Elementary Graph Algorithms

Slides from Heejin Park

Contents

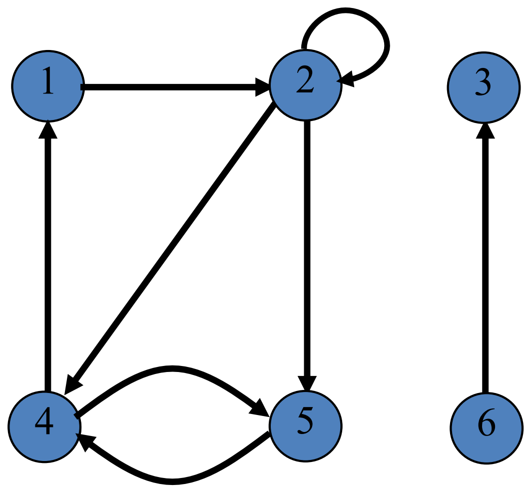
- **Graphs**
  - Graph basics
  - Graph representation
- **Searching a graph**
  - Breadth-first search
  - Depth-first search
- **Applications of depth-first search**
  - Topological sort

- A ***graph*** G is a pair (V, E)

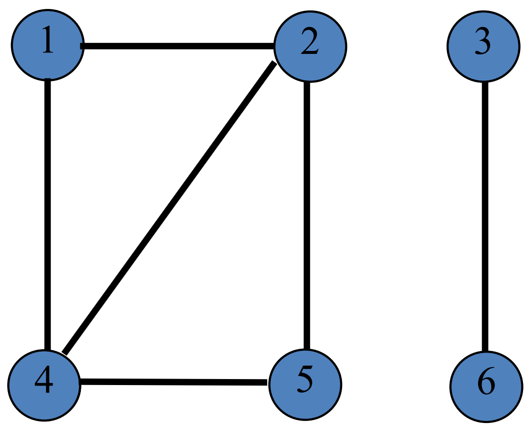
  where V is a ***vertex*** set and E is an ***edge*** set.

- A vertex (node)
  - a circle.

- An edge (**link**).
  - an arrow or a line.

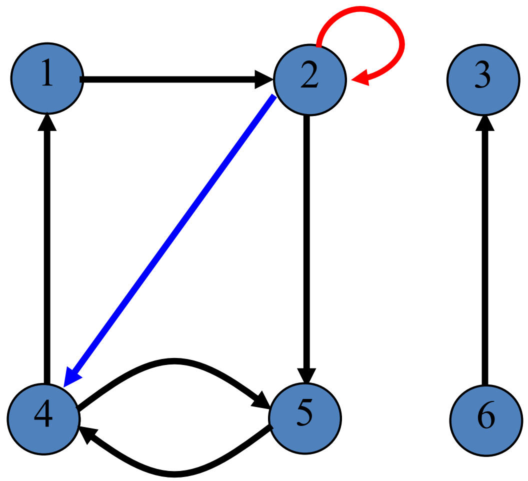


A directed graph

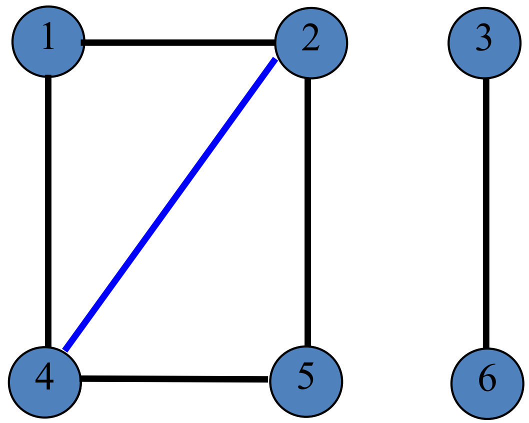


An undirected graph

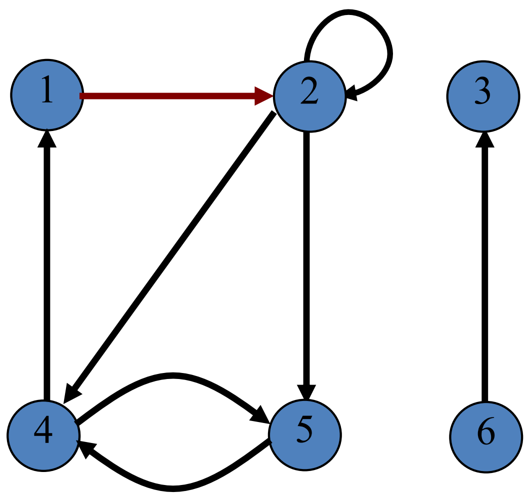
- ***Directed graph*** (or ***digraph***)
  - The blue edge ***leaves*** vertex 2 and *enters* vertex 4.
  - The blue edge is ***incident from*** vertex 2 and ***incident to*** vertex 4.
  - The red edge is a ***self-loop.*** (an edge from a vertex to itself)



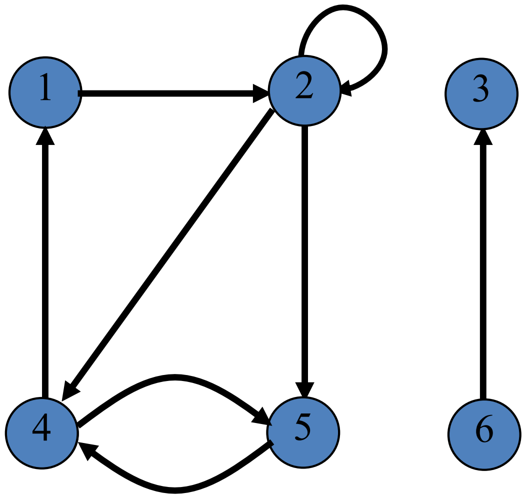
- ***Undirected graph***.
  - Edges have no directions
  - No Self-loops



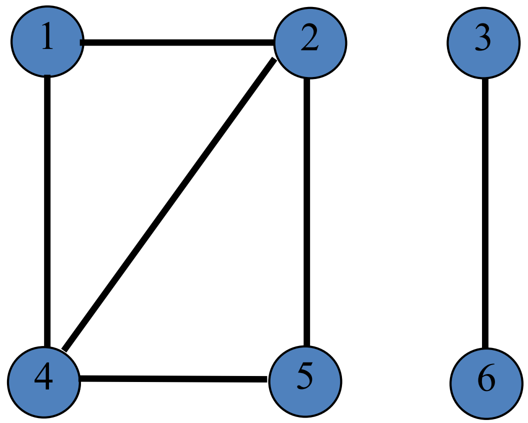
- ***Adjacency***
  - If (u,v) is an edge, vertex v is ***adjacent*** to vertex u.
  - In an undirected graph, adjacency relation is symmetric.
    - If u is adjacent to v, v is adjacent to u.
  - In a directed graph, it is not symmetric.
    - Vertex 2 is adjacent to 1.
    - But vertex 1 is not adjacent to 2.



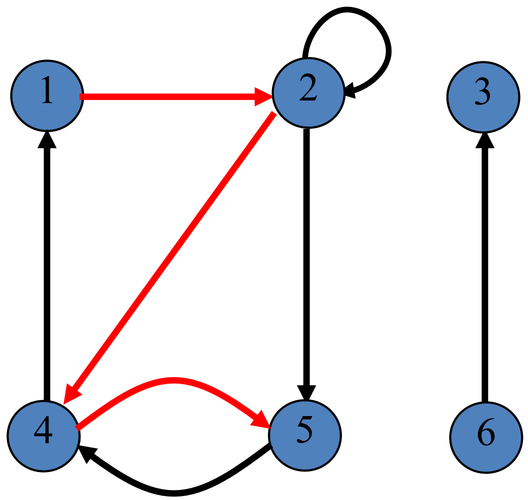
- ***Degree***
  - The ***out-degree*** of vertex 2 is 3.
  - The ***in-degree*** of vertex 2 is 2.
  - ***degree* = out-degree + in-degree.**



- ***Degree***
  - In an undirected graph,
    - The **degree** of vertex 2 is 3.

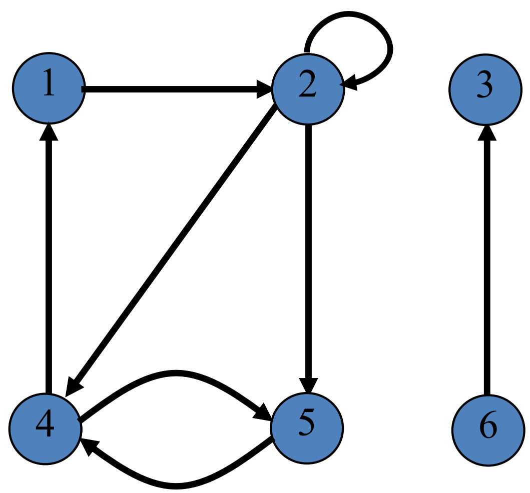


- ***Path***
  - A sequence of consecutive edges
    - <1, 2>, <2, 4>, <4, 5> is a path.
    - <1, 2, 4, 5> for short
    - <1, 2, 4, 1, 2> is a path.
    - <1, 2, 4, 2> is not a path.

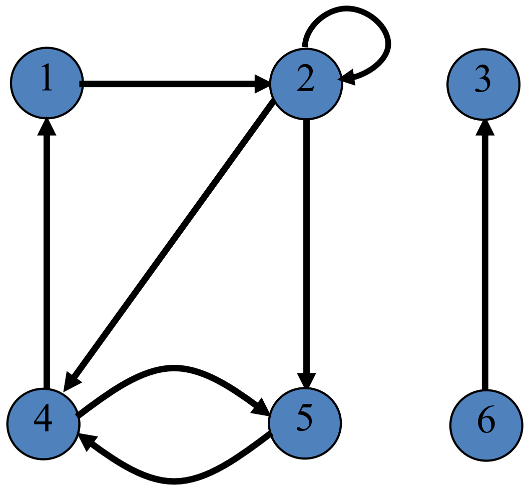


- ***Path***
  - The ***length*** of a path is the number of edges in the path.
    - The length of a path <1, 2, 4, 5> is 3.
    - If there is a path from vertex u to vertex v,

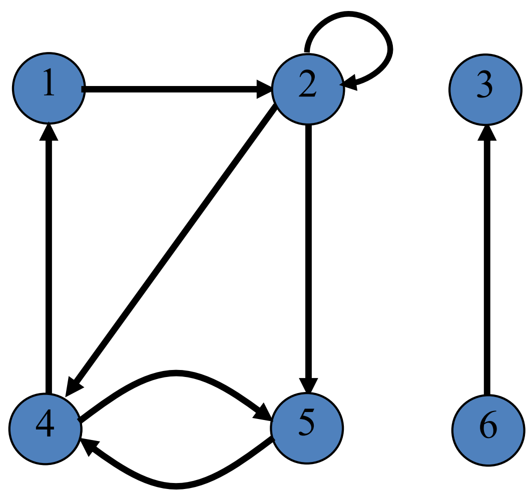
      v is called ***reachable*** from u.
      - Vertex 5 is reachable from vertex 1.
      - Vertex 3 is not reachable from vertex 1.



- ***Simple path***
  - A path is simple if all vertices in the path are distinct.
  - A path <1, 2, 4, 5> is a simple path.
  - A path <1, 2, 4, 1, 2> is not a simple path.

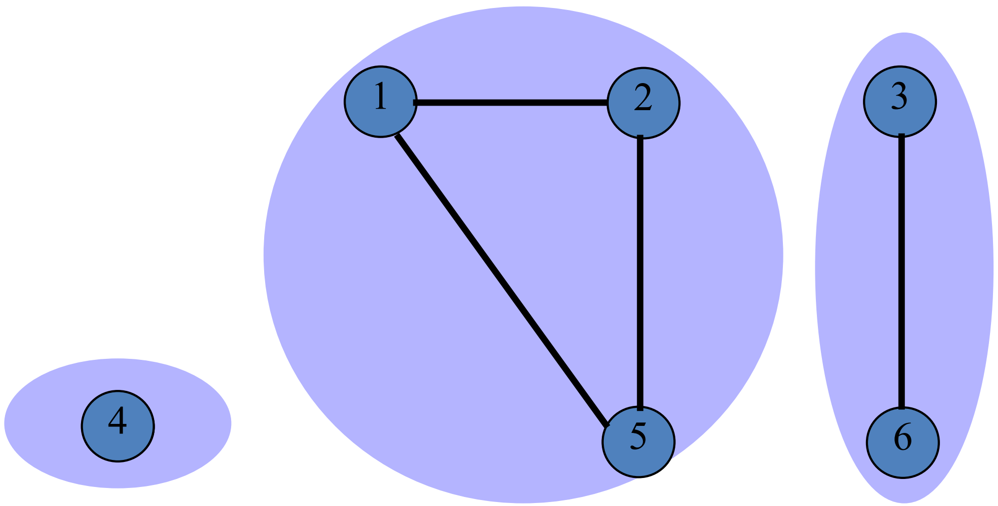


- ***Cycle* and *simple cycle***
  - A path $<v_0, v_1, v_2,\ldots, v_k>$ is a cycle if $v_0 = v_k$
  - A cycle $<v_0, v_1, v_2,\ldots, v_k>$ is simple if $v_1, v_2,\ldots, v_k$ are distinct.
  - A path <1, 2, 4, 5, 4, 1> is a cycle but it is not a simple cycle.
  - A path <1, 2, 4, 1> is a simple cycle.

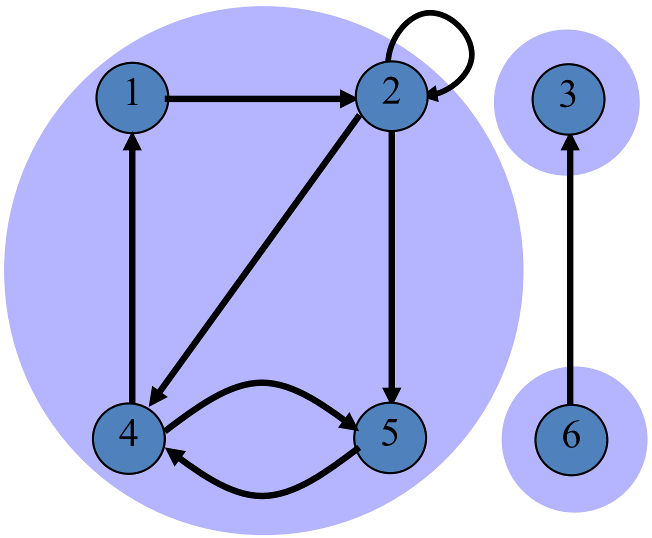


- ***An acyclic graph***
  - A graph without cycles
- ***A connected graph***
  - An undirected graph is ***connected***

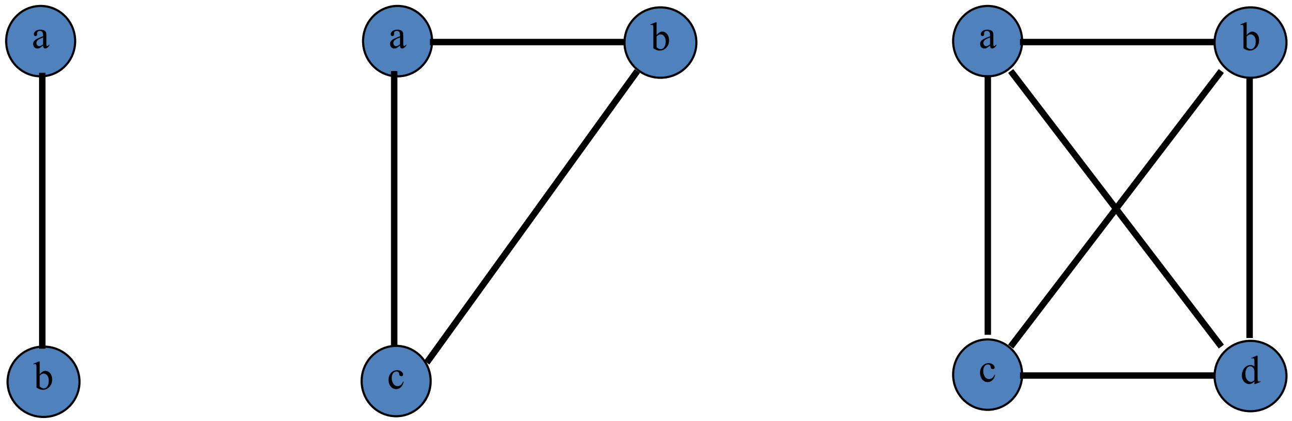
    if every pair of vertices is connected by a path.
- ***Connected components***
  - Maximally connected subsets of vertices of an undirected graph.



- ***Strongly connected***
  - A directed graph is ***strongly connected*** if every pair of vertices is reachable from each other.
- ***Strongly connected components***
  - Maximally strongly connected subsets of vertices in a directed graph.

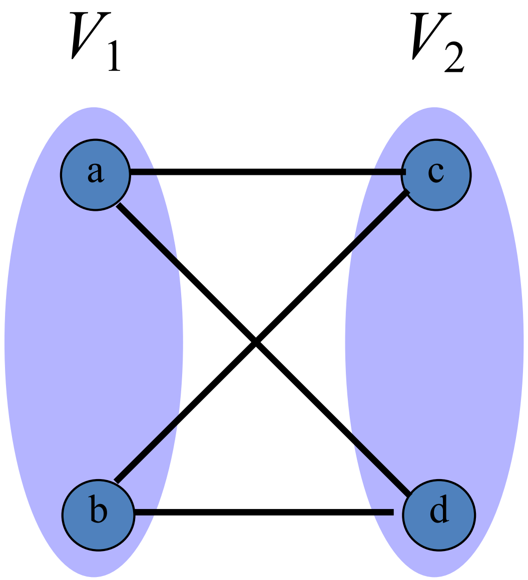


- ***A complete graph***
  - An undirected graph in which every pair of vertices is adjacent.

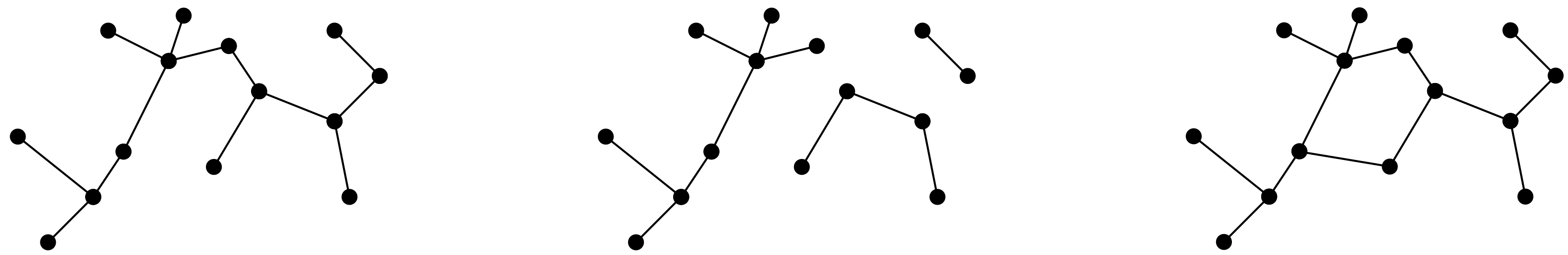


  - The number of edges with n vertices?

- ***A bipartite graph***
  - An undirected graph $G = (V,E)$ in which
    V can be partitioned into two sets V_1 and V_2 such that
    for each edge (u,v), either $u \in V_1$ and $v \in V_2$, or $u \in V_2$ and $v \in V_1$.

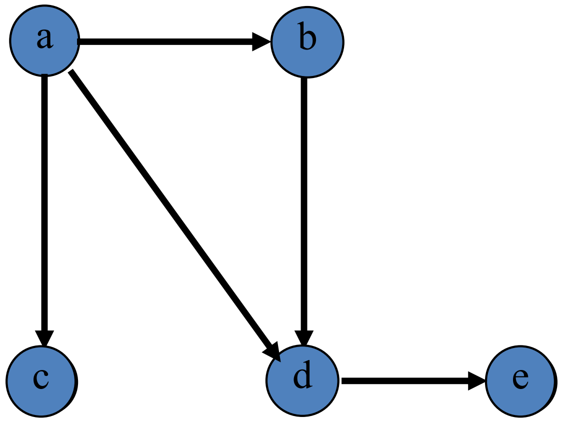


- ***Forest***
  - An acyclic, undirected graph
- ***Tree***
  - A connected forest
  - A connected, acyclic, undirected graph



  - Is a connected component of a forest a tree?

