Algorithms and Data Structures

Module 1

Lecture 5

Efficient algorithms for DAGs, part 2

Adigeev Mikhail Georgievich

mgadigeev@sfedu.ru

Algorithms for DAGs

Let us consider weighted DAGs: G(V,E), $w: E \rightarrow R$.

Weights are represented as a matrix W: w_{ij} is the weight of arc (i, j). If there is no arc (i, j) on G, the value of w_{ij} should be assigned depending on the problem we are solving.

Shortest path

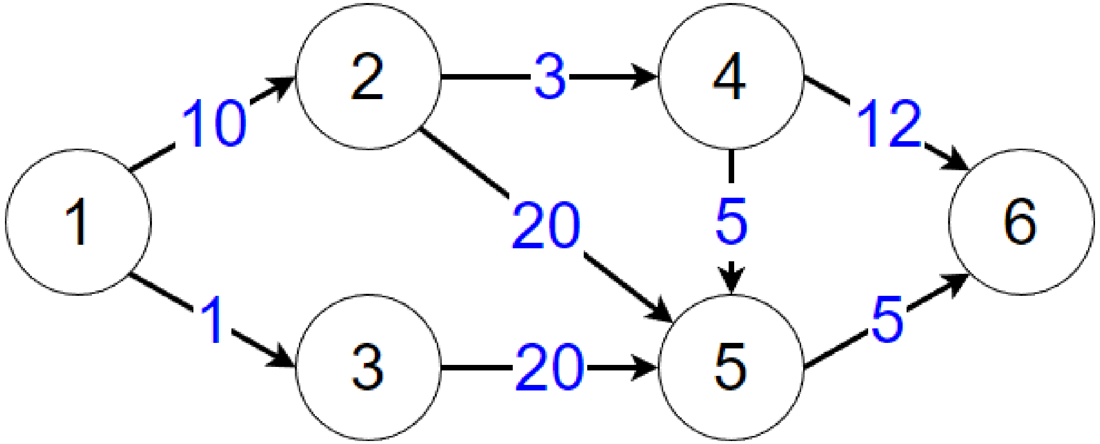
The shortest path problem

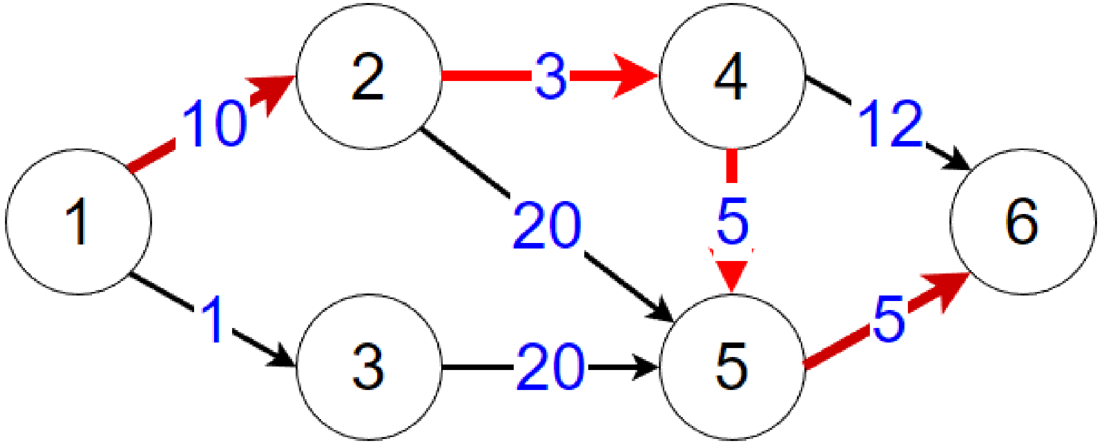
- Let us define the ***path weight*** as the sum of arcs' weights of the path.
- The shortest path is defined as the path of the minimum weight.
- There are 3 possible versions of the shortest path problem:

1) For the given vertices $s, t \in V$, build the shortest path from s to t.
2) For the given vertex $s \in V$, build the shortest paths from s to all other vertices.
3) Build the shortest paths between all pairs of vertices on the given graph.

Let us consider versions (1) and (2) for the case the given graph is a DAG.

Shortest path





Shortest path

Algorithm for building shortest paths on a DAG

Input:

a) The weight matrix W. For absent arcs (i, j) set $w_{ij} = +\infty$.

b) Vertex $s \in V$.

