Graphs

Single Source Shortest Paths for DAGs

Paolo Camurati and Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

Shortest path on weighted DAGs

- For a DAG the SSSP problem can be solved with a simplified algorithm
- Shortest paths are always well defined even if there are negative-weight edges
  - Obviously negative-weight cycles cannot exist

Shortest path on weighted DAGs

- As there are no cycles it is enough to
  - Topologically sort the DAG
    - Impose a linear order on the vertices

Perform a DFS computing end-processing times
Order vertices using the end-processing times

  - Relax all vertices following the sorted order gien by the topological sort
    - In other words, it suffices to make just one pass over the vertices in the topological sorted order
    - As we process a vertex, we relax each edge that leaves the vertex

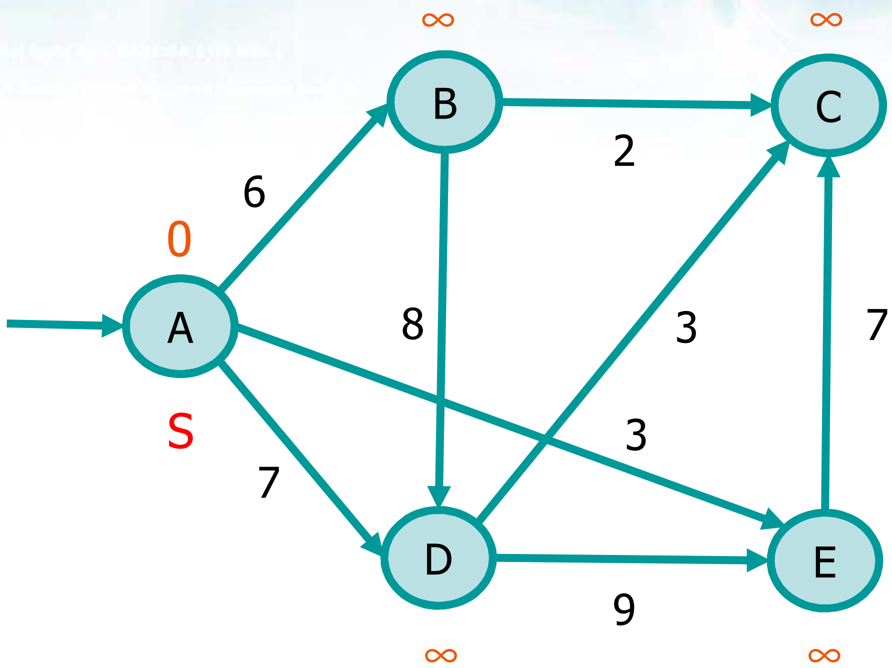
SSSP for DAGs

Pseudo-code

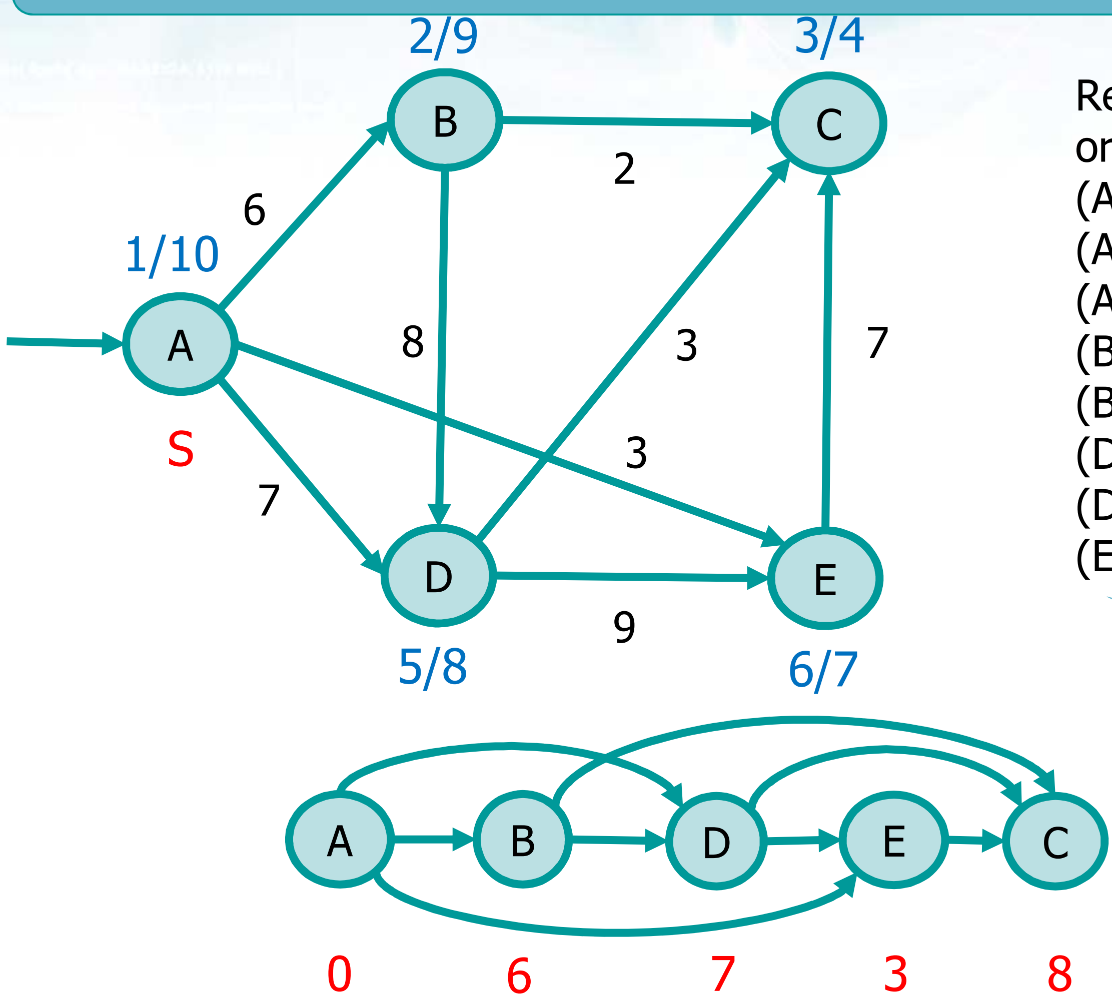
```
sssp_for_DAGs (G, w, s)
  topological sort the vertices of G
  initialize_single_source (G, s)

  for each vertex u ∈ V
    for each vertex v ∈ adjacency list of u
      relax (u, v, w)
```