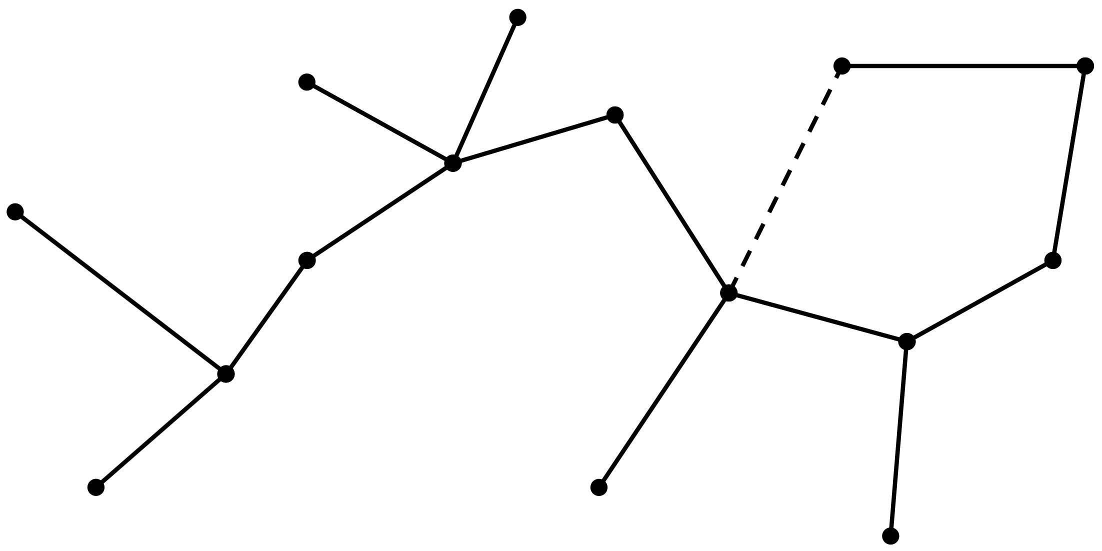
- ***Dag***
  - A directed acyclic graph



- ***Handshaking lemma***
  - If $G = (V, E)$ is an undirected graph

$$\sum_{v \in V} \mathrm{degree}(v) = 2|E|$$

- **Tree: connected, acyclic, and undirected graph**
  - Any two vertices are connected by a unique simple path.
  - If any edge is removed, the resulting graph is disconnected.
  - If any edge is added, the resulting graph contains a cycle.
  - $|E| = |V| - 1$



- **G is a tree.**

  = G is a connected, acyclic, and undirected graph

  = In G, any two vertices are connected by a unique simple path.

  = G is connected, and if any edge is removed, the resulting graph is disconnected.

  = G is connected, $|E| = |V| - 1$.

  = G is acyclic, $|E| = |V| - 1$.

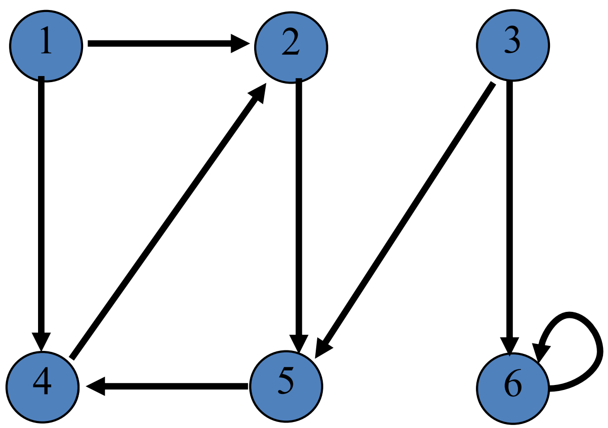
  = G is acyclic, but if any edge is added, the resulting graph contains a cycle.

- **The number of edges**
  - Directed graph
    - $|E| \leq |V|^2$
  - Undirected graph
    - $|E| \leq |V|\ (|V| - 1)\ /\ 2$

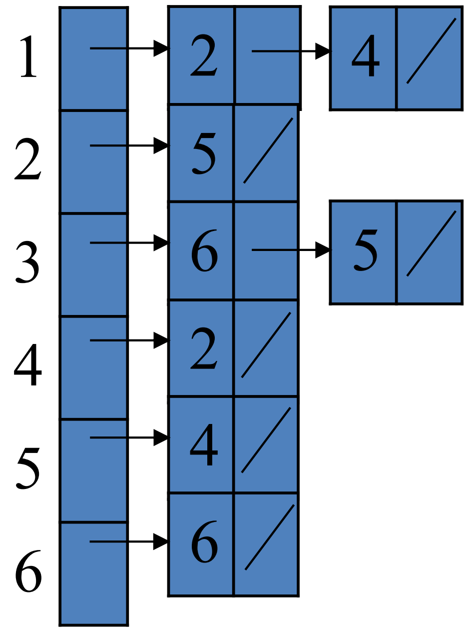
Contents

- **Representations of graphs**
  - Adjacency-list representation
  - Adjacency-matrix representation

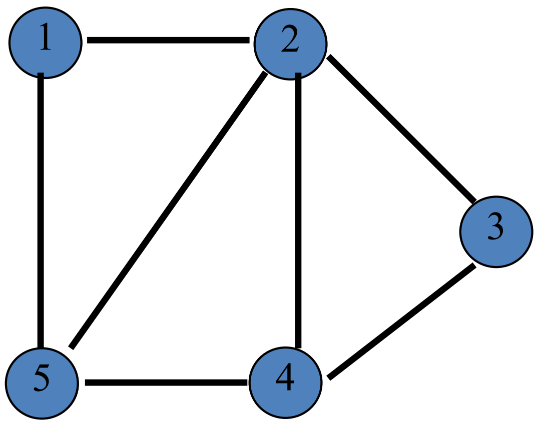
- **Adjacency-list representation**
  - An array of $|V|$ lists, one for each vertex.
  - For vertex u, its adjacency list contains all vertices adjacent to u.



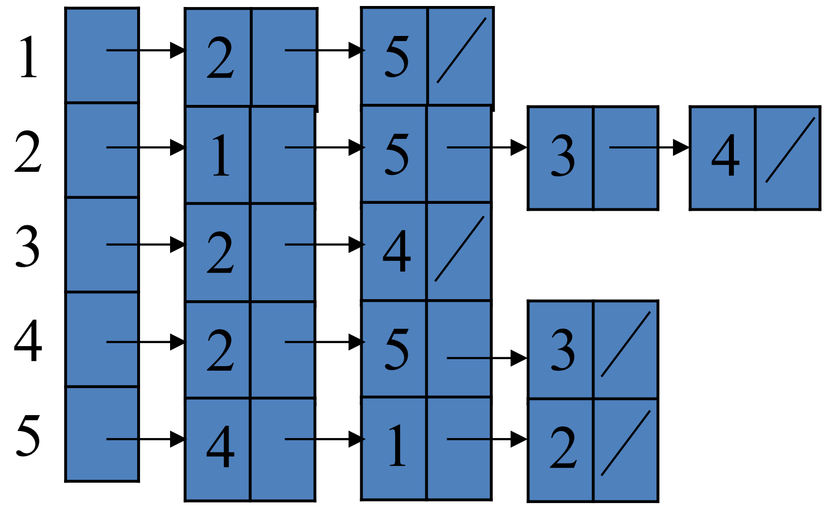
A directed graph



- **Adjacency-list representation**
  - For an undirected graph, its directed version is stored.

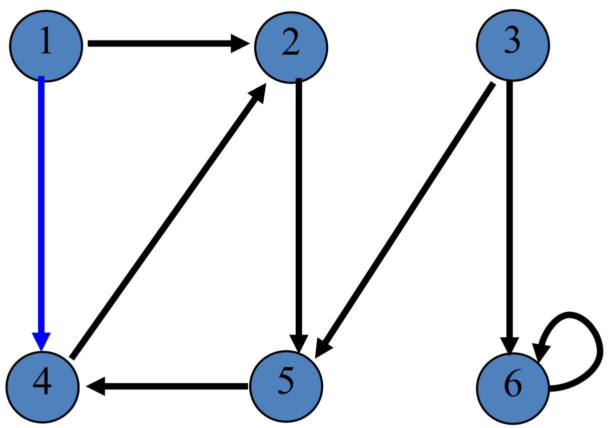


An undirected graph



  - $\Theta(V+E)$ space

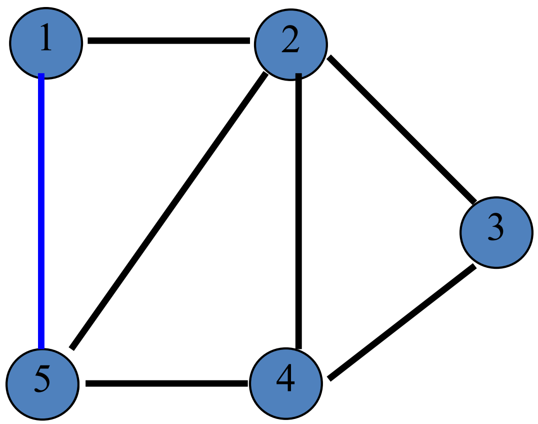
- **Adjacency-matrix representation**
  - $|V|$ x $|V|$ matrix: $\Theta(V^2)$ space
  - Entry (i,j) is 1 if there is an edge and 0 otherwise.



A directed graph

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

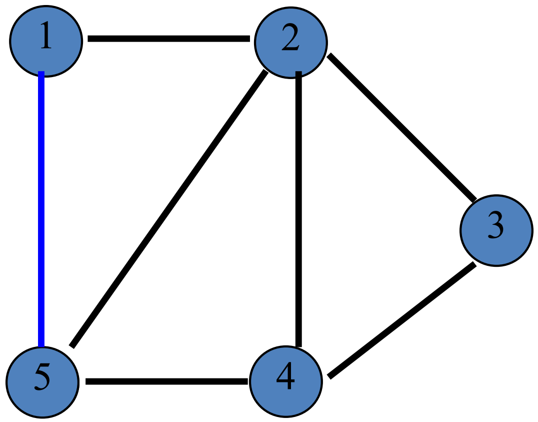
- **Adjacency-matrix representation**
  - $|V|$ x $|V|$ matrix
  - Entry (i,j) is 1 if there is an edge and 0 otherwise.



An undirected graph

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

- **Adjacency-matrix representation**
  - For an undirected graph, there is a symmetry along the main diagonal of its adjacency matrix.
  - Storing the lower matrix is enough.



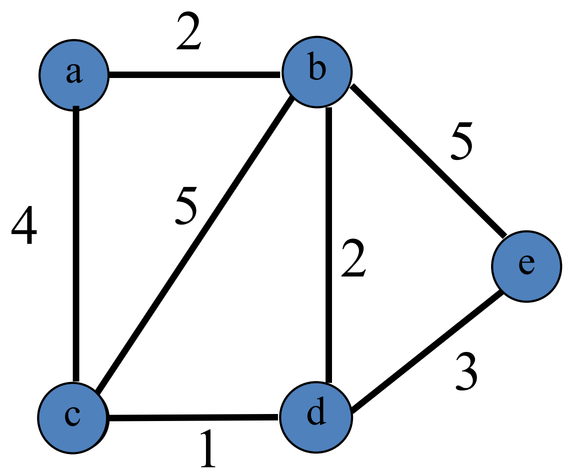
An undirected graph

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

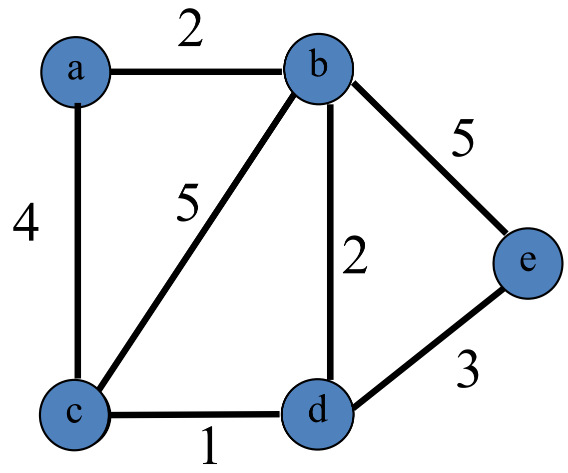
- **Comparison of adjacency list an adjacency matrix**
  - Storage
    - If G is sparse, adjacency list is better.
      - because $|E| < |V|^2$.
    - If G is dense, adjacency matrix is better.
      - because adjacency matrix uses only one bit for an entry.
  - Edge present test: does an edge (i, j) exist?
    - Adjacency matrix: $\Theta(1)$ time.
    - Adjacency list: $O(V)$ time.

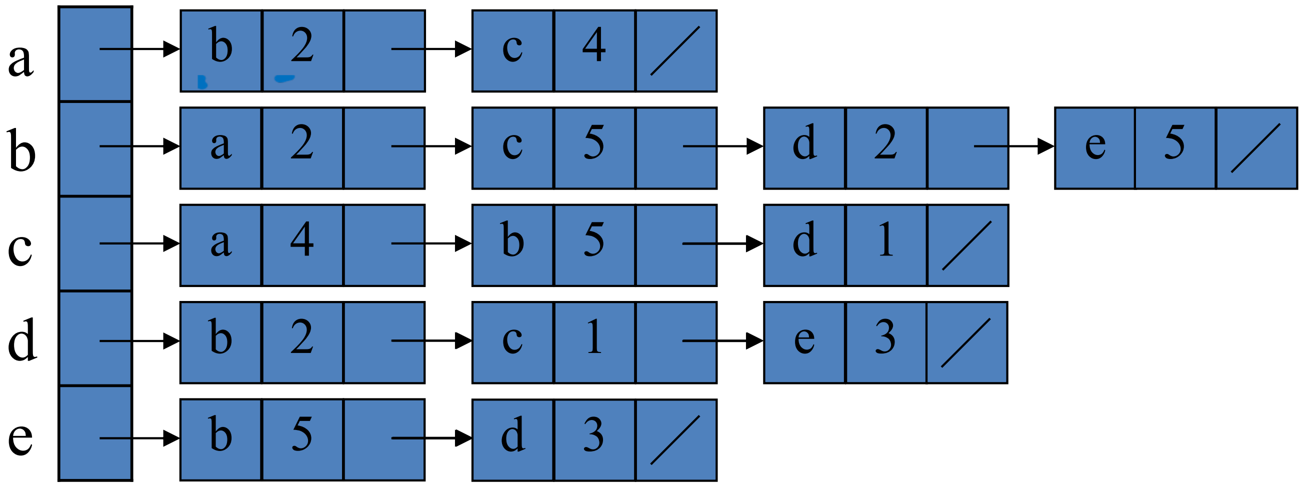
- **Comparison of adjacency list and adjacency matrix**
  - Listing or visiting all edges
    - Adjacency matrix: $\Theta(V^2)$ time.
    - Adjacency list: $\Theta(V + E)$ time.

- **Weighted graph**
  - Edges have weights.

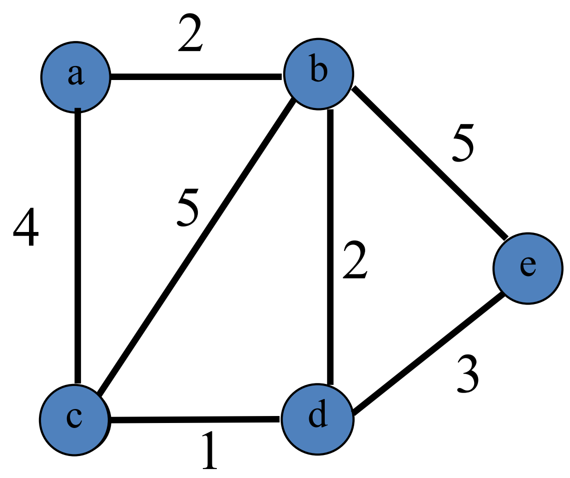


- **Weighted graph representation**
  - adjacency list





- **Weighted graph representation**
  - adjacency matrix
    - $\Theta(V^2)$ space



	a	b	c	d	e
a	0	2	4	∞	∞
b	2	0	5	2	5
c	4	5	0	1	∞
d	∞	2	1	0	3
e	∞	5	∞	3	0

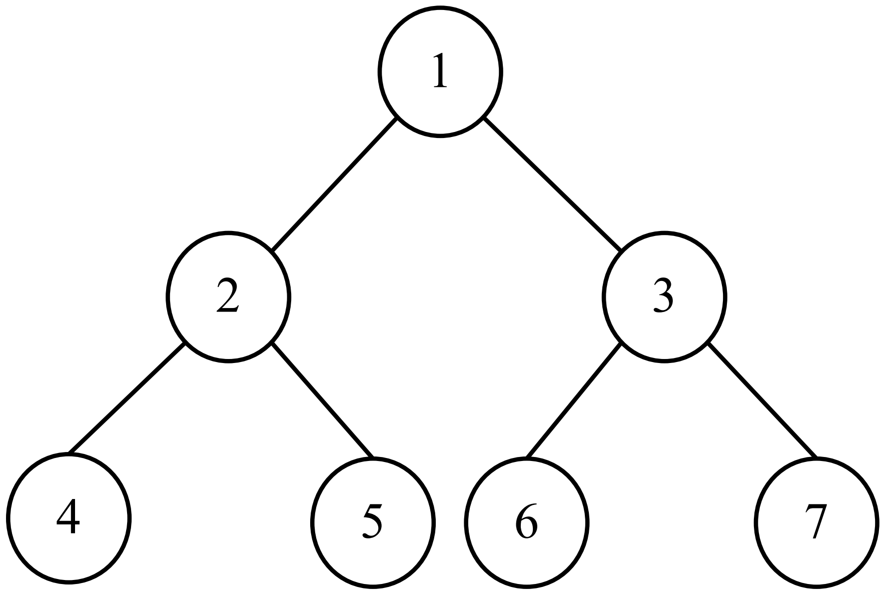
- **Transpose of a matrix**
  - The ***transpose*** of a matrix $A = (a_{ij})$ is
  - $A^T = (a_{ij}^T)$ where $a_{ij}^T = a_{ji}$
  - An undirected graph is its own transpose: $A = A^T$.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

- **Exercise 22.1-3**
  - The transpose of a directed graph
- **Exercise 22.1-4**
  - Removing duplicate edges in a multigraph in $O(V+E)$ time.
- **Exercise 22.1-6**
  - Universal sink detection in $O(V)$ time.

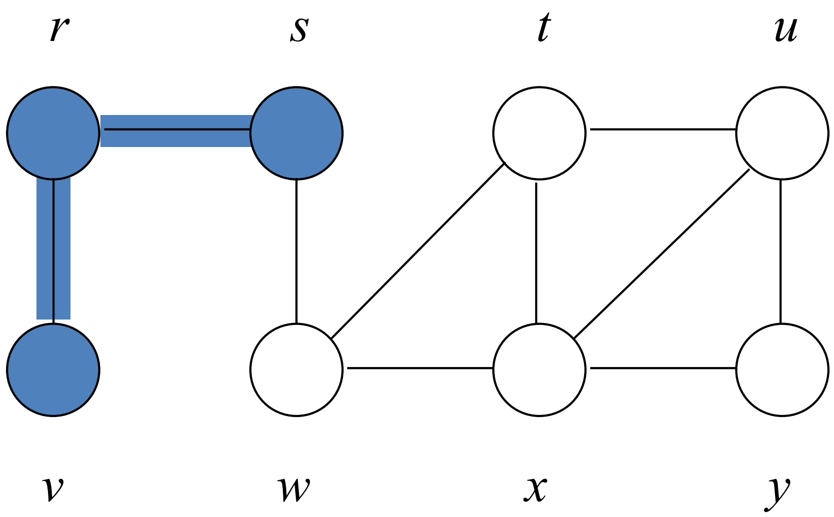
Contents

- *Graphs*
  - *Graphs basics*
  - *Graph representation*
- **Searching a graph**
  - Breadth-first search
  - Depth-first search
- **Applications of depth-first search**
  - Topological sort

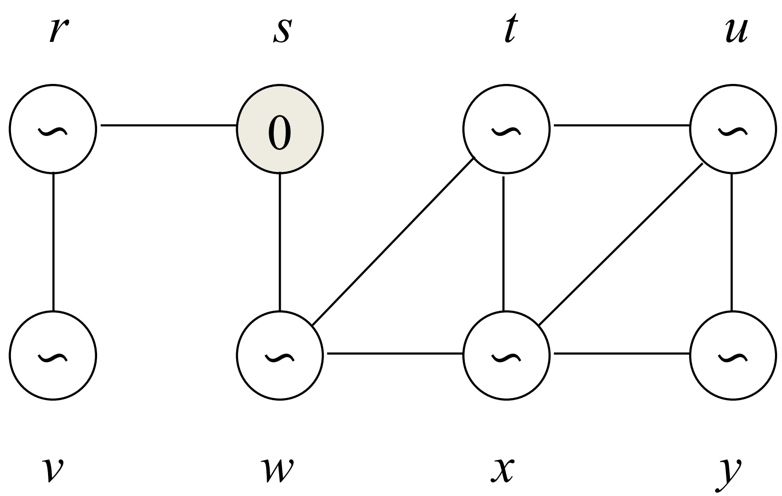


- **Breadth-first search**
- **Depth-first search**

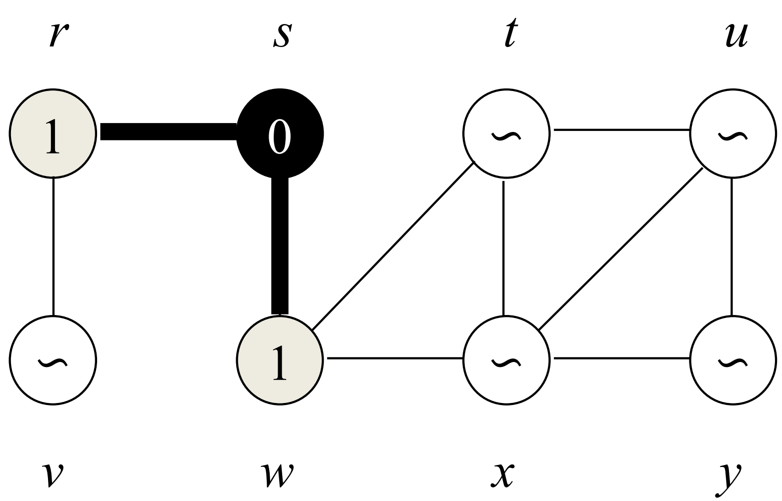
- ***Distance***
  - Distance from u to v
    - The number of edges in the shortest path from u to v.
    - The distance from s to v is 2.