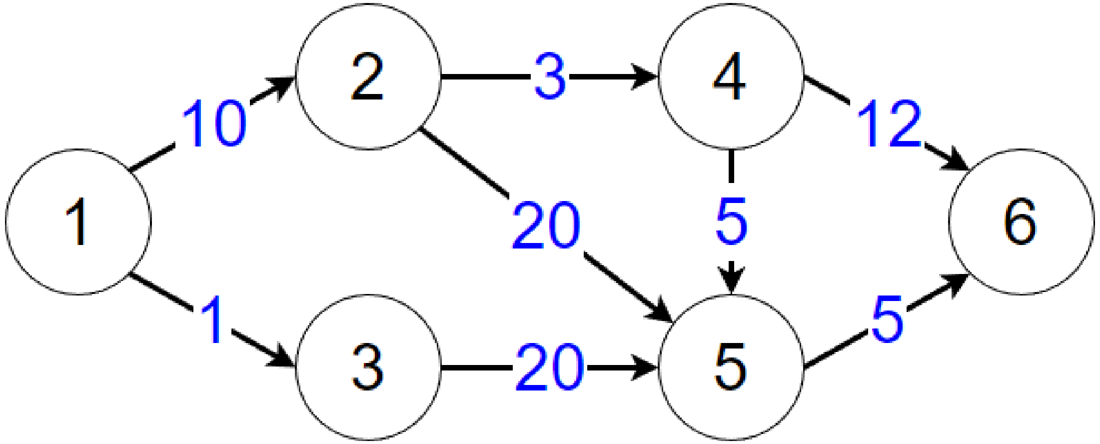
Output:

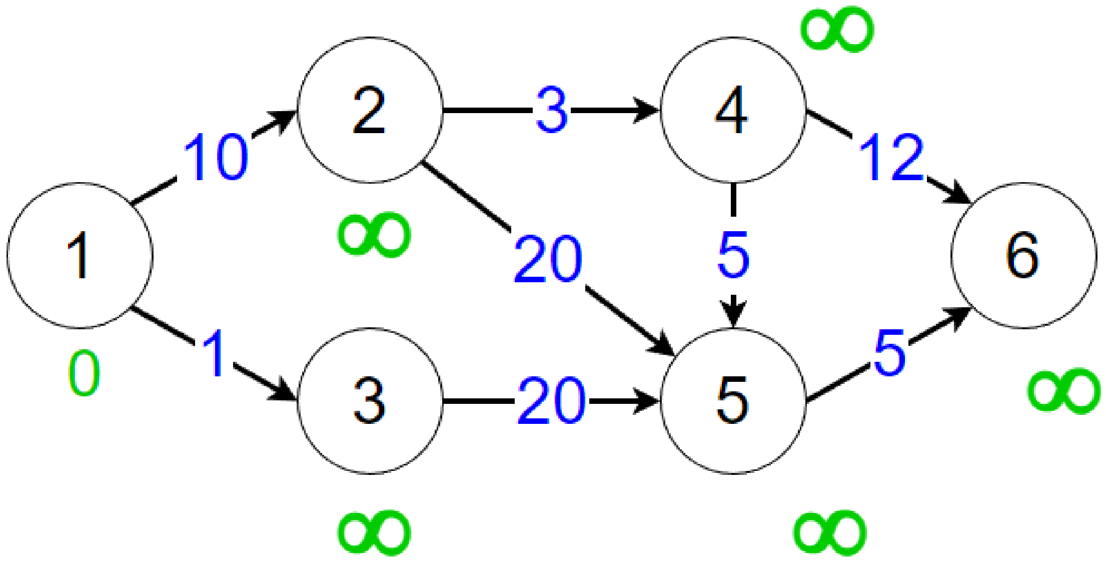
1) Array D: $d[i]$=distance (the shortest path weight) from s vertex i.
2) Array P: $p[i]$= the vertex which is penultimate (second to last) in the shortest path from s to i.