Taken in topologically sorted order

Example



Example



Relaxation order
(A, B)
(A, D)
(A, E)
(B, D)
(B, C)
(D, E)
(D, C)
(E, C)

Relaxation order

Complexity

Pseudo-code

```
sssp_for_DAGs (G, w, s)
  topological sort the vertices of G
  initialize_single_source (G, s)

  for each vertex u ∈ V
    for each vertex v ∈ adjacency list of u
      relax (u, v, w)
```

$\Theta(|V|+|E|)$

$\Theta(|V|)$

Executed E times alltogheter

$\Theta(1) \rightarrow \Theta(|E|)$

Taken in topological sorted order

Overall running time complexity
$T(n) = \Theta(|V| + |E|)$

Complexity

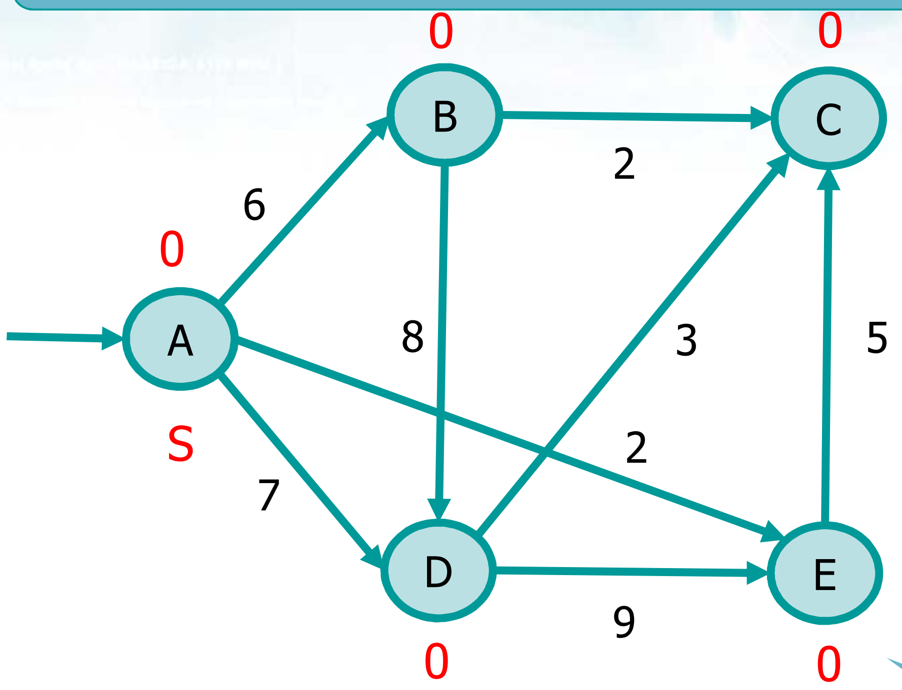
- Applicable on DAGs with negative edges
  - No negative-weigth cycles can exist
- $T(n) = O(|V|+|E|)$