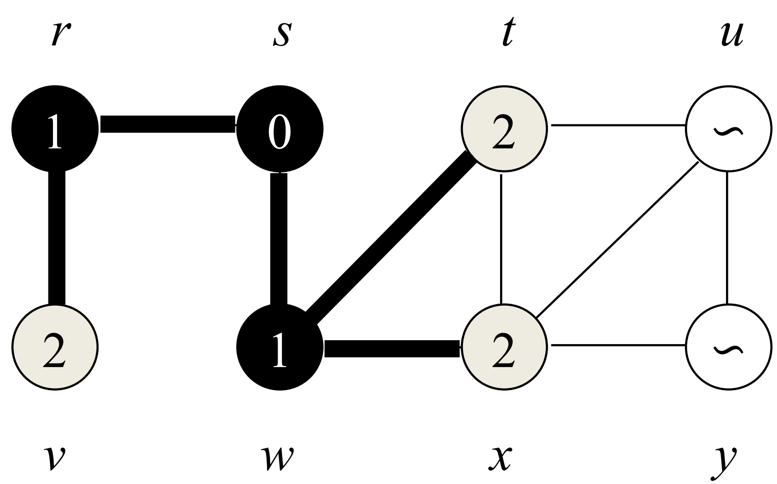
- **Breadth-first search**
  - Given a graph $G = (V, E)$ and a ***source*** vertex s, it explores the edges of G to "discover" every reachable vertex from s.
  - It discovers vertices in the increasing order of distance from the source. It first discovers all vertices at distance 1, then 2, and etc.



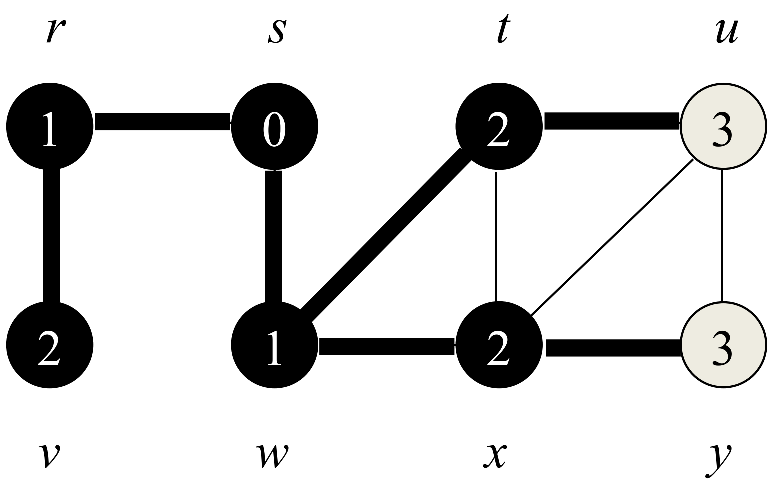
- **Breadth-first search**
  - Given a graph $G = (V, E)$ and a ***source*** vertex s, it explores the edges of G to "discover" every reachable vertex from s.
  - It discovers vertices in the increasing order of distance from the source. It first discovers all vertices at distance 1, then 2, and etc.



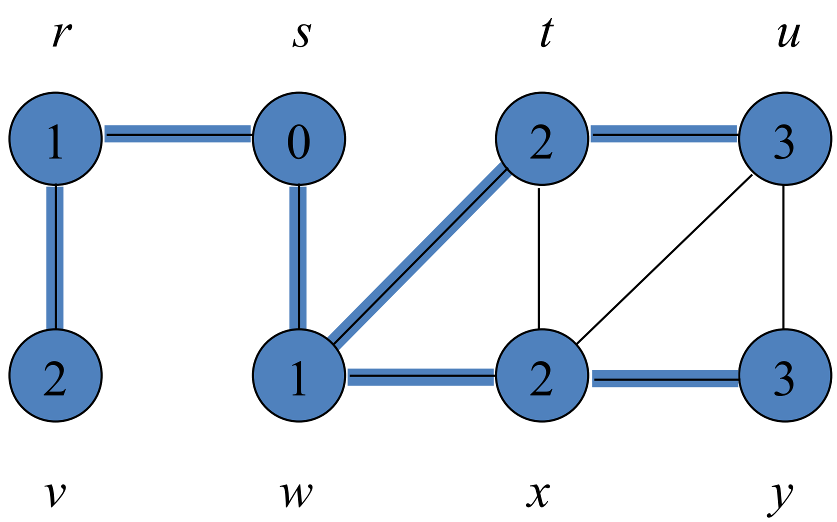
- **Breadth-first search**
  - Given a graph $G = (V, E)$ and a ***source*** vertex s, it explores the edges of G to "discover" every reachable vertex from s.
  - It discovers vertices in the increasing order of distance from the source. It first discovers all vertices at distance 1, then 2, and etc.



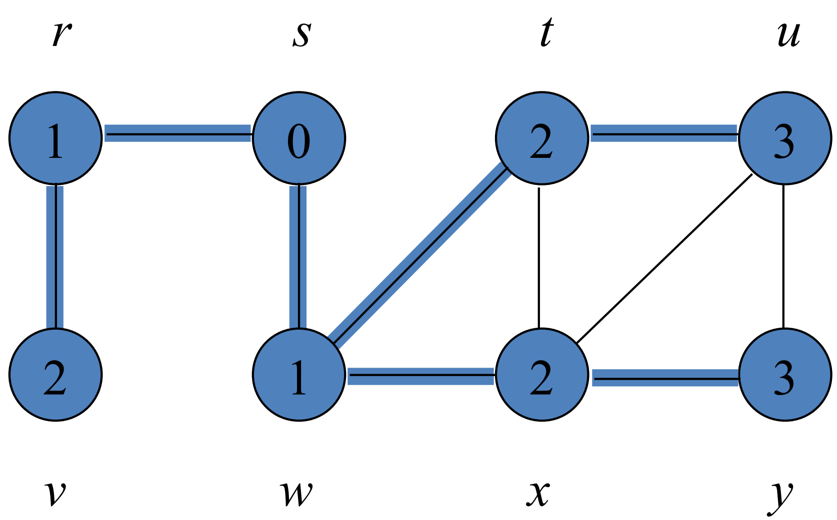
- **Breadth-first search**
  - Given a graph $G = (V, E)$ and a ***source*** vertex s, it explores the edges of G to "discover" every reachable vertex from s.
  - It discovers vertices in the increasing order of distance from the source. It first discovers all vertices at distance 1, then 2, and etc.



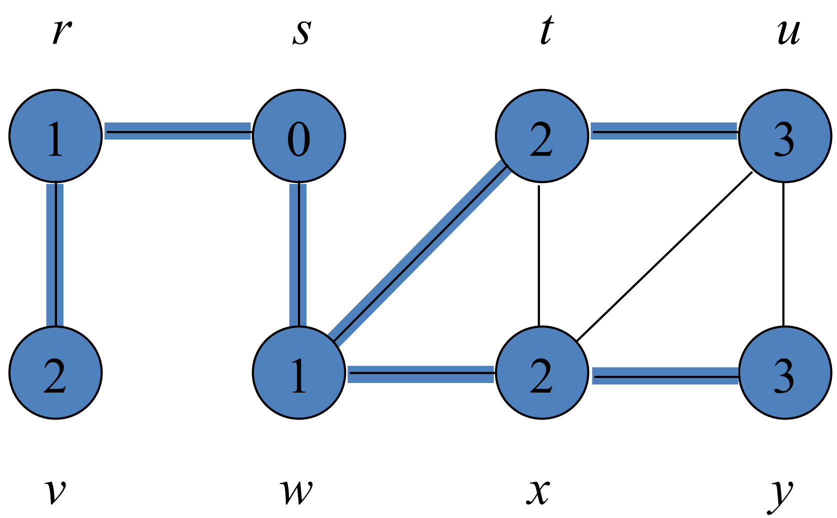
- **Breadth-first search**
  - It also computes
    - the distance of vertices from the source: $u.d = 3$
    - the predecessor of vertices: $u.\pi = t$



- The ***predecessor subgraph*** of G as $G_\pi = (V_\pi, E_\pi)$,
  - $V_\pi = \{v \in V \mid v.\pi \neq \text{NIL}\} \cup \{ s\}$
  - $E_\pi = \{(v.\pi, v) \mid v \in V_\pi - \{s\}\}$.

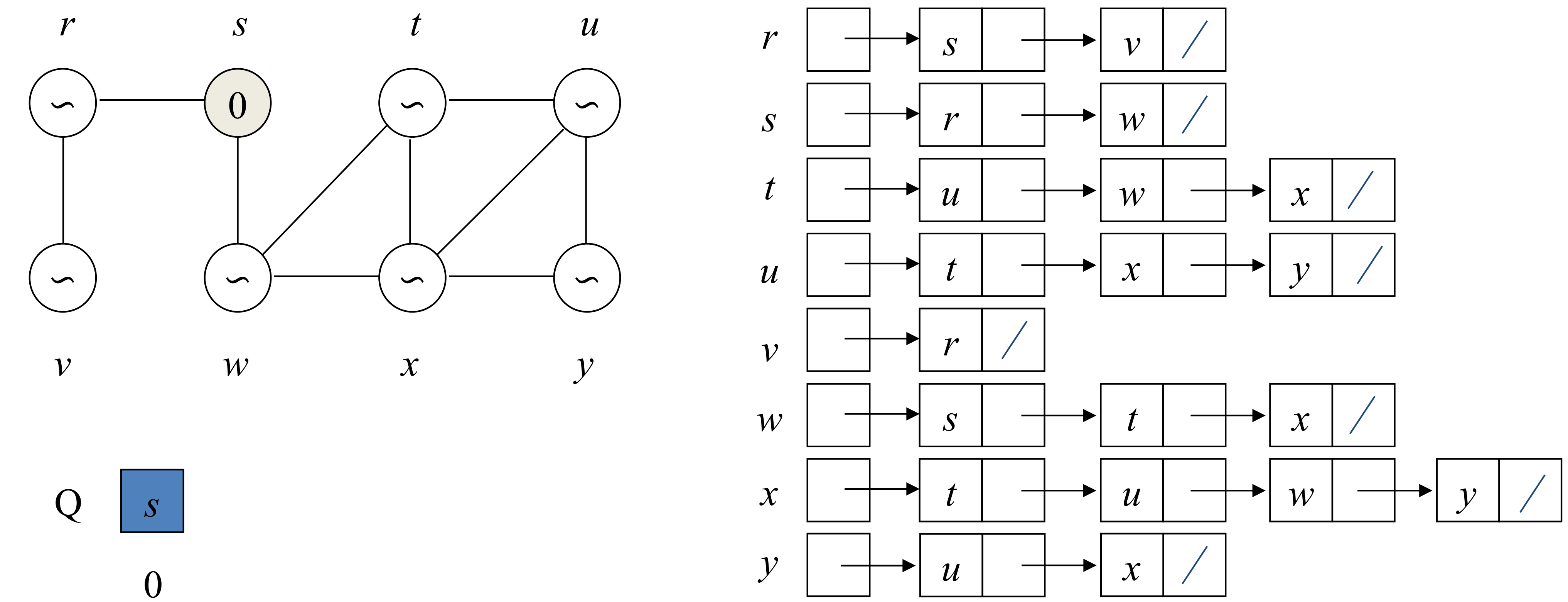


- The predecessor subgraph G_π is a ***breadth-first tree.***
  - since it is connected and $|E_\pi| = |V_\pi|$ -1.
  - The edges in E_π are called ***tree edges***.

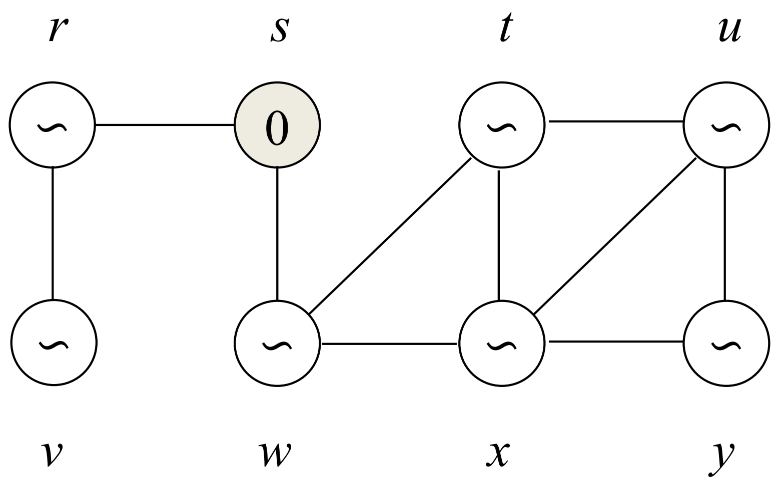


Breadth-first search

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

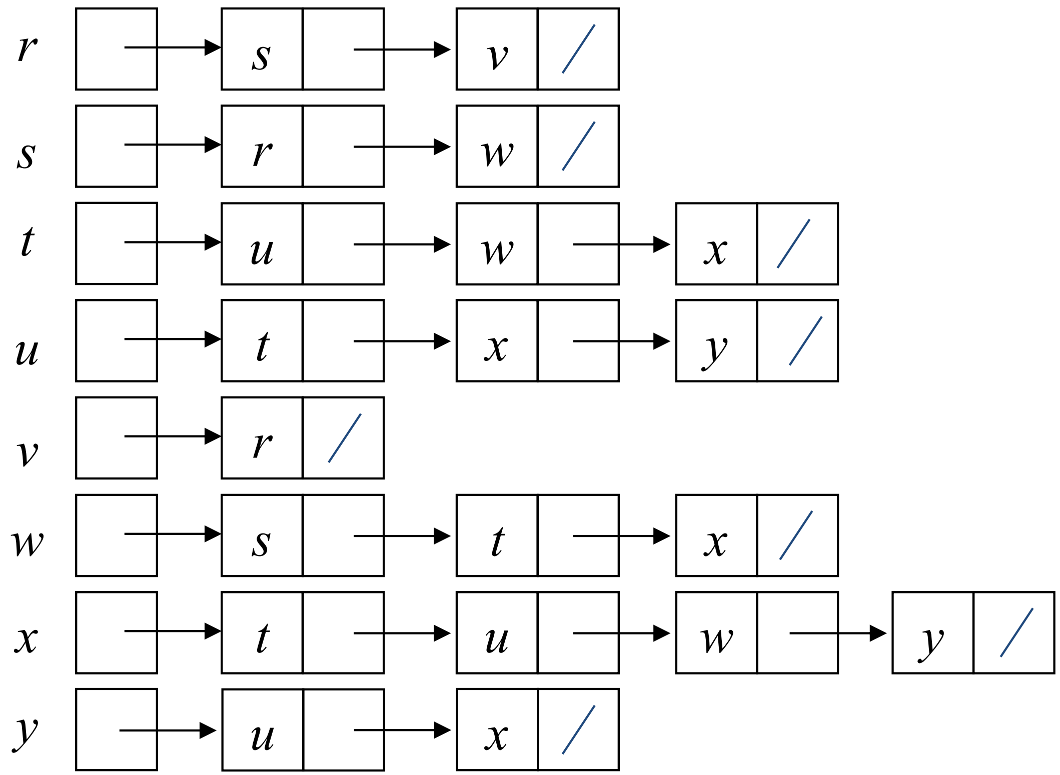


- white: not discovered (not entered the Q)
- gray: discovered (in the Q)
- black: finished (out of the Q)