Shortest path

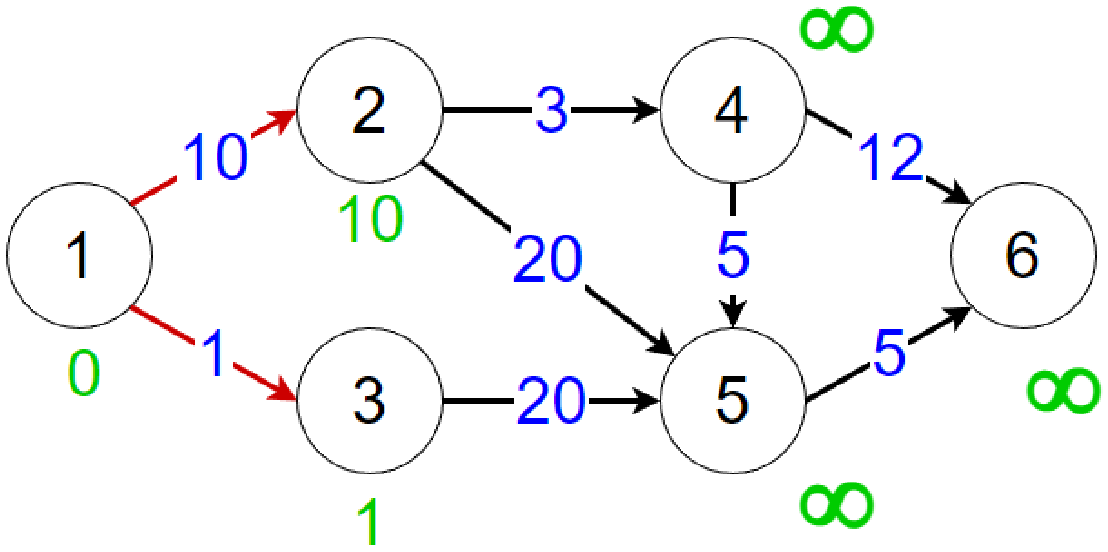
```
For each v ∈ V:
    {
        d[i] := +∞;
        p[i] := NULL;
    }
d[s] := 0;
Build the topological sorting of graph vertices;
For each v ∈ V in topological order:
    For each u in Adj(v)
        If d[u] > d[v] + w[v,u]:
        {
            d[u] := d[v] + w[v,u];
            p[u] := v;
        }
```