Longest path on weighted graph

- Problem intractable on generic weighted graph
- As on a DAG there are no cycles, the problem become computationally feasible
  - Topologically sort the DAG
  - For all ordered vertices
  - Apply the "inverse" relaxation rule starting from that vertex

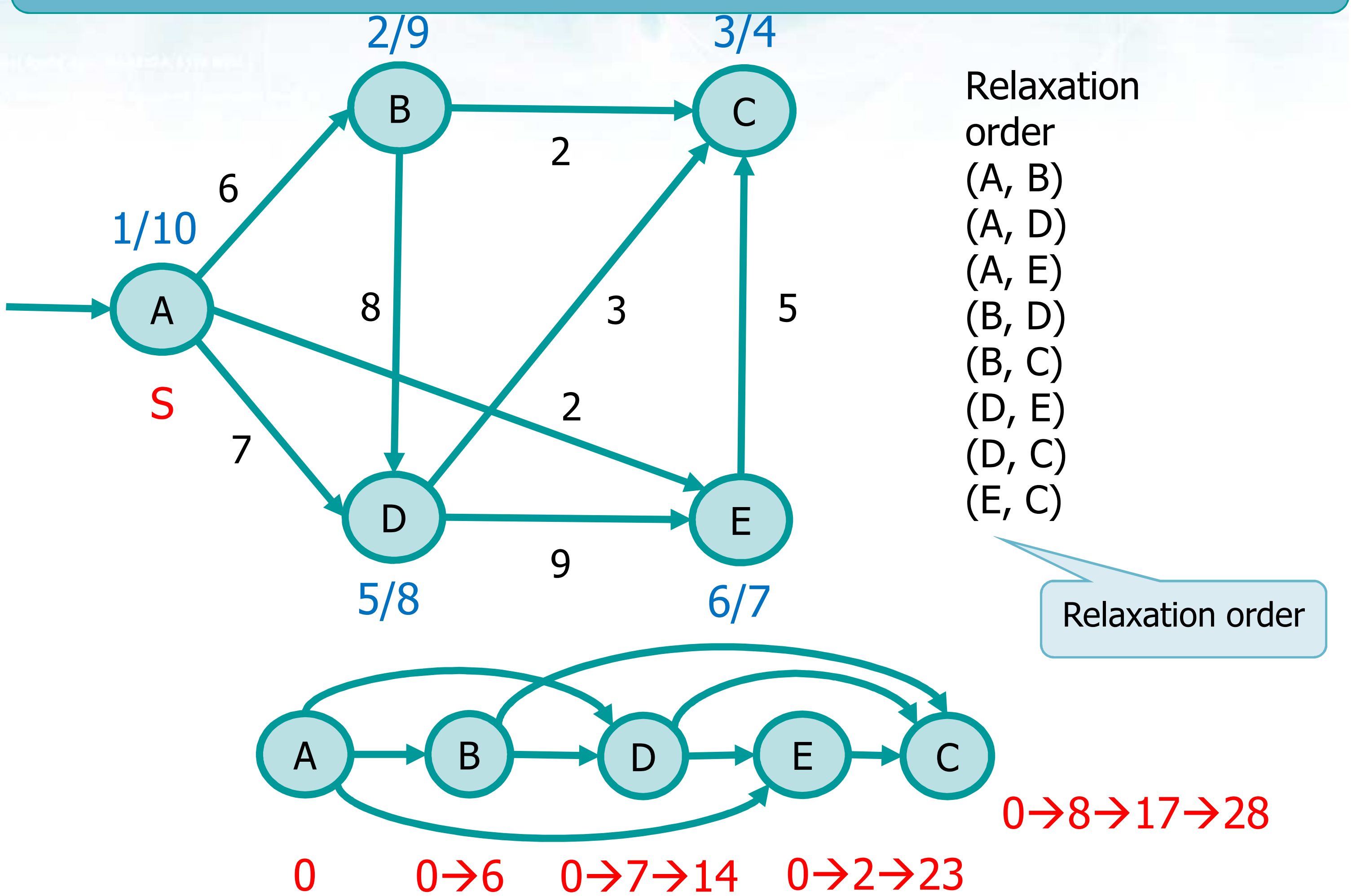
```
inverse_relax (u, v, w)
  if (v.d < u.d + w(v,u)) {
    v.d = u.d + w(v,u)
    v.pred = u
  }
```

Example



The initial estimate is equal to zero for all vertices

Example



Relaxation order
(A, B)
(A, D)
(A, E)
(B, D)
(B, C)
(D, E)
(D, C)
(E, C)

Complexity

- As the algorithm analized for the shortest paths for DAGs