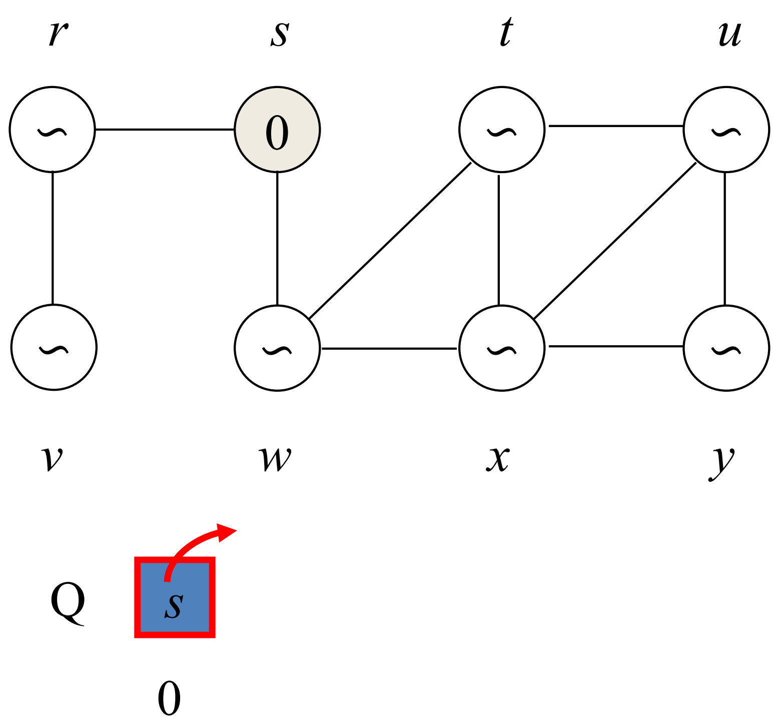


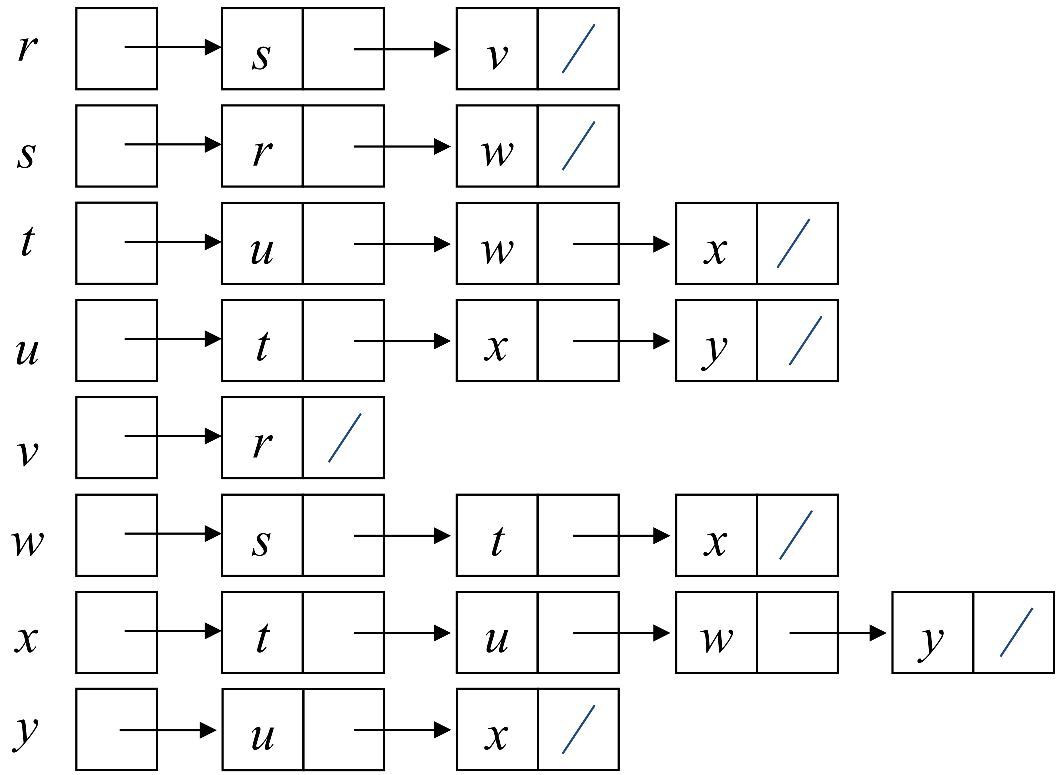
Q s

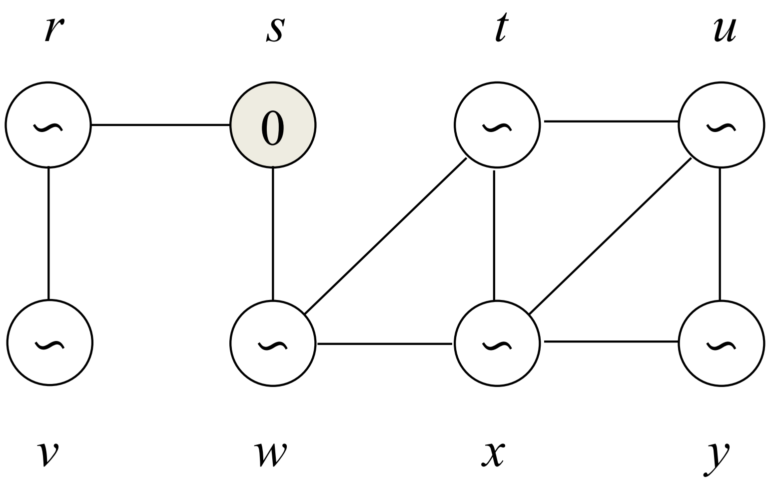
0



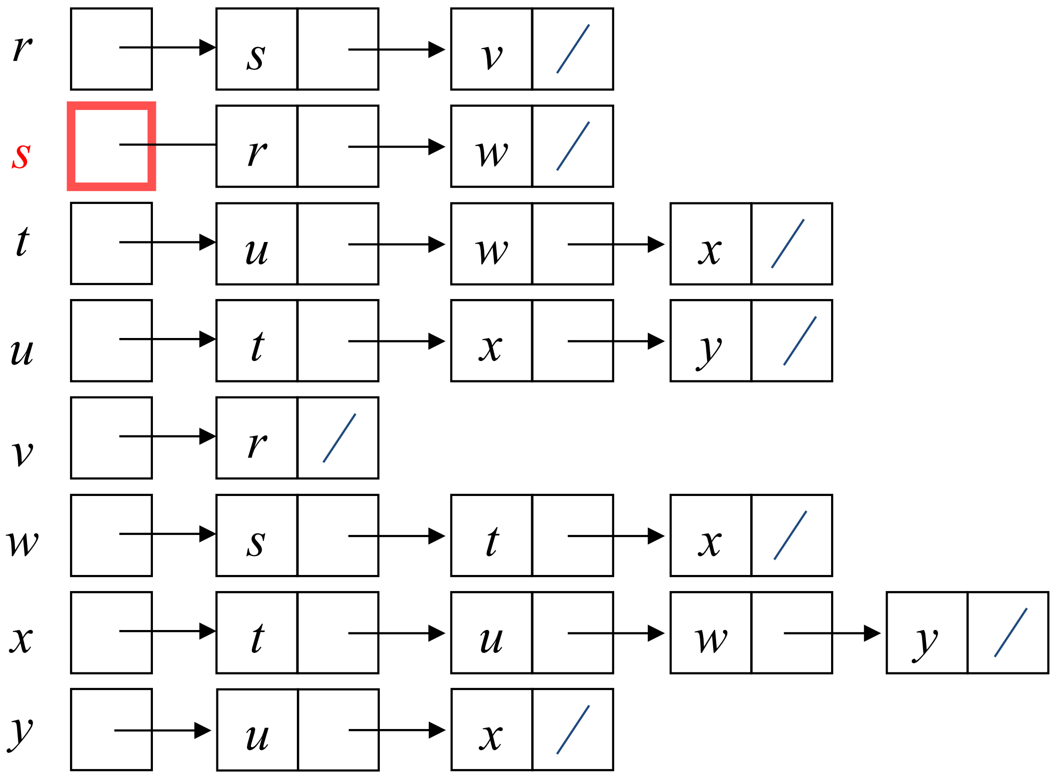
Breadth-first search

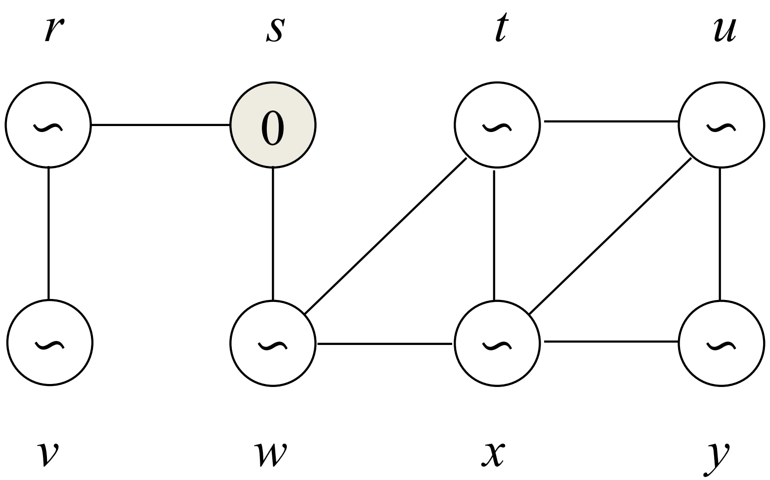




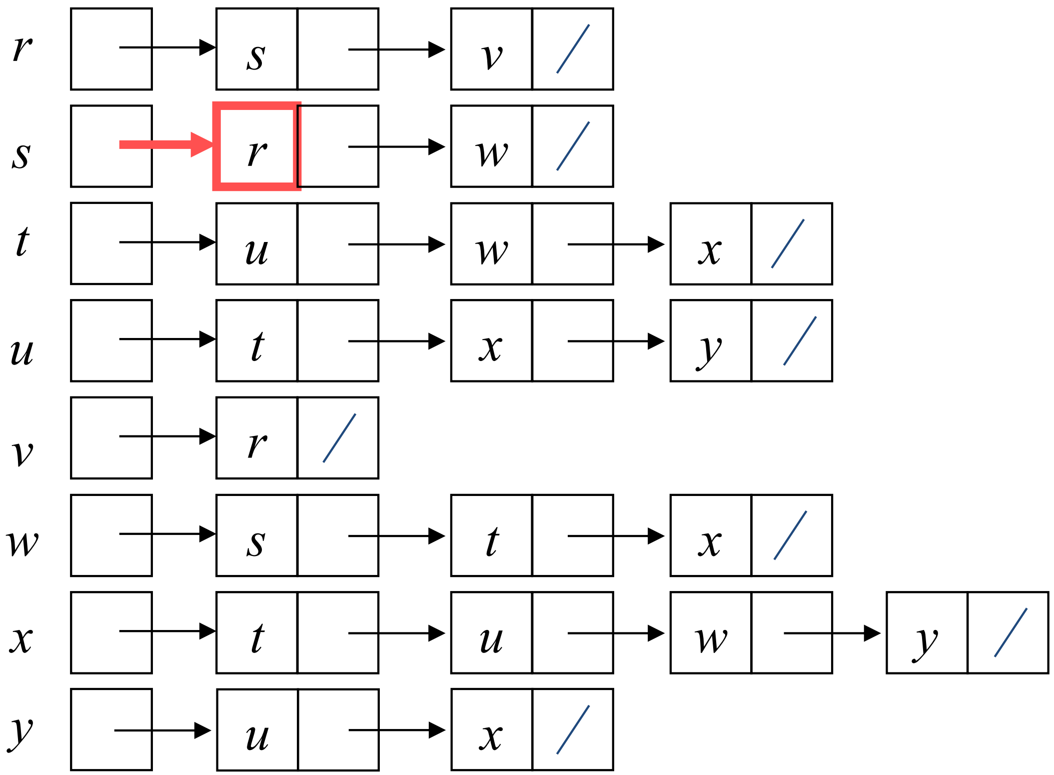


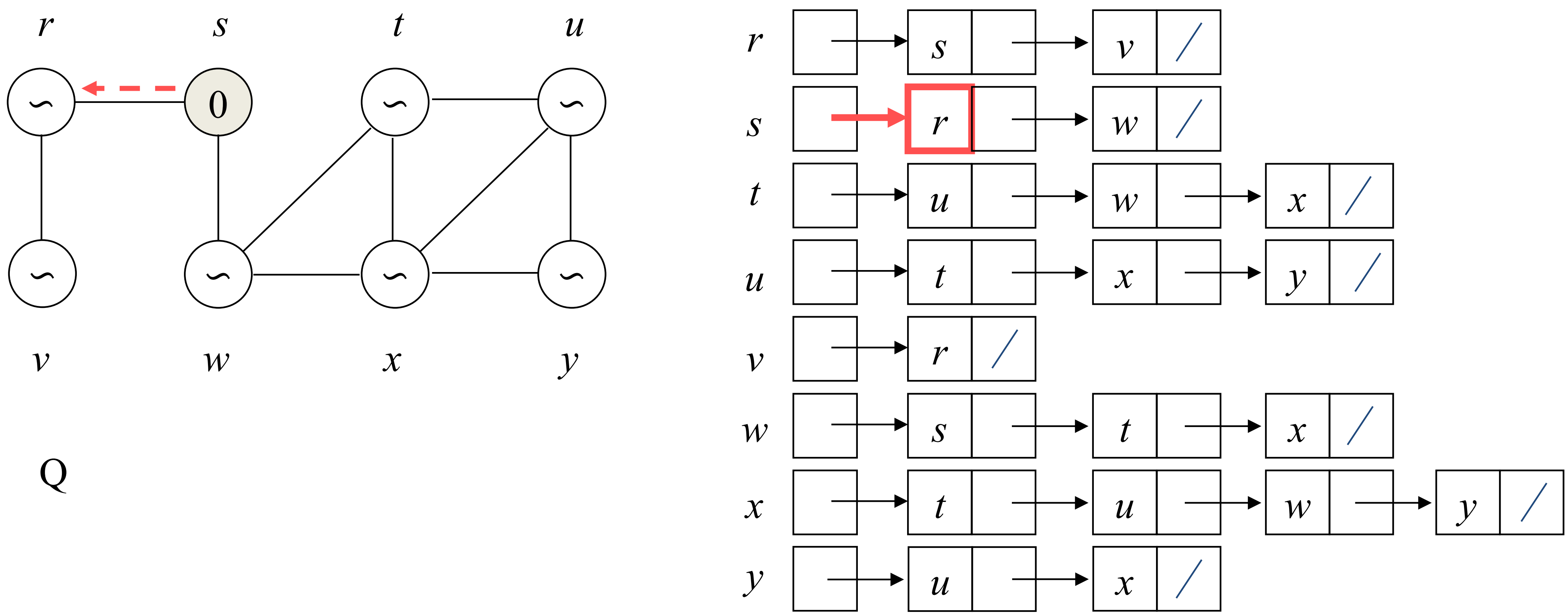
Q



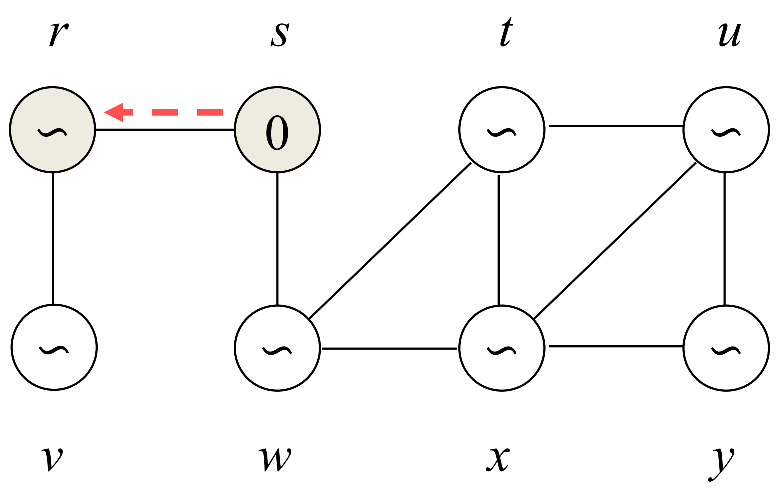


Q

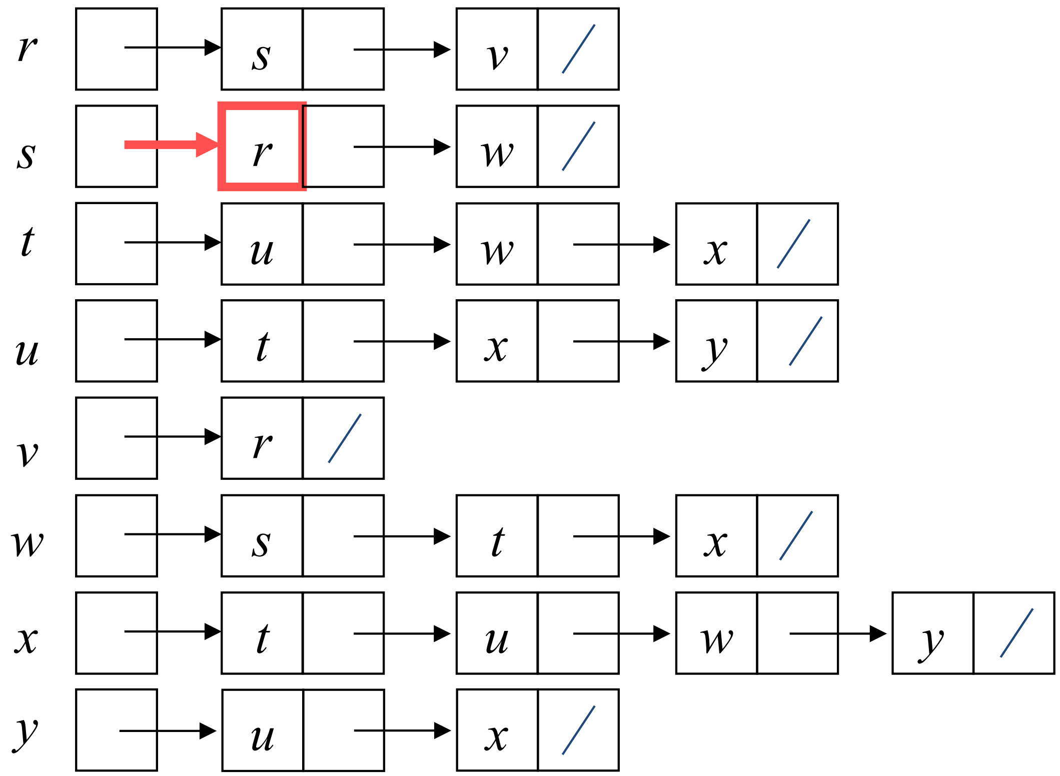


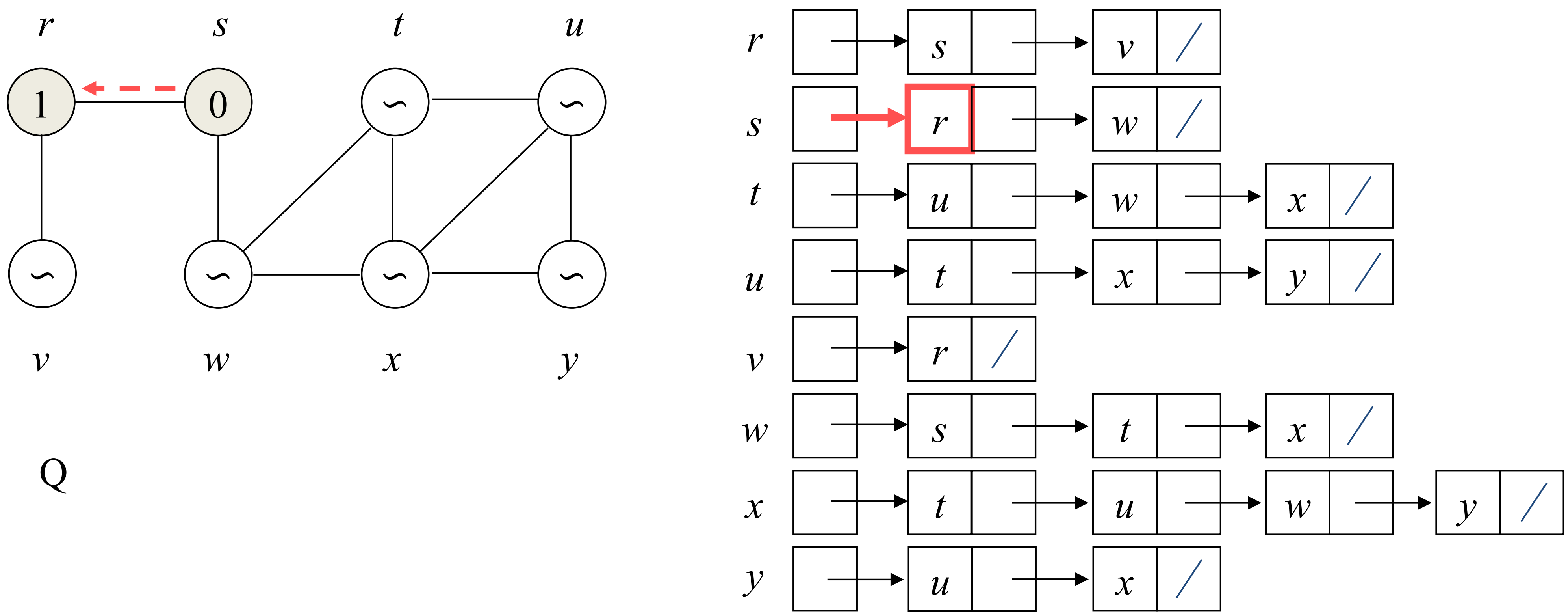


Q

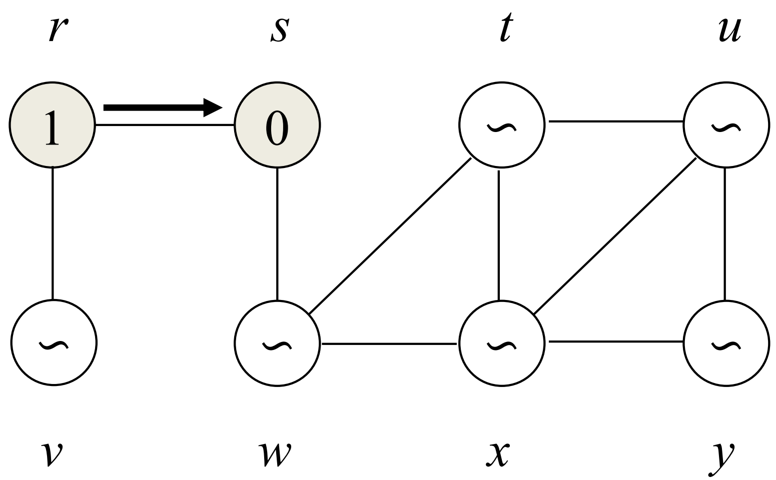


Q

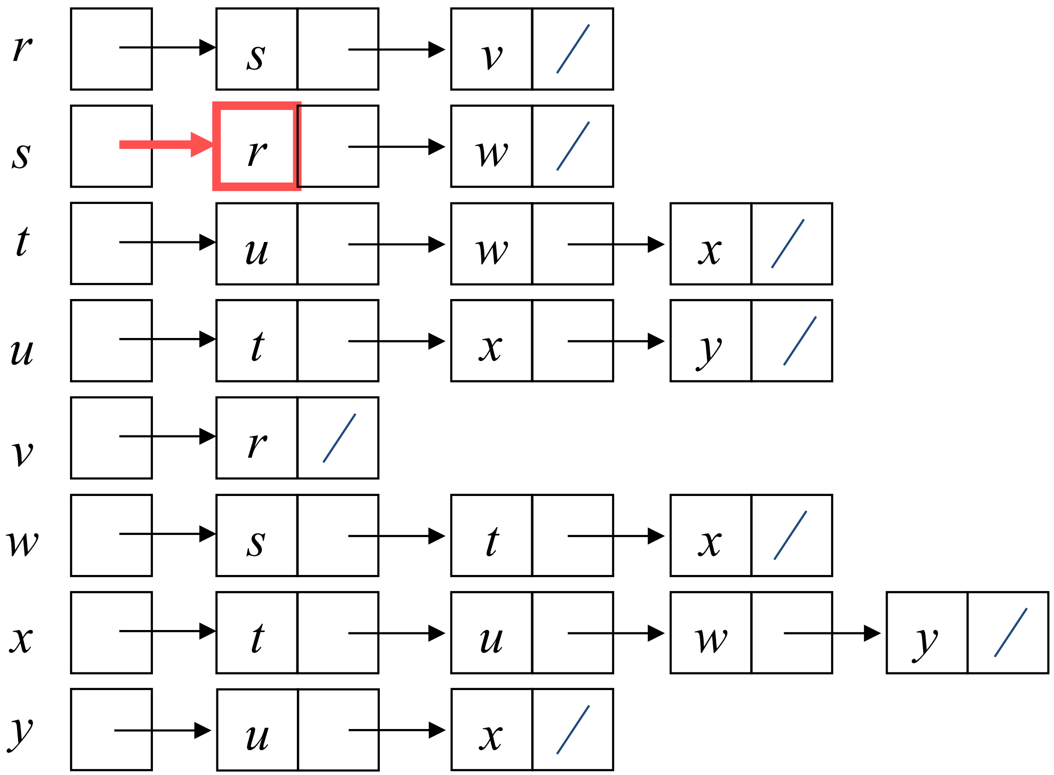




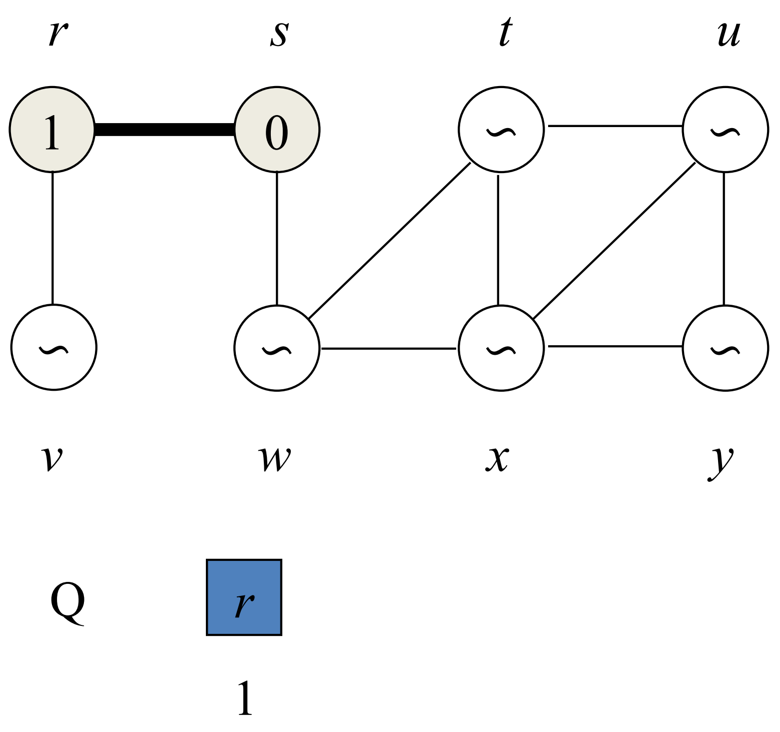
Q

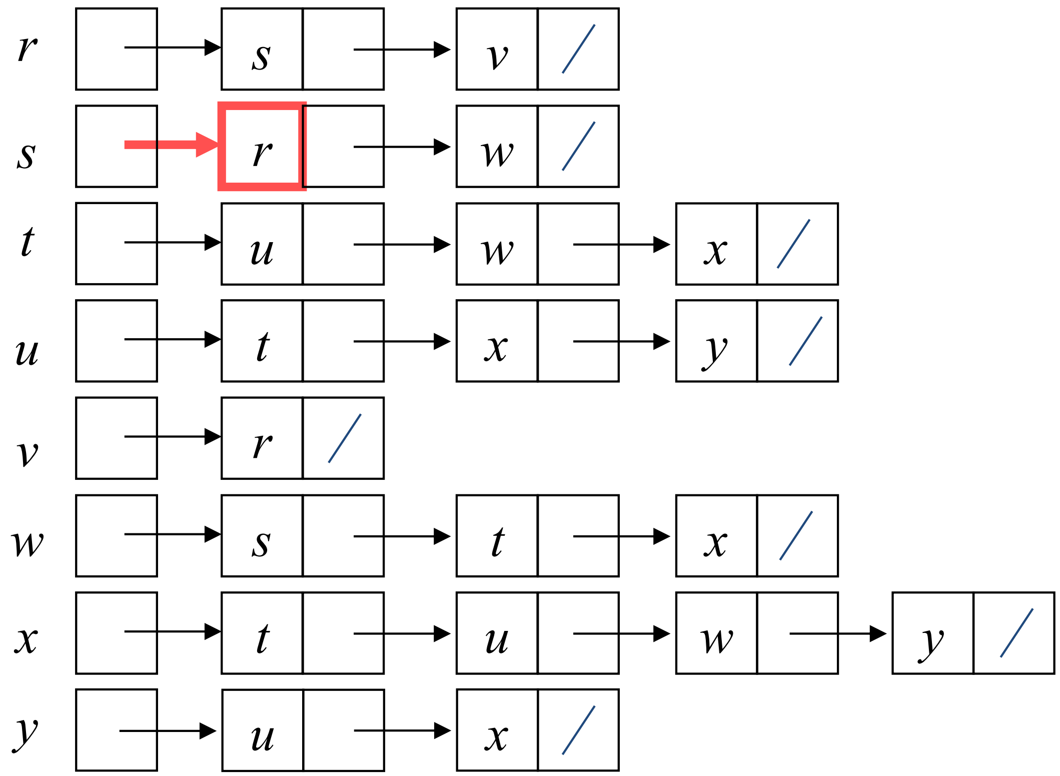


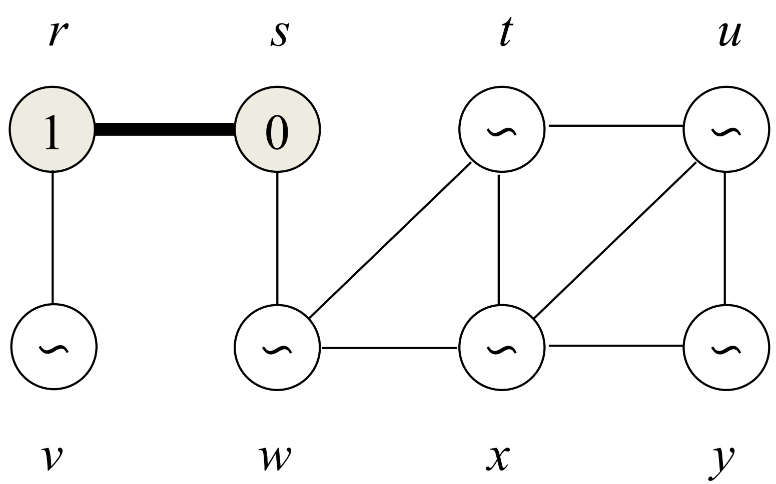
Q

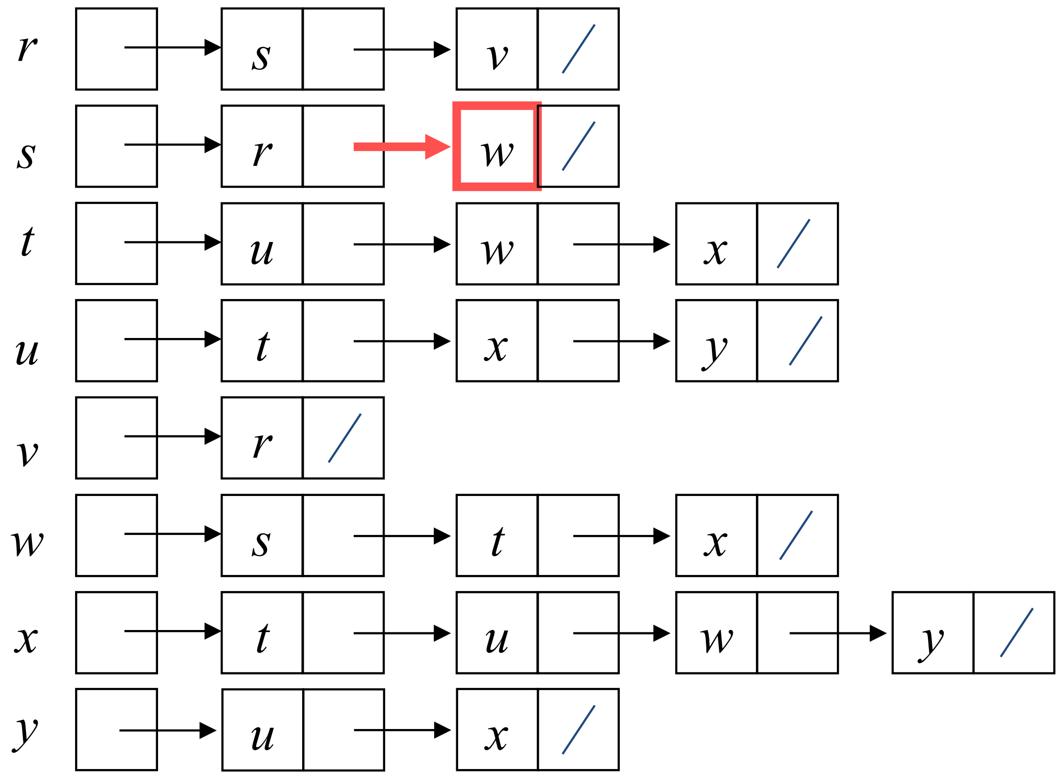


Breadth-first search