Shortest path

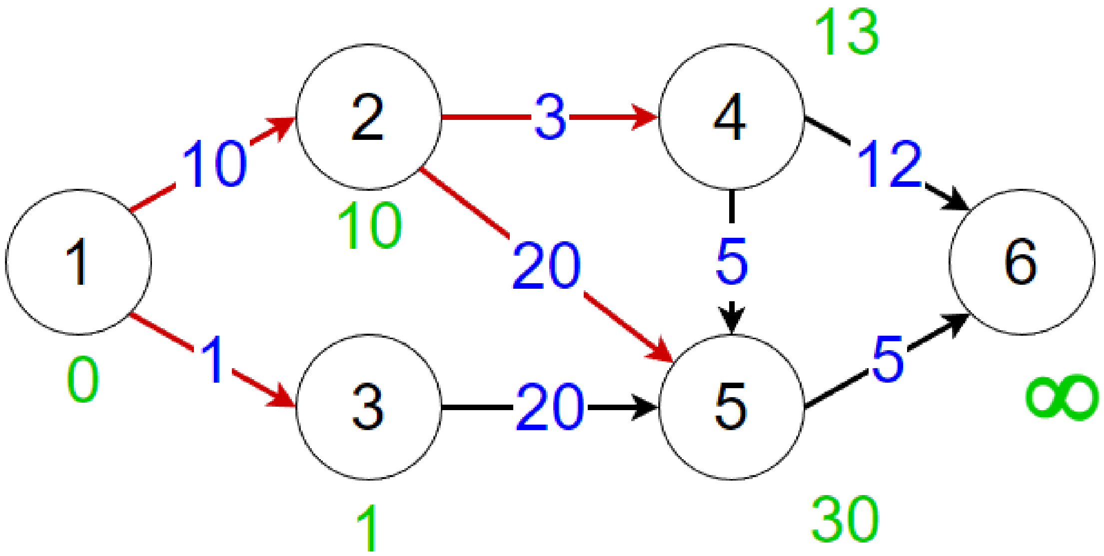




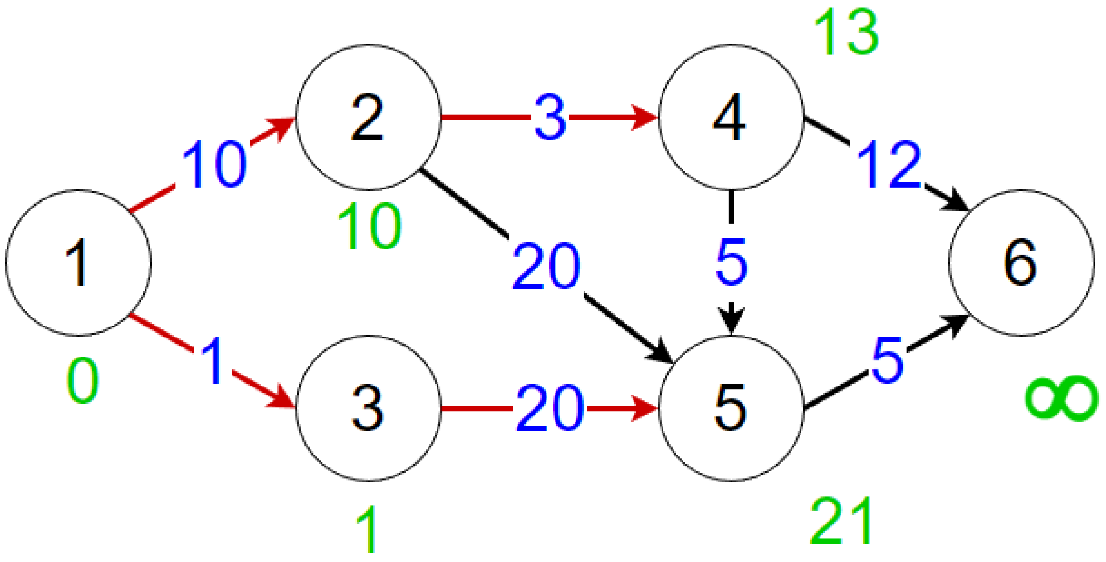
Shortest path



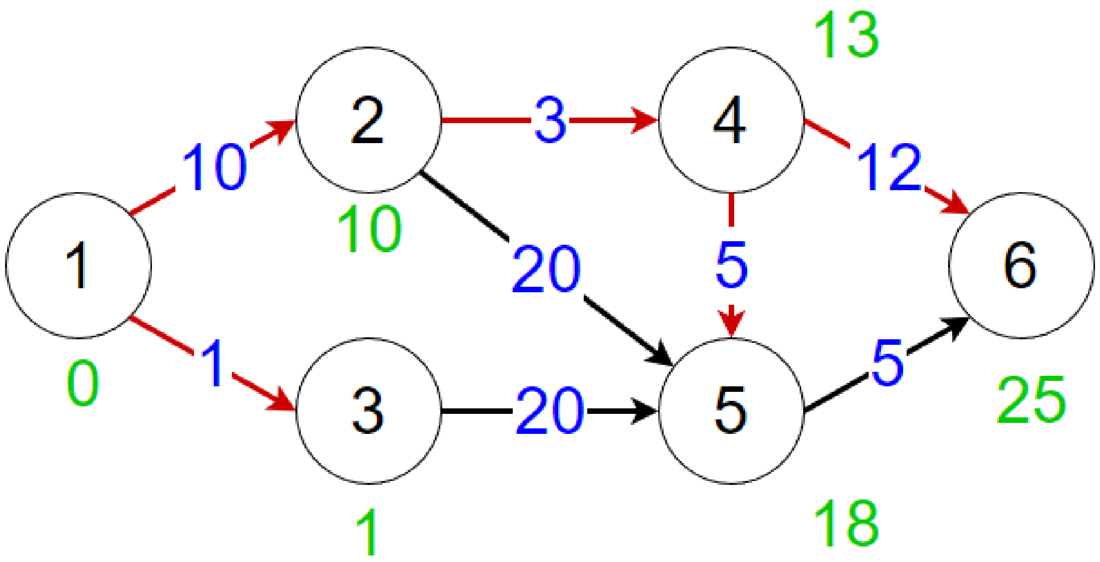
Shortest path



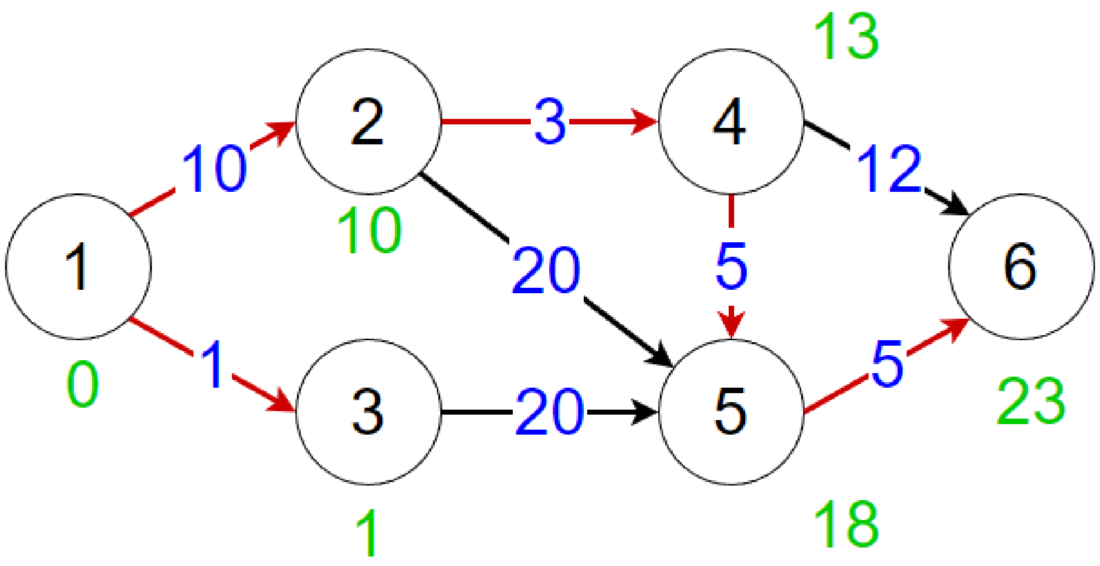
Shortest path



Shortest path



Shortest path



Shortest path

Time complexity of the algorithm: $O(n + m)$.
Let us prove its correctness.

<u>Theorem</u>. Upon the finish of the algorithm, for each $v \in V$ the value $d[v]$ is equal to the distance from s to v.

Proof

Let us use the mathematical induction method with the topological number (TN) as the induction parameter.

<u>Base step</u>: TN=0. This is true for the vertex *s* only. For *s*, the algorithm sets $d[s] = 0$ at the initialization phase and this value does not change further.

<u>Induction hypothesis</u>: assume that for all vertices with TN < *k* the statement holds.

<u>Inductive step</u>: let us consider a vertex *u*: TN(*u*)=*k*.

Shortest path

<u>Inductive step</u>: let us consider a vertex u: TN(u)=k.

Since the graph is a DAG, for u there is a minimum weight path. Let us denote it $\pi = s$-a-b-…-v-u.

For all vertices of this path, the condition TH $< k$ holds. Thus, the induction hypothesis holds for these vertices, so $d[x]$ = minimum weight of a path from s to x.

Upon processing vertex v, the following condition holds: d[u] <= d[v]+w[v,u].

But we see that d[v]+w[v,u] is the weight of the shortest path π. Thus, d[u] = minimum weight of a path from s to u. And this value does not change at further iterations.

QED.

Longest path

<u>The longest path problem</u>

- Let us define the ***path weight*** as the <u>sum</u> of arcs' weights of the path.
- The ***longest*** path is defined as the path of the ***maximum*** weight.

For the general case the longest path problem is computationally hard. But for DAGs we can build an efficient algorithm.

We can consider 2 possible ways to solve this problem:

