E \to R$.

Problem 1: For vertices $s \in V$ (*source*) and $t \in V$ (*target*) find the distance and the shortest path from s to t.

Problem 2: For a vertex $s \in V$ (*source*) find distances and the shortest paths from s to every other vertex.

Problem 3: Find distances and the shortest paths from s to t for all pairs of vertices.

If there are several shortest paths between two vertices, (usually) it is enough to find any of them.

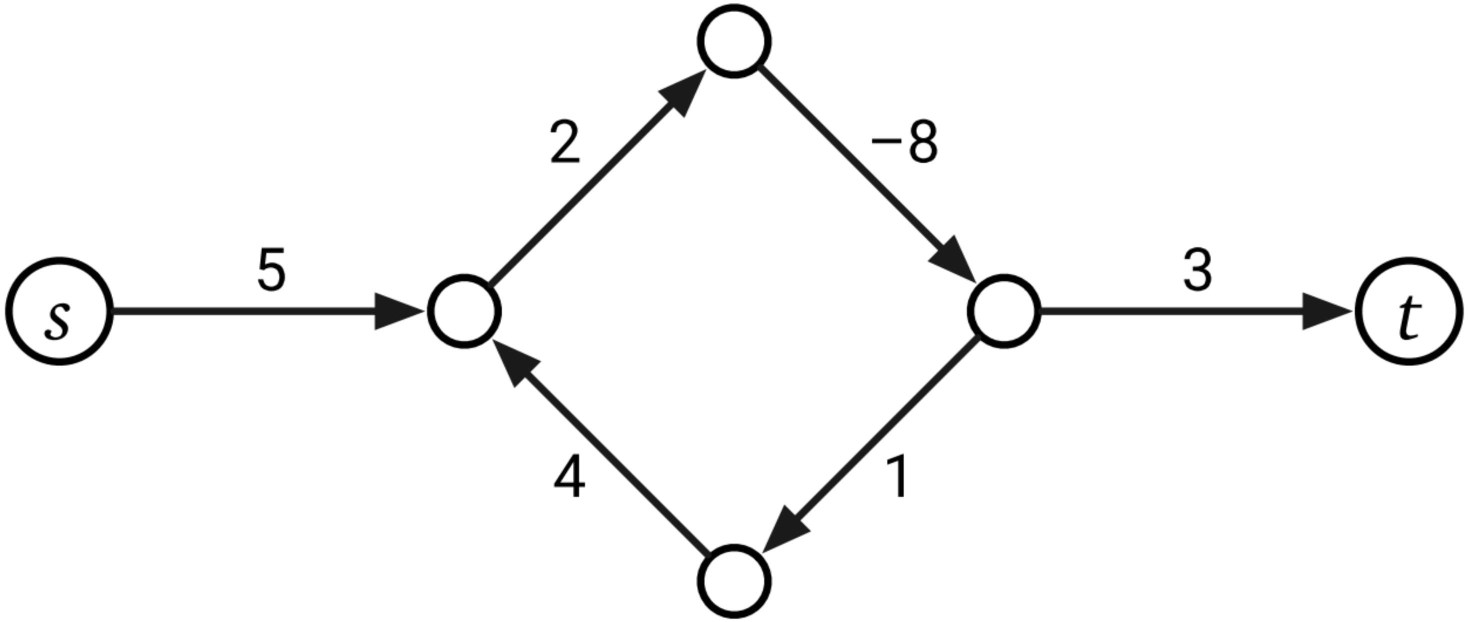
Negative weights

With respect to algorithmic issues, it is convenient to distinguish the general case and the case of problems with non-negative weights.

Reason: negative edge weights make several difficulties for algorithms, and for many practical applications weights are naturally non-negative.

Negative weights

If a graph contains a negative cycle (cycle whose total weight is negative), some pairs of vertices have no shortest paths.



http://jeffe.cs.illinois.edu/teaching/algorithms/

Negative weights

Definition. A path is called *simple* iff it does not contain any edge more than once.