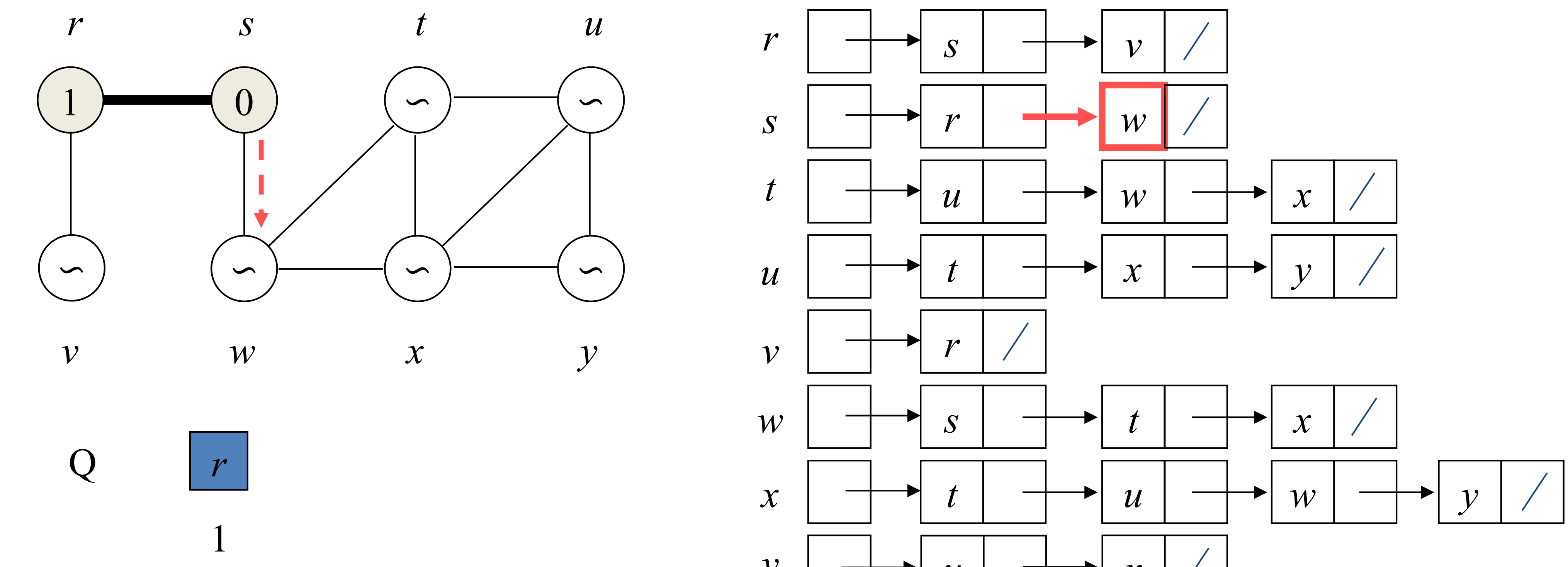


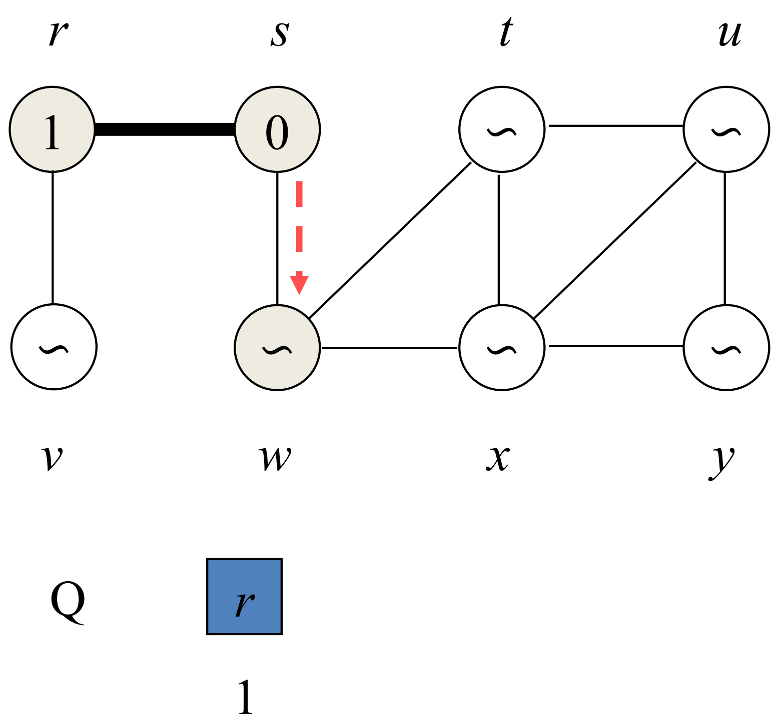


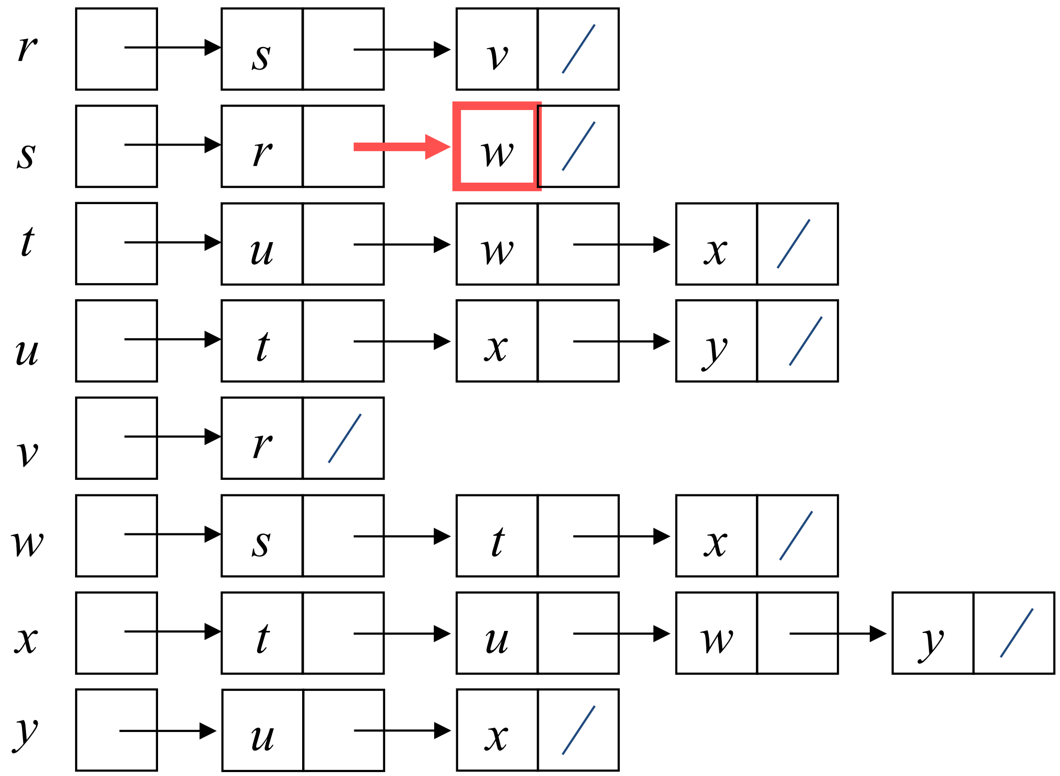


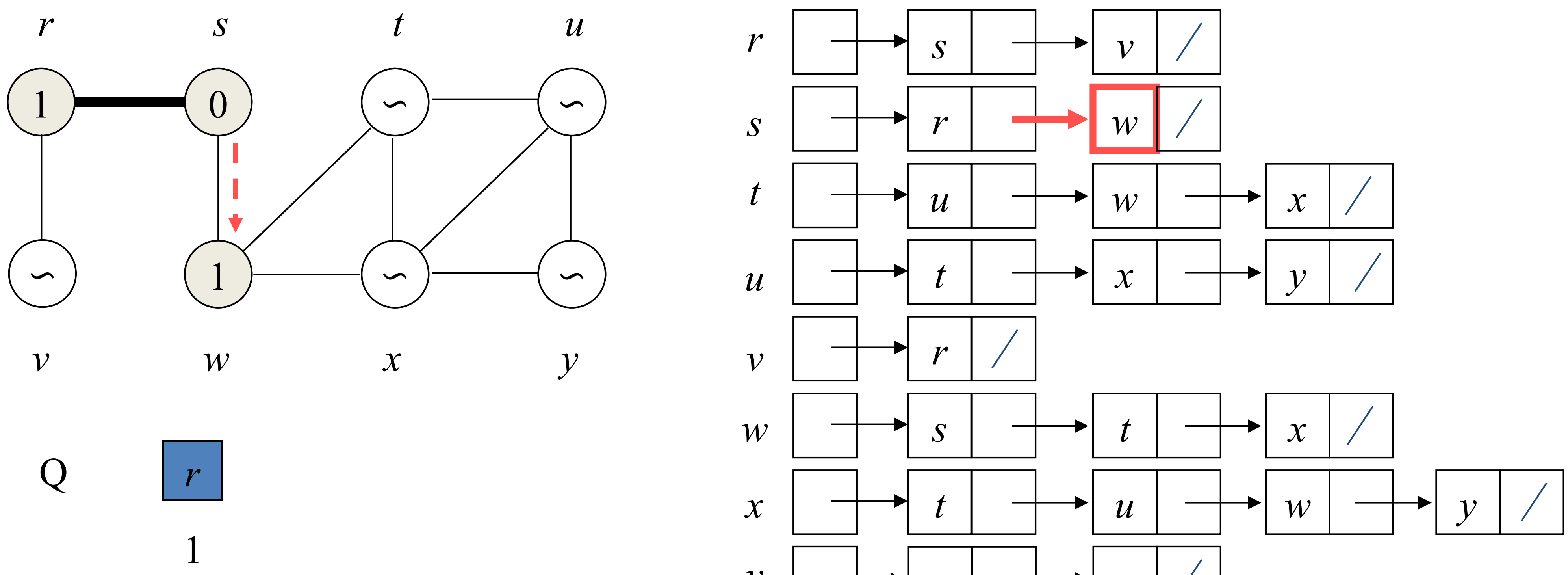
Q r

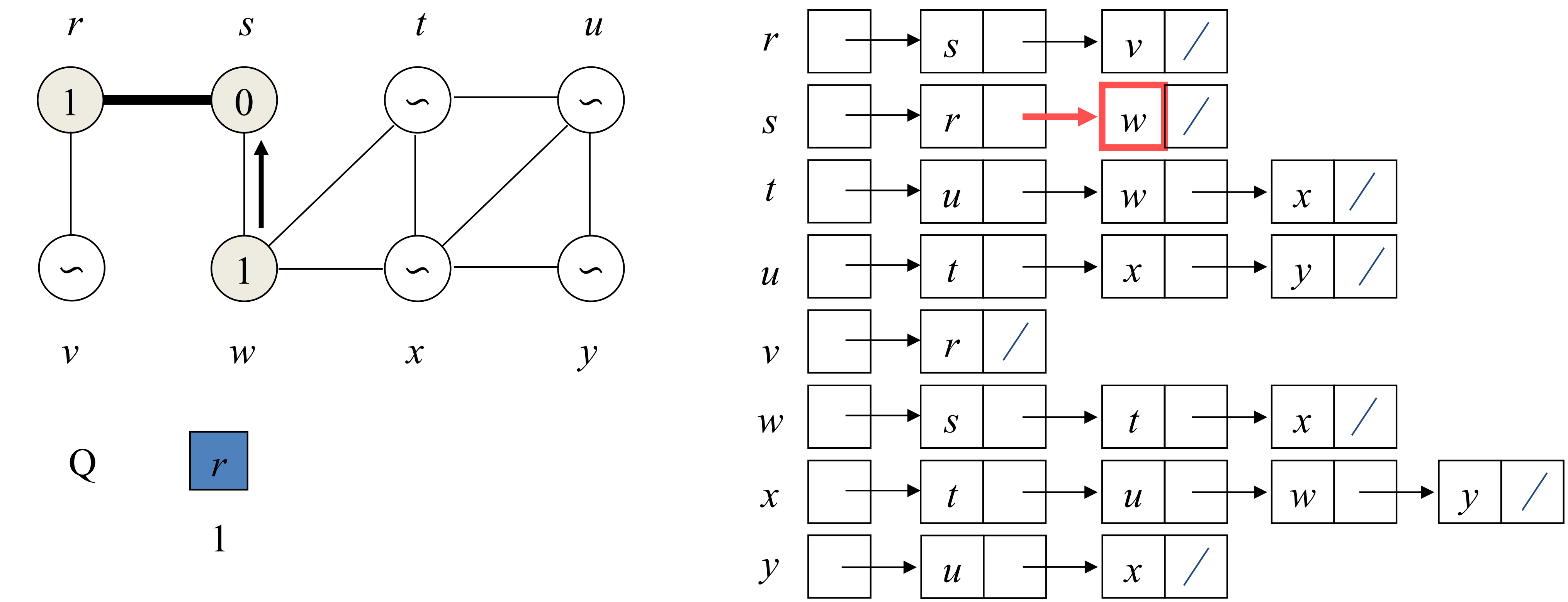
1

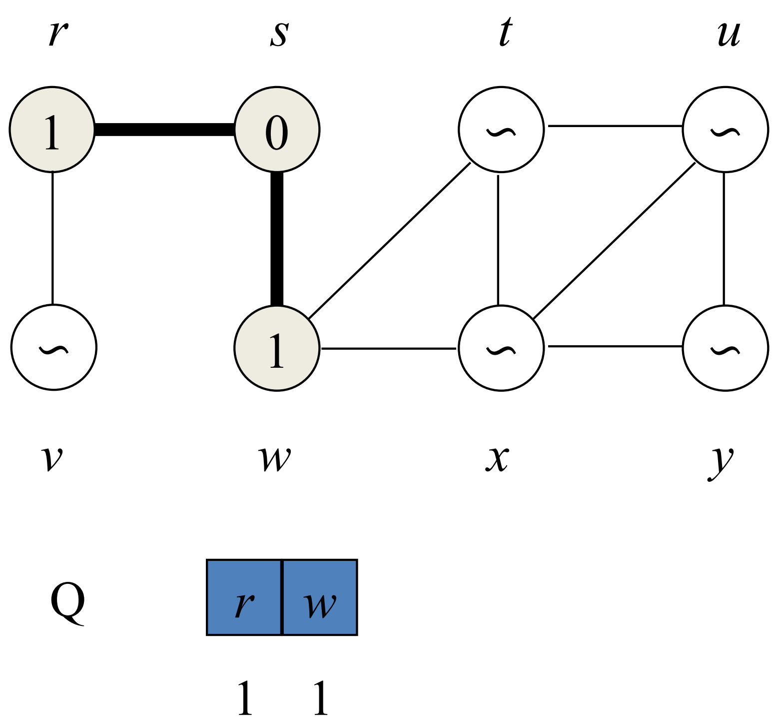

r
s
t
u
1
0
v
w
x
y
Q
r
1
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

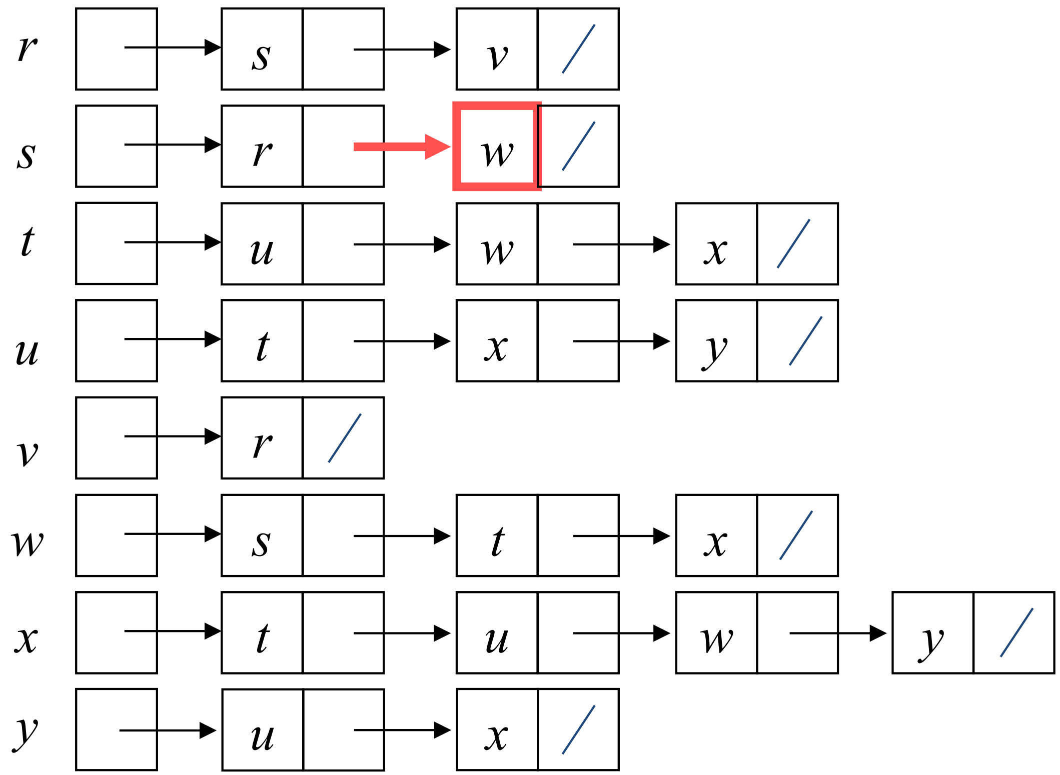

r
s
t
u
1
0
v
w
x
y
Q
r
1

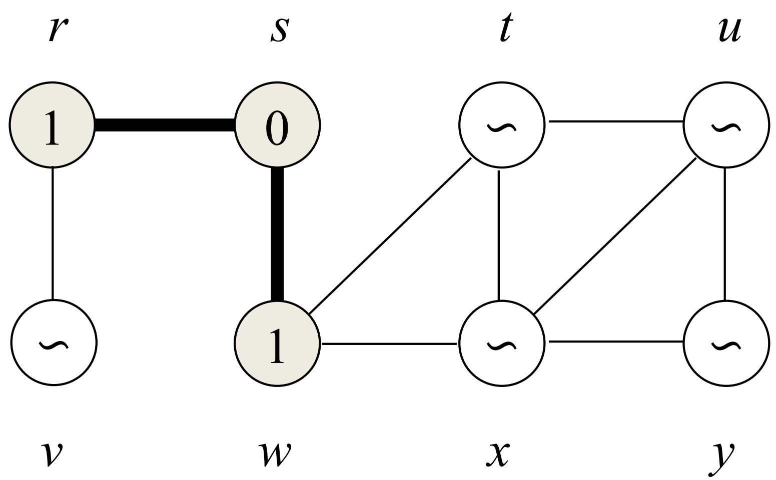

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

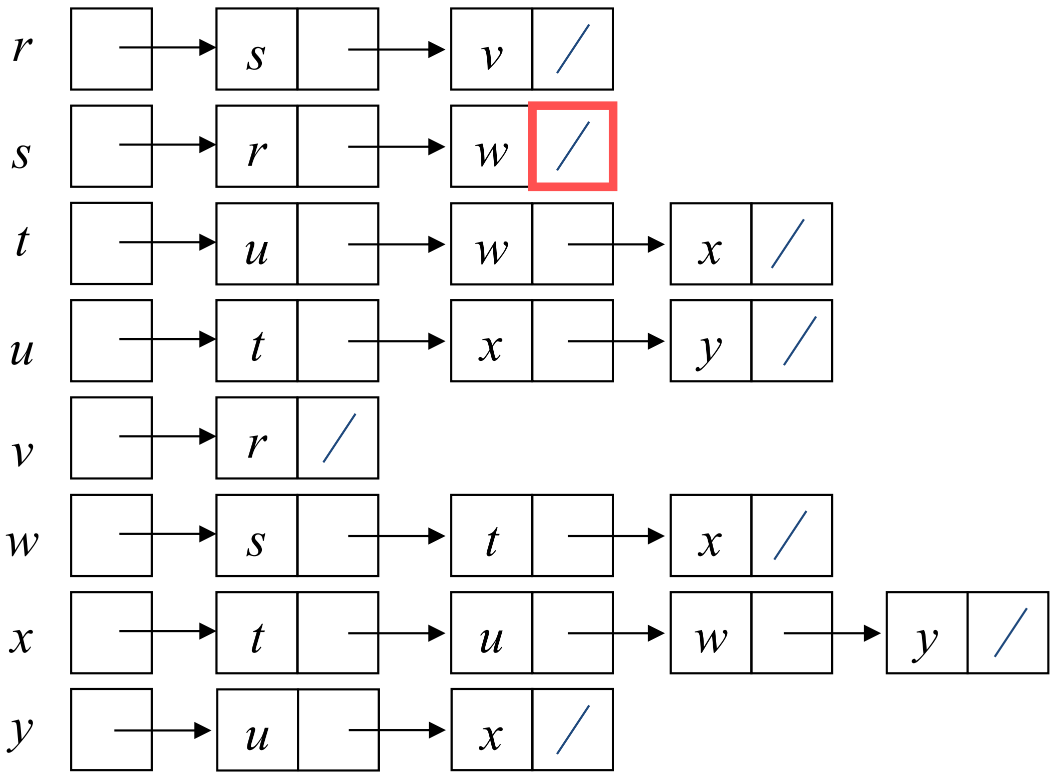

r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y
Q
r
1
r
s
t
u
v
w
x
y
s
v
r
w
u
w
x
t
x
y
r
s
t
x
t
u
w
y
u
x


r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y
Q
r
1
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