1) Reduce it to the shortest path problem by multiplying the weights by -1.
2) Modify that algorithm.

Longest path

```
For each v ∈ V:
     {
          d[i] := −∞;
          p[i] := NULL;
     }
d[s] := 0;
Build the topological sorting of graph vertices;
For each v ∈ V in topological order:
     For each u in Adj(v)
          If d[u] < d[v] + w[v,u]:
          {
               d[u] := d[v] + w[v,u];
               p[u] := v;
          }
```

The most reliable path

The most reliable path problem

- The arc weight denotes the probability (chance) to successfully go through the arc.
- Let us define the ***path weight*** as the product of arcs' weights of the path.
- The ***most reliable*** path is defined as the path of the ***maximum*** weight.

Let us consider one more modification of the basic algorithm.

The most reliable path

```
For each v ∈ V:
      {
            d[i] := 0;
            p[i] := NULL;
      }
d[s] := 1;
Build the topological sorting of graph vertices;
For each v ∈ V in topological order:
      For each u in Adj(v)
            If d[u] < d[v] * w[v,u]:
            {
                  d[u] := d[v] * w[v,u];
                  p[u] := v;
            }
```

The maximum capacity path

<u>The maximum capacity path problem</u>

- The arc weight denotes the *capacity* (the maximum possible flow through the arc) of the arc.
- Let us define the ***path weight*** as the <u>minimum</u> of arcs' weights of the path.
- The ***maximum capacity*** path is defined as the path of the ***maximum*** weight.

Let us consider one more modification of the basic algorithm.

The maximum capacity path

```
For each v ∈ V:
      {
            d[i] := 0;
            p[i] := NULL;
      }
d[s] := +∞;
Build the topological sorting of graph vertices;
For each v ∈ V in topological order:
      For each u in Adj(v)
            If d[u] < min{d[v],w[v,u]}:
            {
                  d[u] := min{d[v],w[v,u]};
                  p[u] := v;
            }
```