For any pair of vertices s and t, if t is reachable from s then there is a shortest simple path from s and t, even in case of negative cycles. But if there are negative cycles, finding a shortest path becomes an NP-hard problem, i.e. it cannot be solved efficiently.

Negative weights

If a <u>directed</u> graph has negative edges but has no negative cycles, the shortest path problem can be solved efficiently with the algorithms considered in this lecture.

For <u>undirected</u> graphs, there are specialized algorithms, that will not be studied in this course.

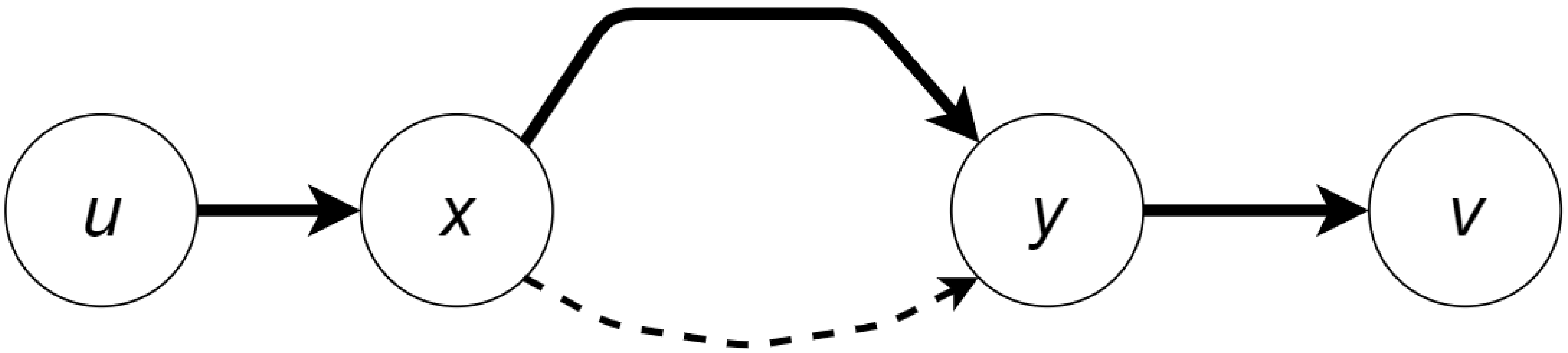
Principle of optimality

The principle of optimality is the basic condition for applicability of dynamic programming for optimization problems.

Principle of optimality for the shortest path problem. Let $G(V, E)$ be a graph with non-negative edge weights $w: E \to R_+$ and u, v – two vertices of G. Any part of a shortest path between u and v is a shortest path between its endpoints.