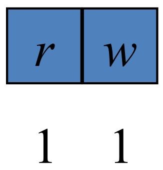

r
s
t
u
1
0
~
~
~
1
~
~
v
w
x
y
Q
r
w
1
1

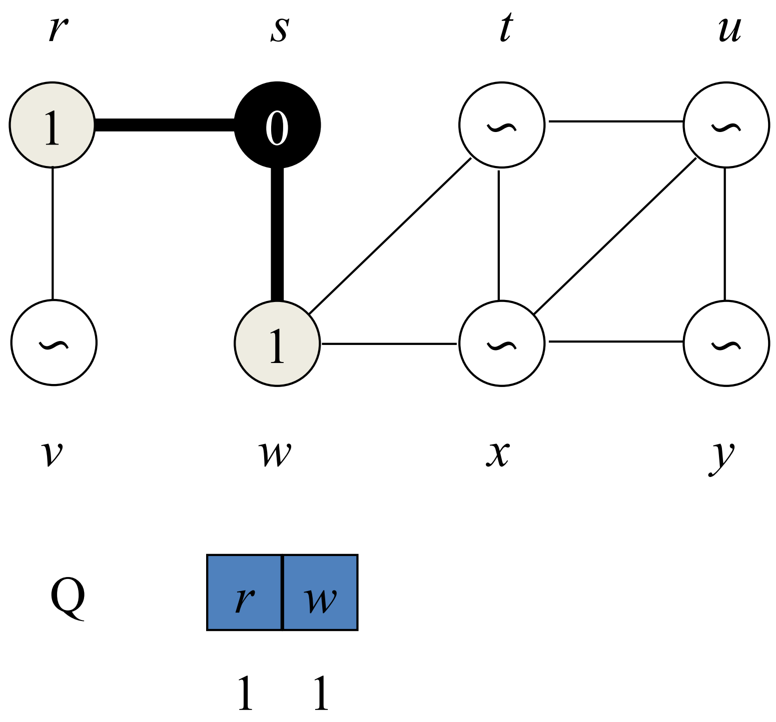

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

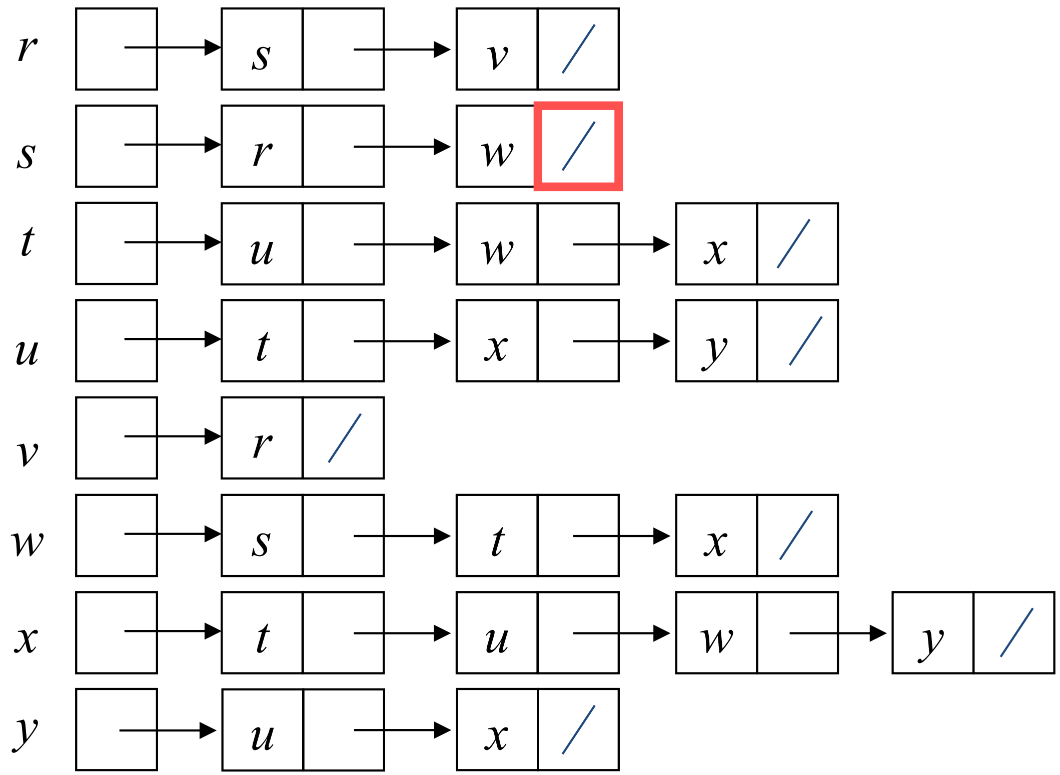


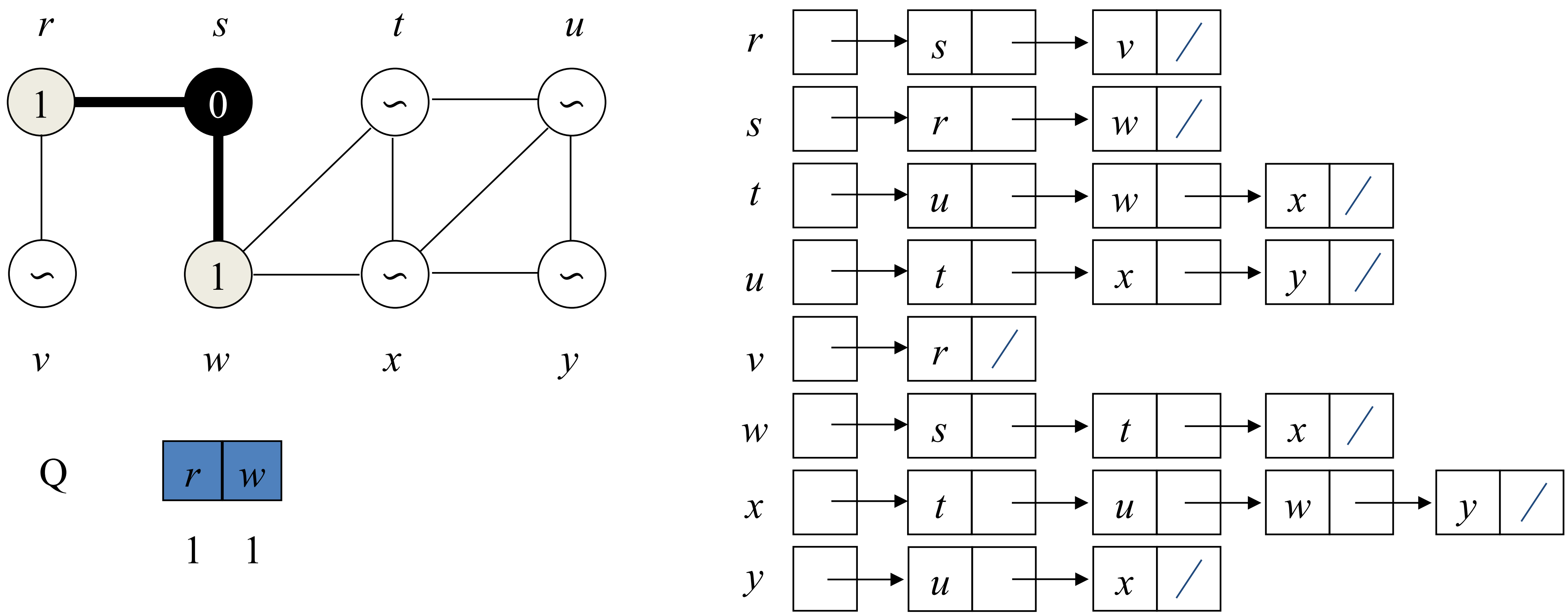


Q

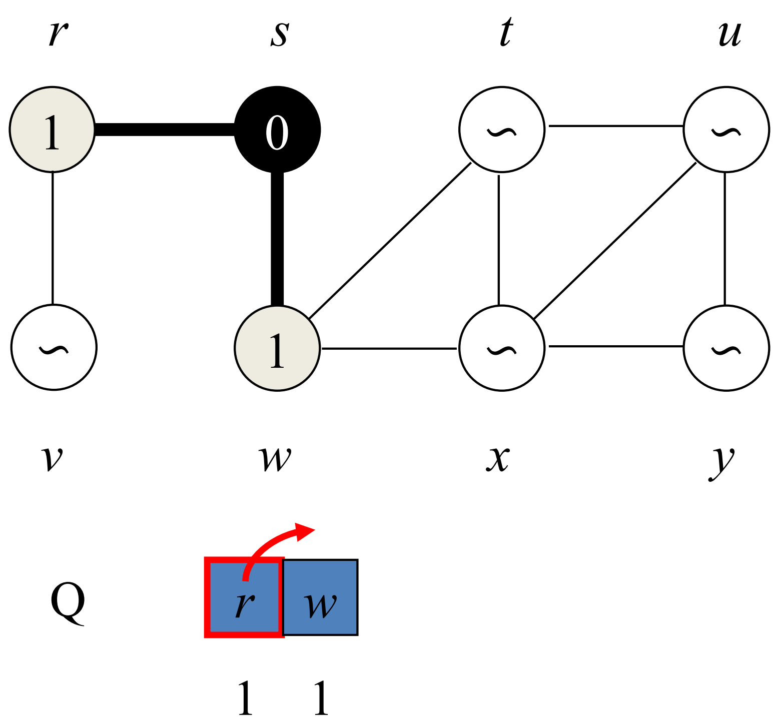


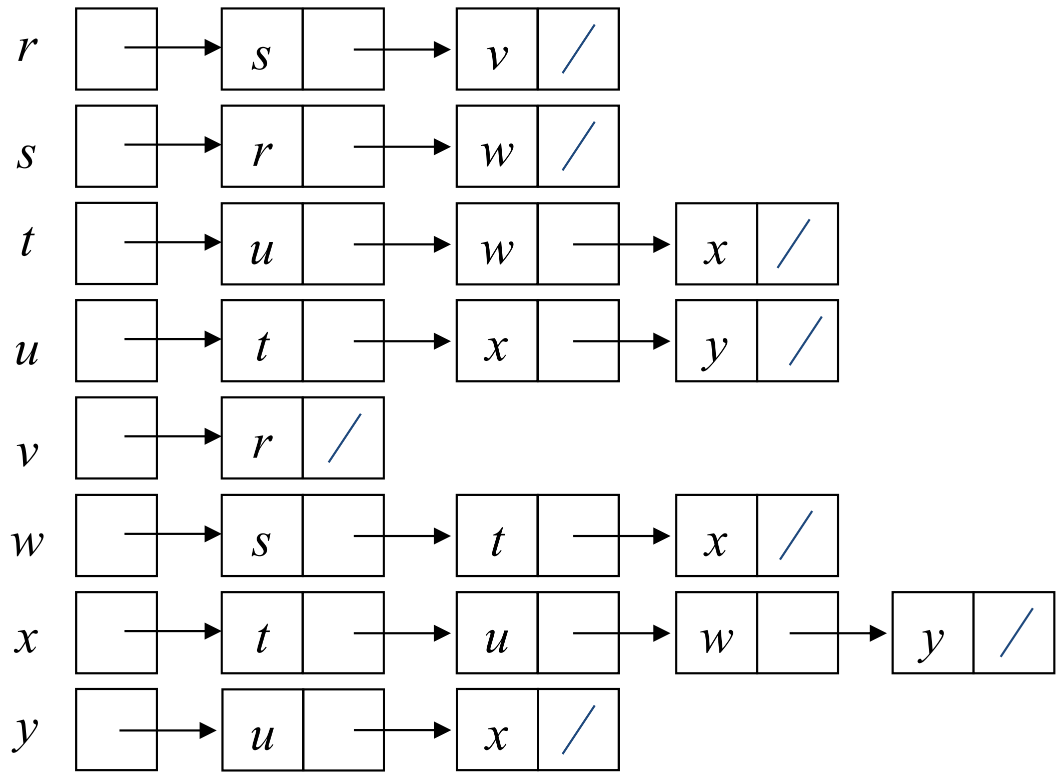

r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y
Q
r
w
1
1

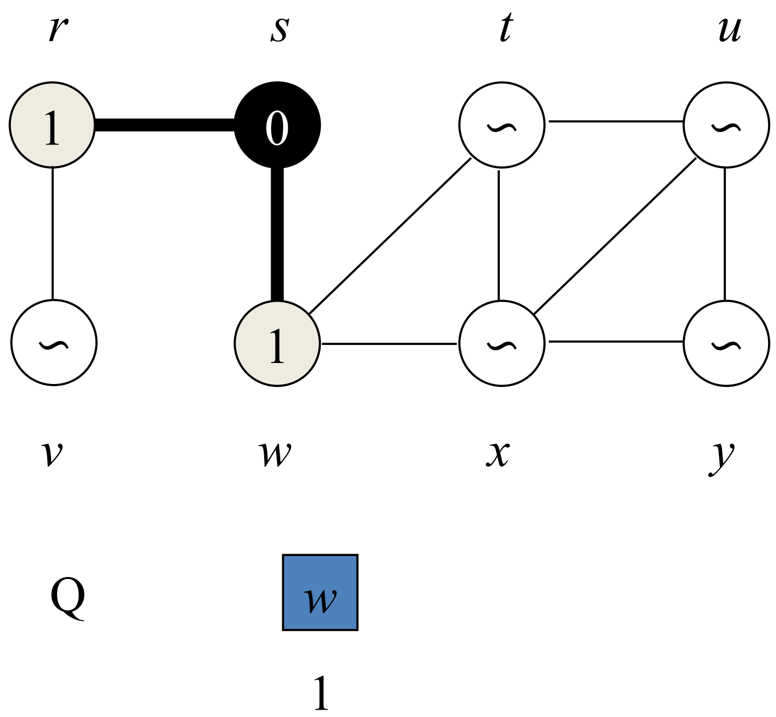

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

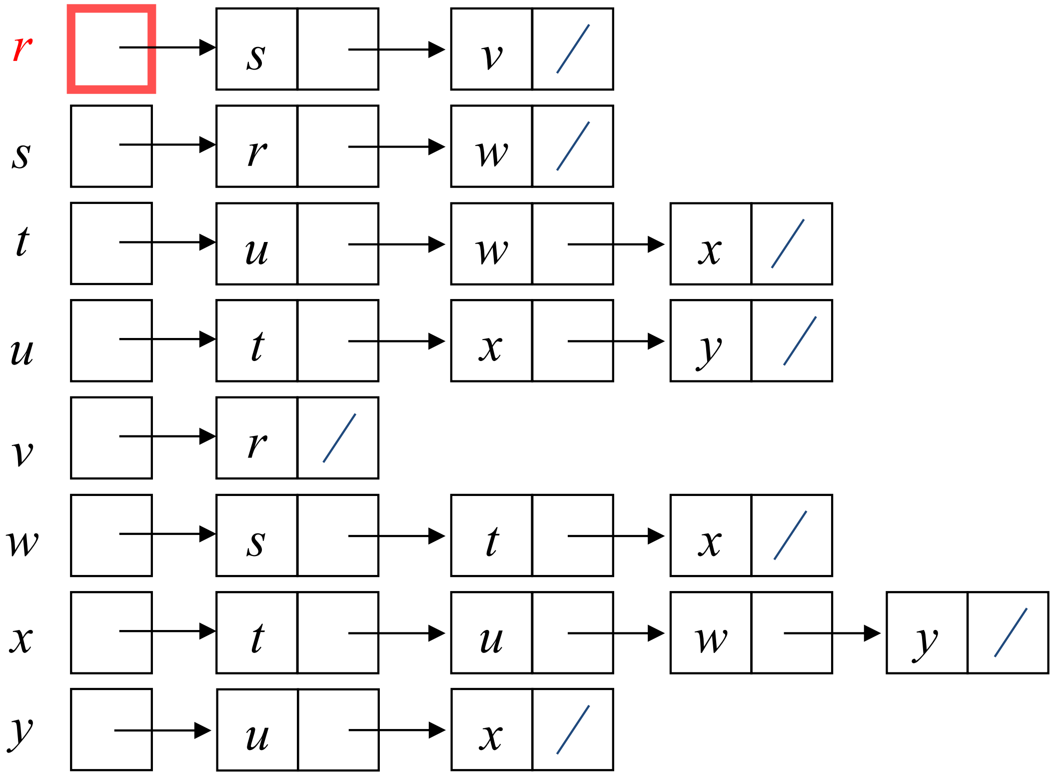

r
s
t
u
1
0
v
w
x
y
1
Q
r
w
1 1
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

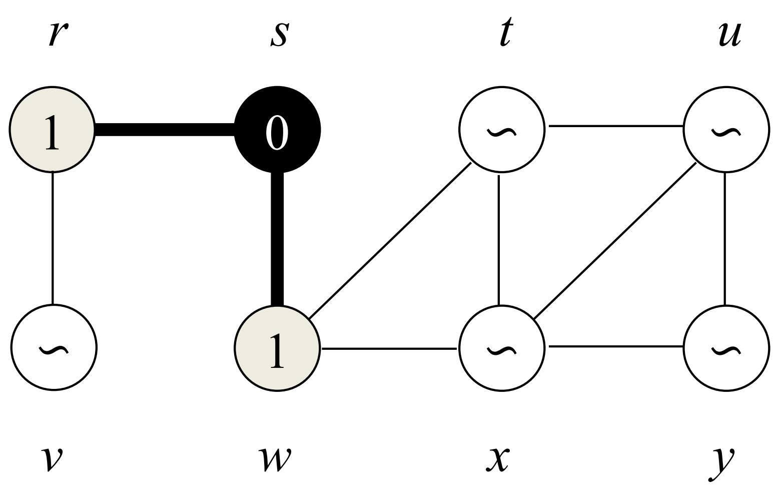
Breadth-first search

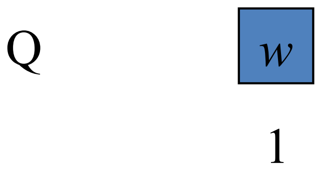


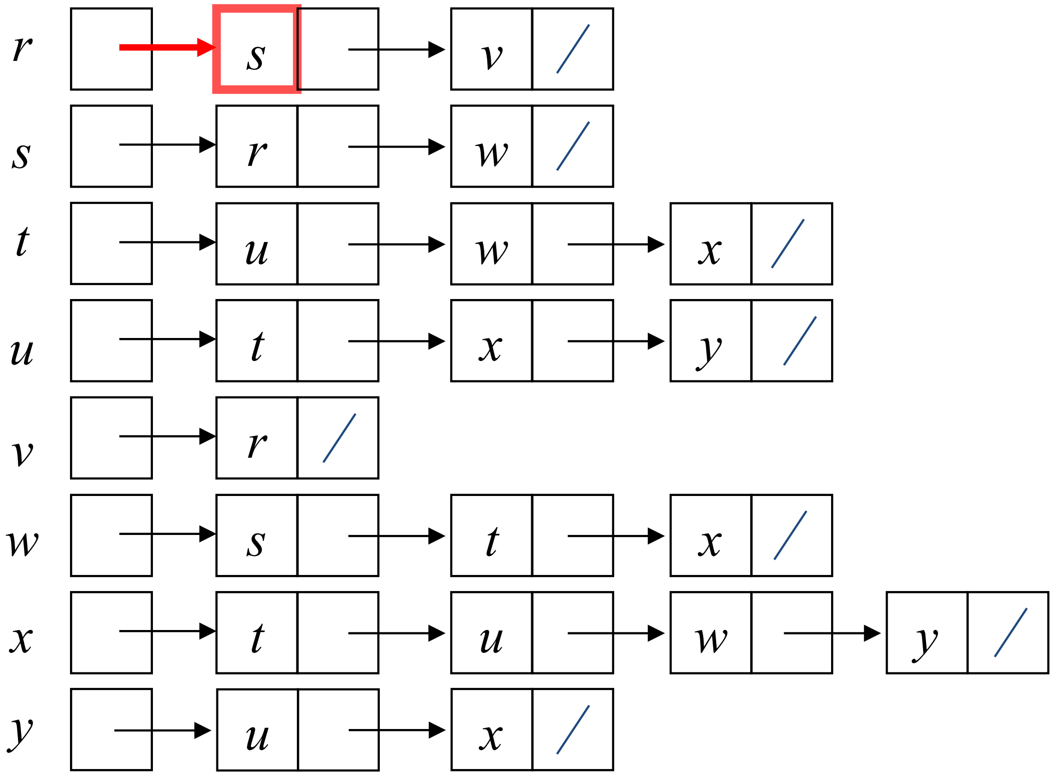


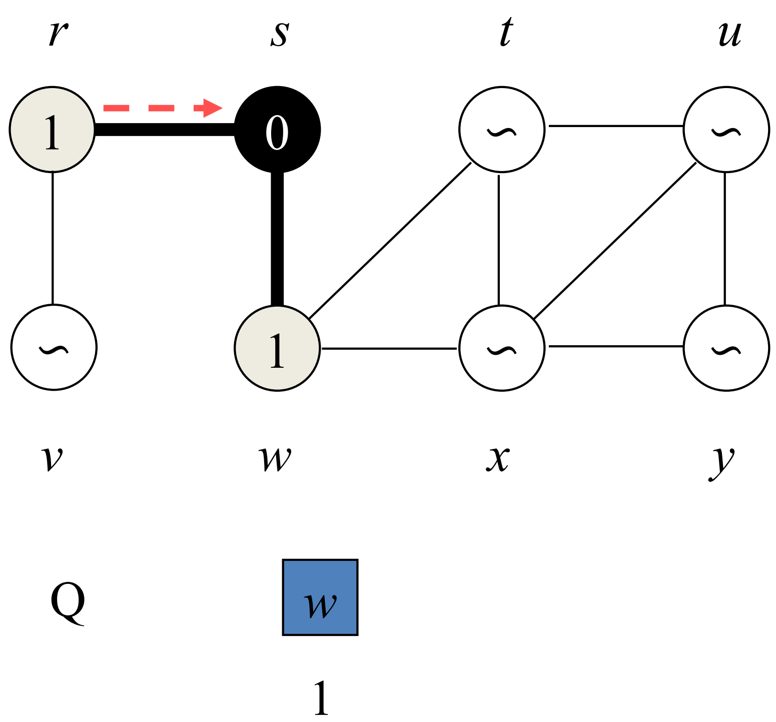

r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y
Q
w
1

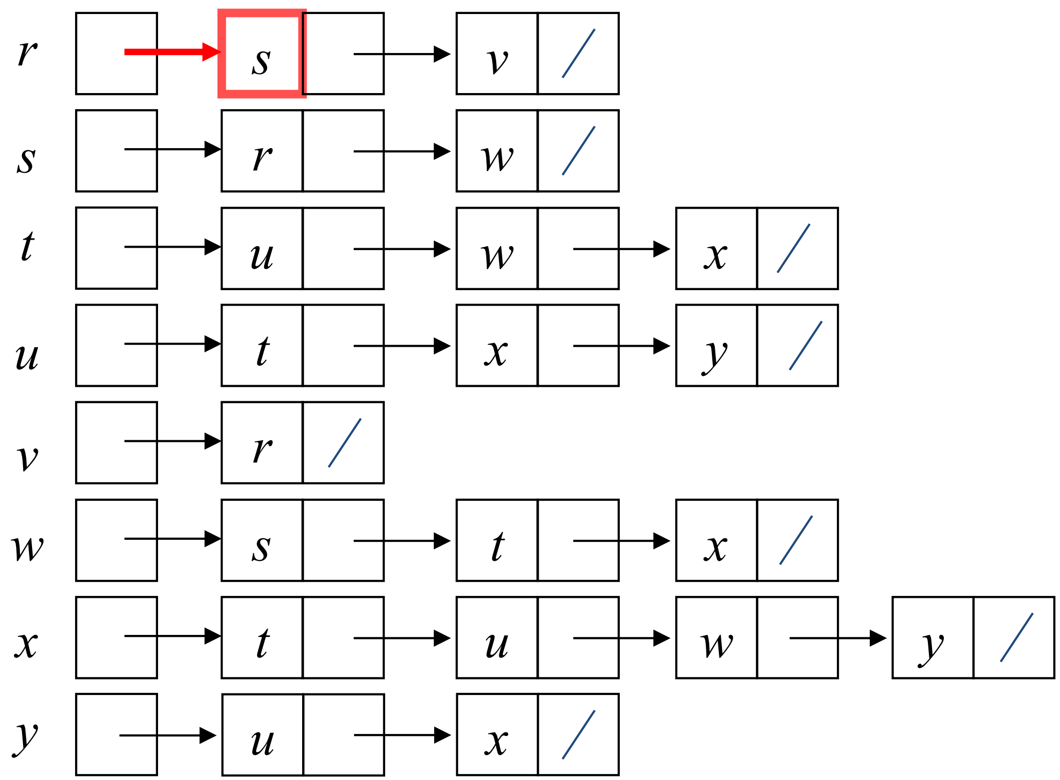

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y

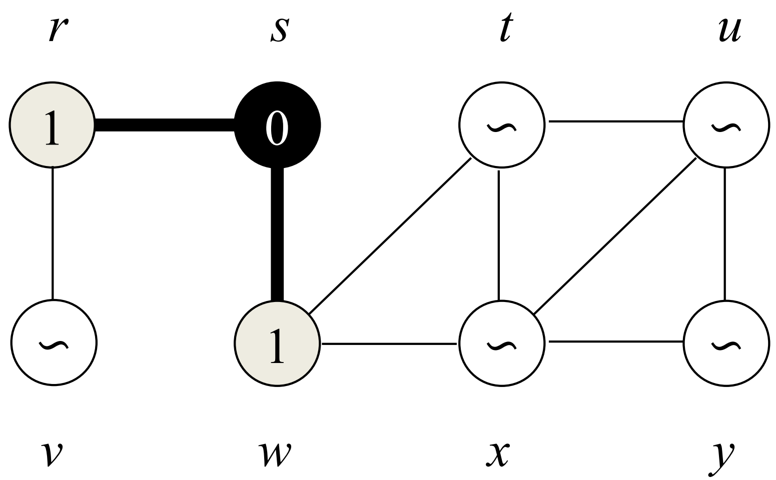

Q
w
1

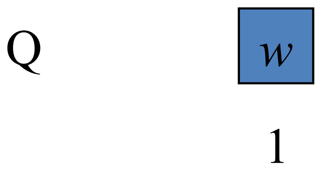

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

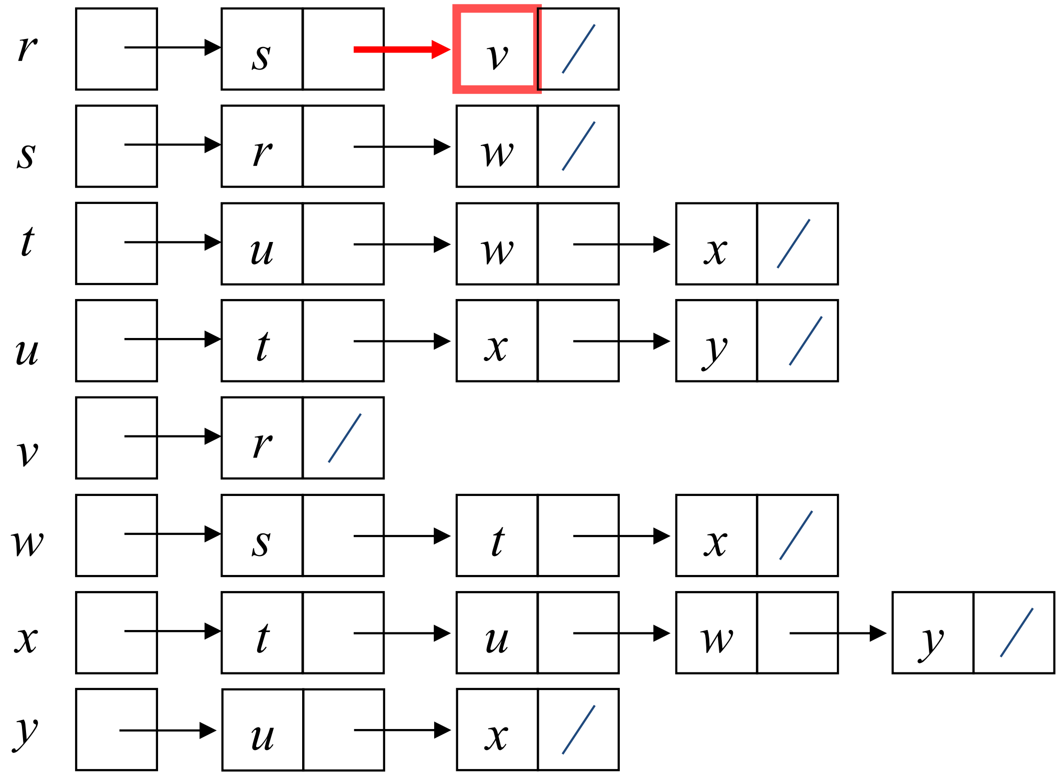

r
s
t
u
1
0
~
~
~
1
~
~
v
w
x
y
Q
w
1

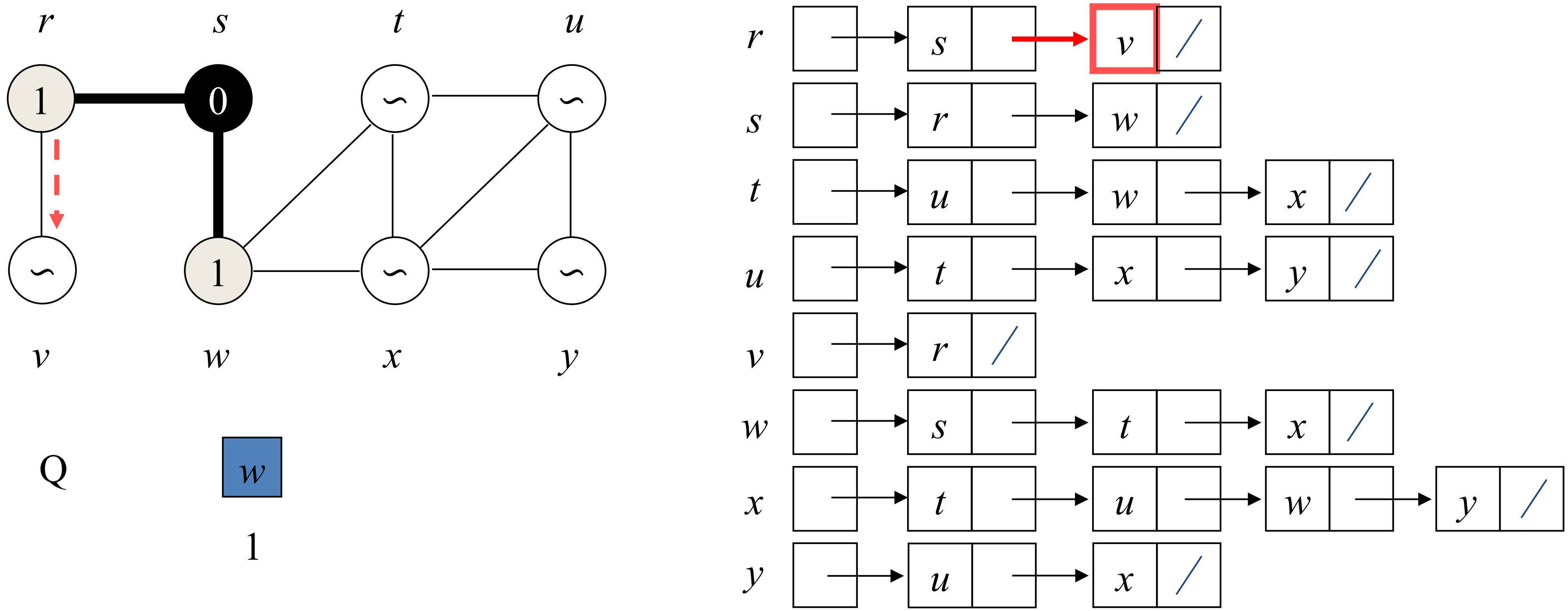

r
s
t
u
v
w
x
y

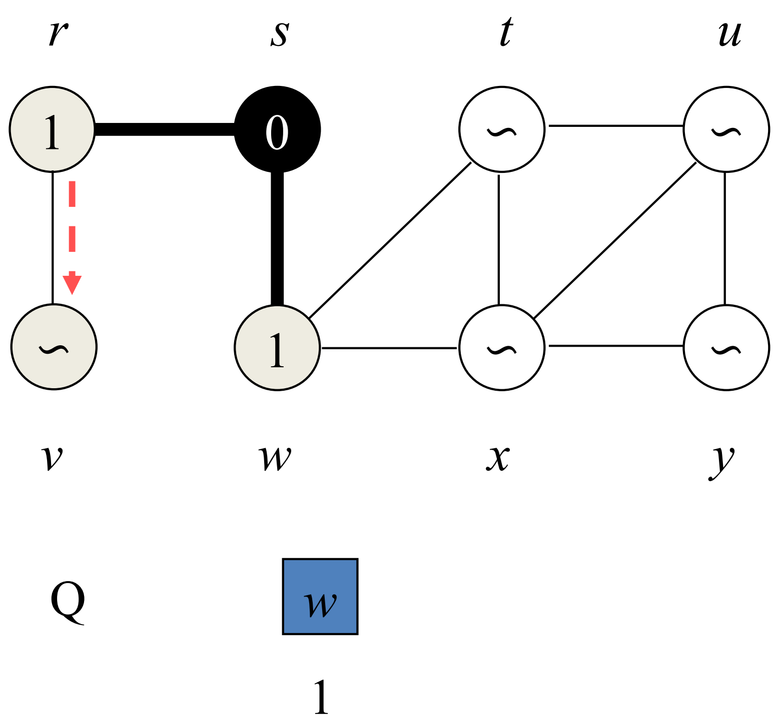
Breadth-first search