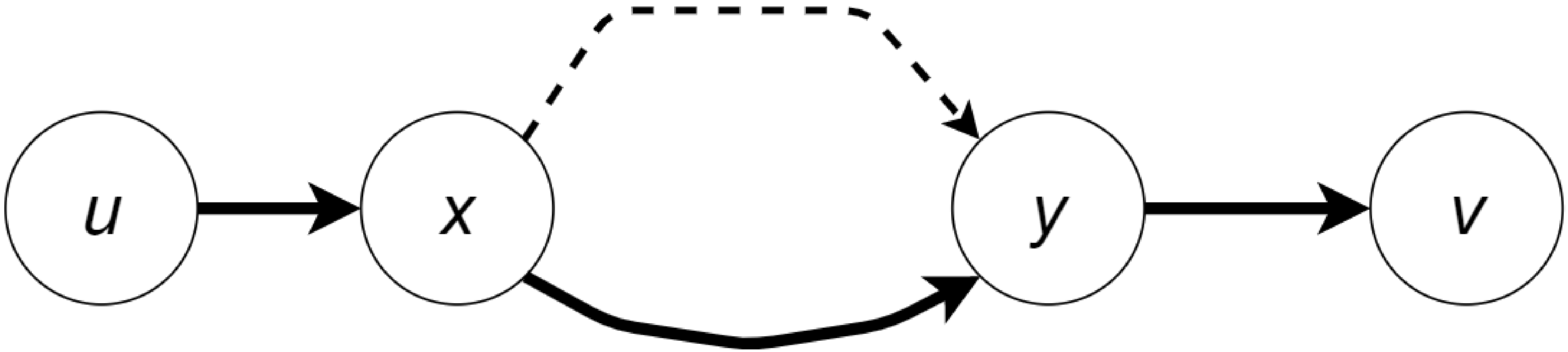
Principle of optimality

<u>**Proof**</u>. Let us consider a path p between u and v, which goes through vertices x and y (it may be that $x = u$ or $y = v$). Suppose that the part of p between x and y is not the shortest path between x and y.



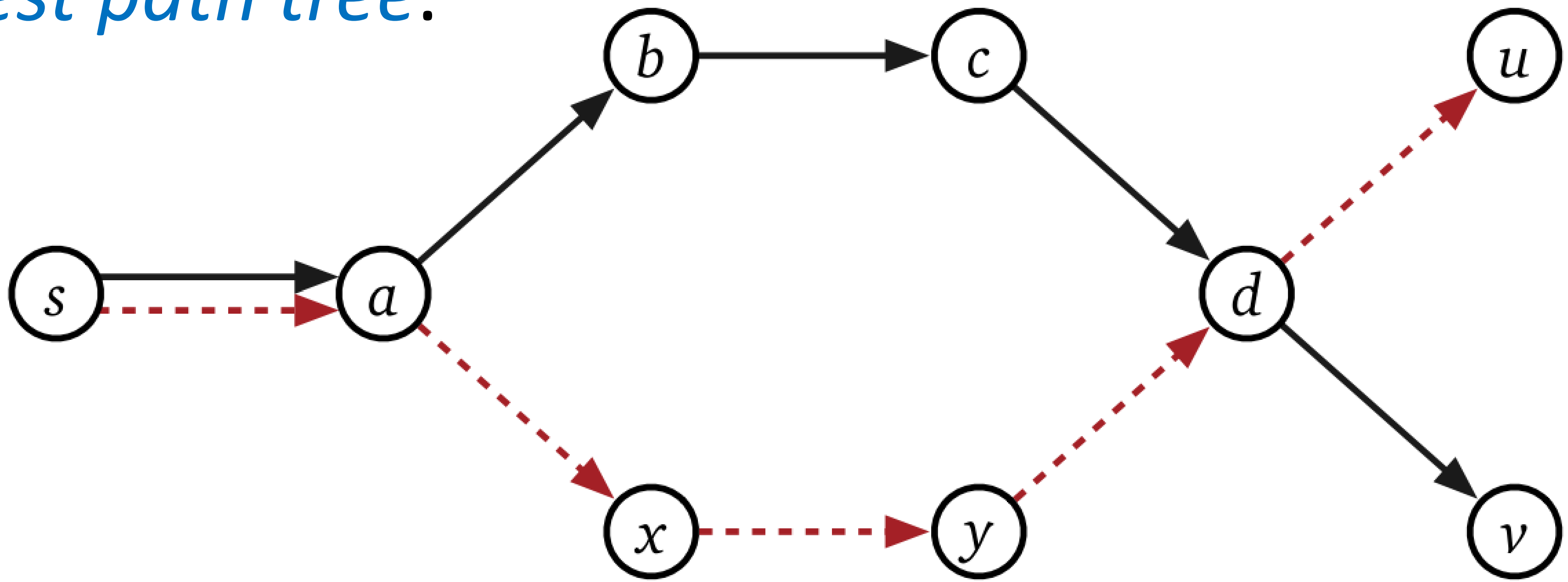
Principle of optimality

Let us replace this part of p with the shortest path between x and y. We will yield the path p', whose weight is less then the weight of p. Thus, p is not the shortest path between u and v.



Principle of optimality

Due to the principle of optimality, the shortest paths from a given source vertex to all other vertices make the *shortest path tree*.



Principle of optimality

For undirected graphs with negative edges, the principle of optimality does not hold.