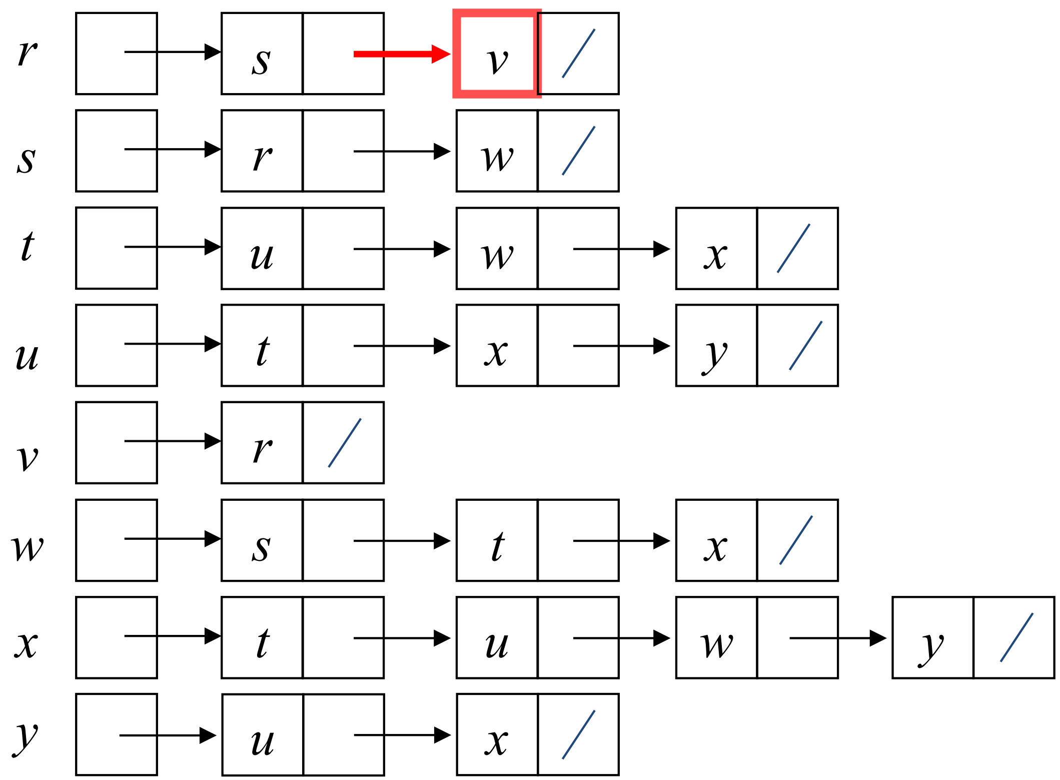


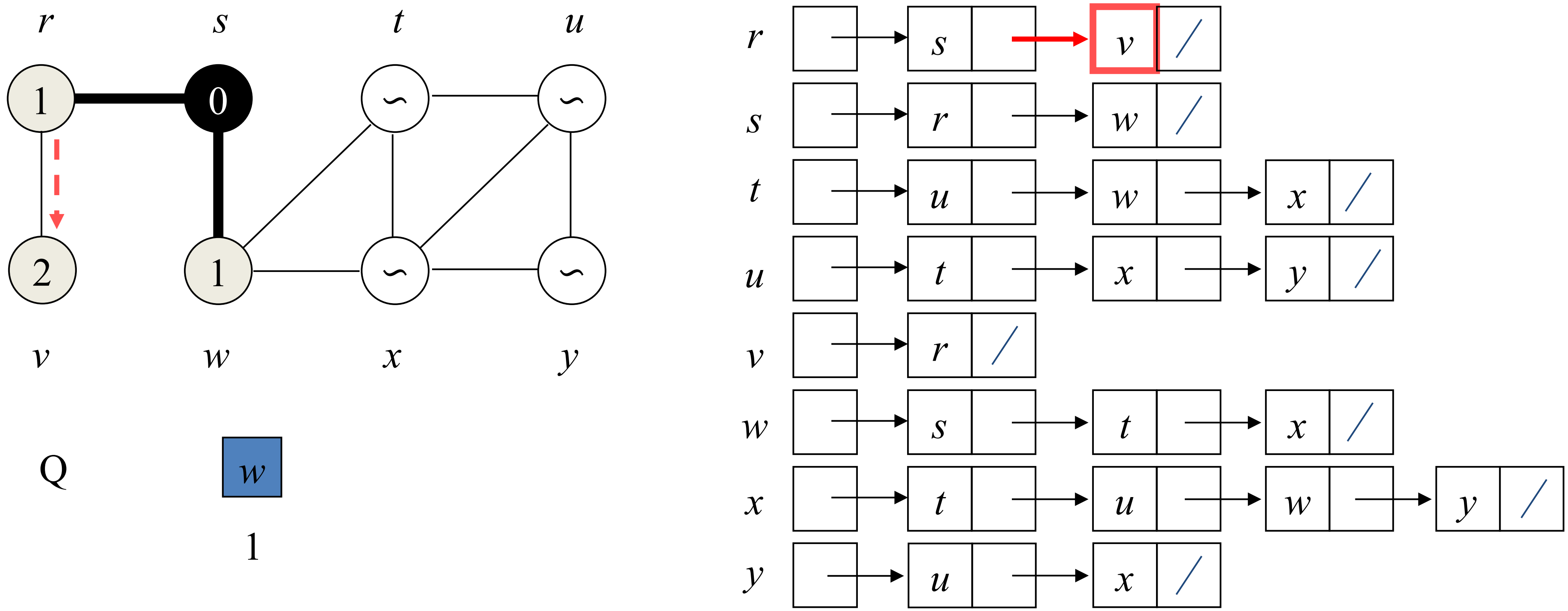


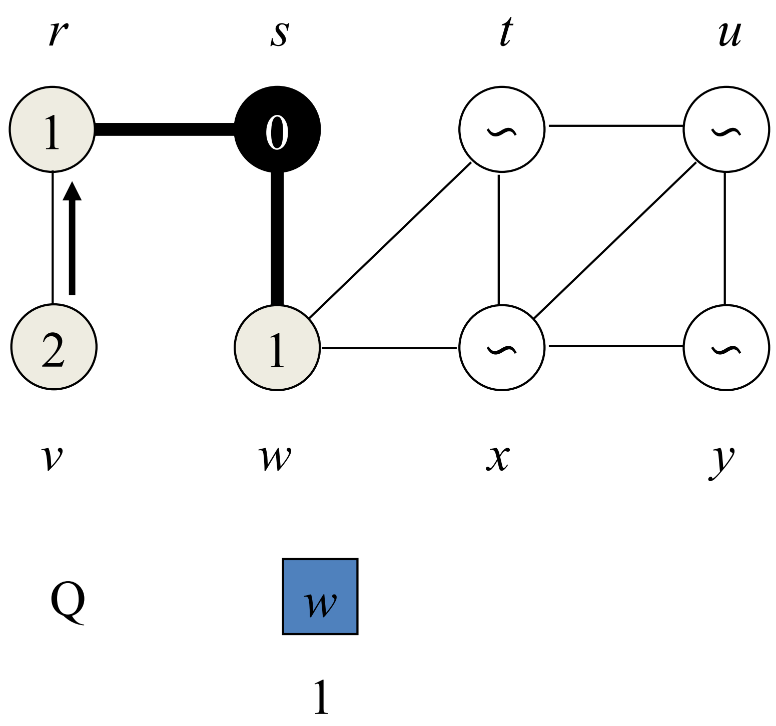


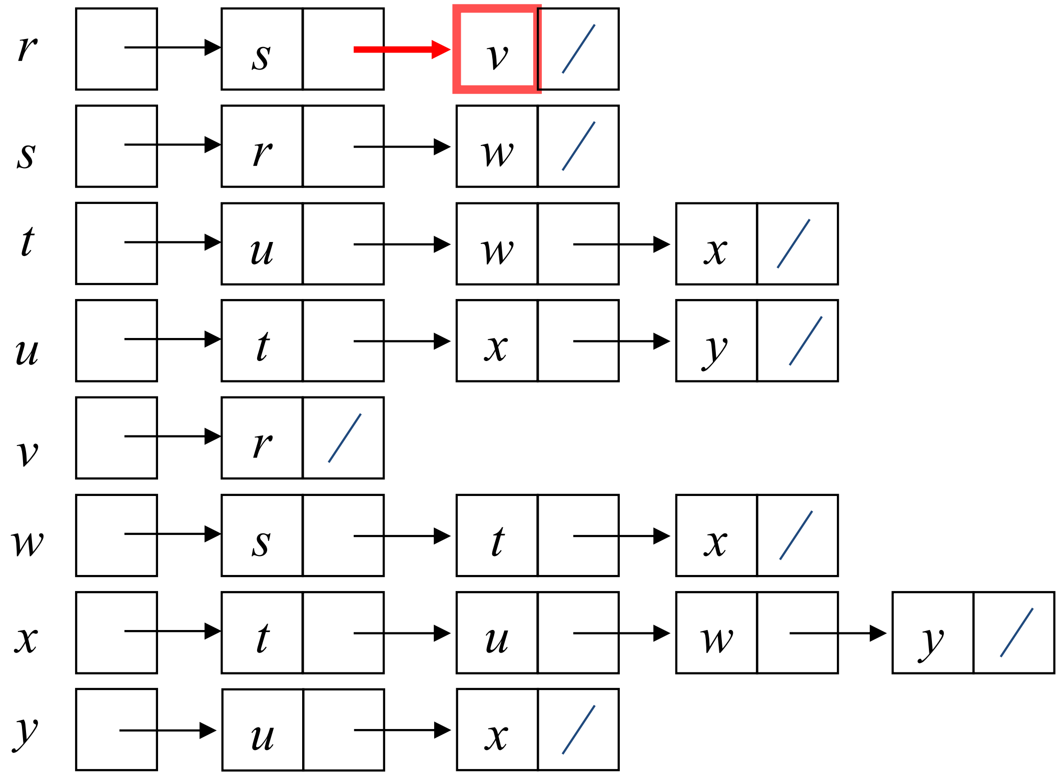

r
s
t
u
1
0
~
~
~
1
~
~
v
w
x
y
Q
w
1
r → s → v
s → r → w
t → u → w → x
u → t → x → y
v → r
w → s → t → x
x → t → u → w → y
y → u → x

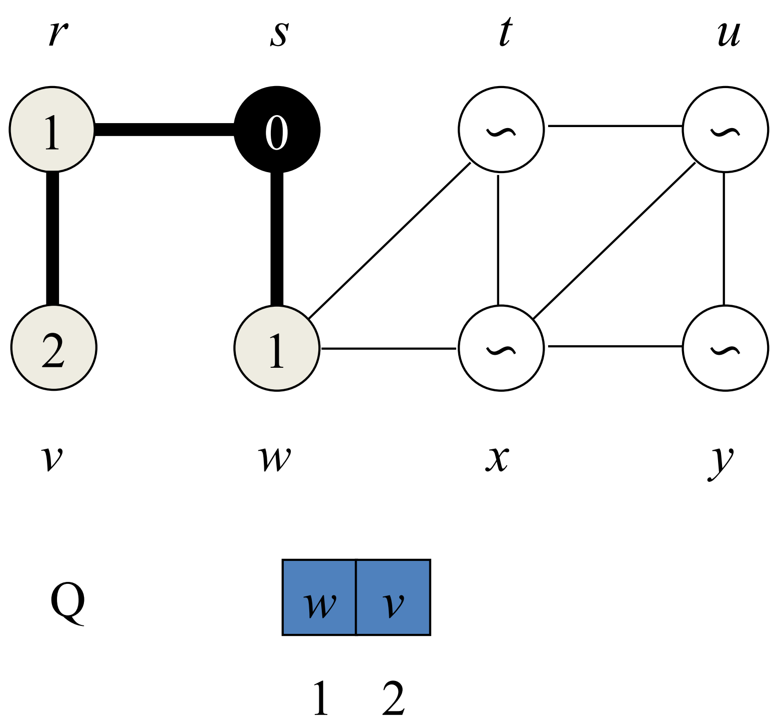

r
s
t
u
1
0
∽
∽
∽
1
∽
∽
v
w
x
y
Q
w
1

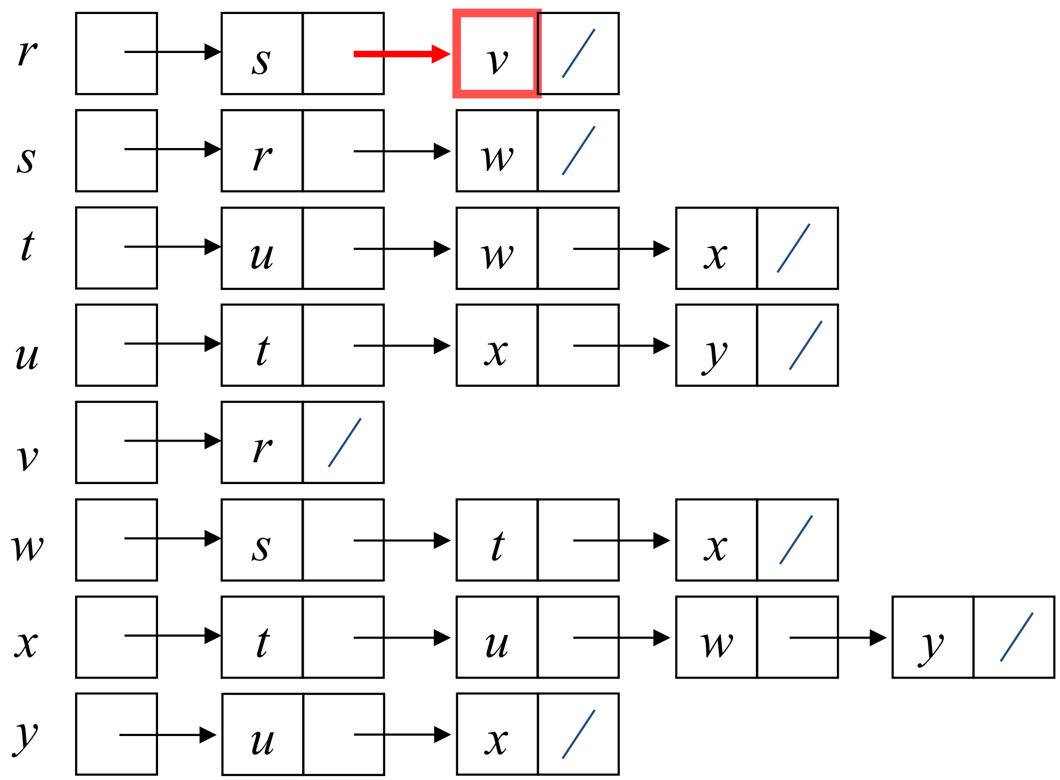

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
w
1
r: s → v /
s: r → w /
t: u → w → x /
u: t → x → y /
v: r /
w: s → t → x /
x: t → u → w → y /
y: u → x /

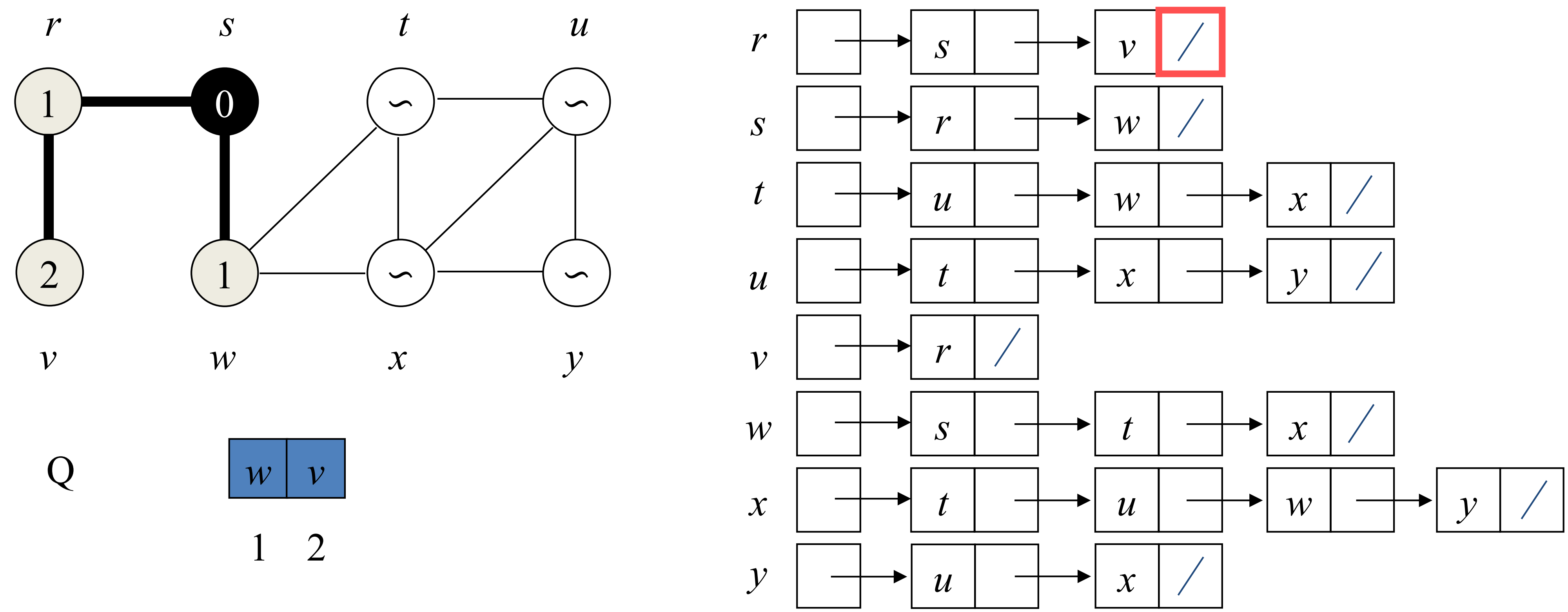

r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
w
1

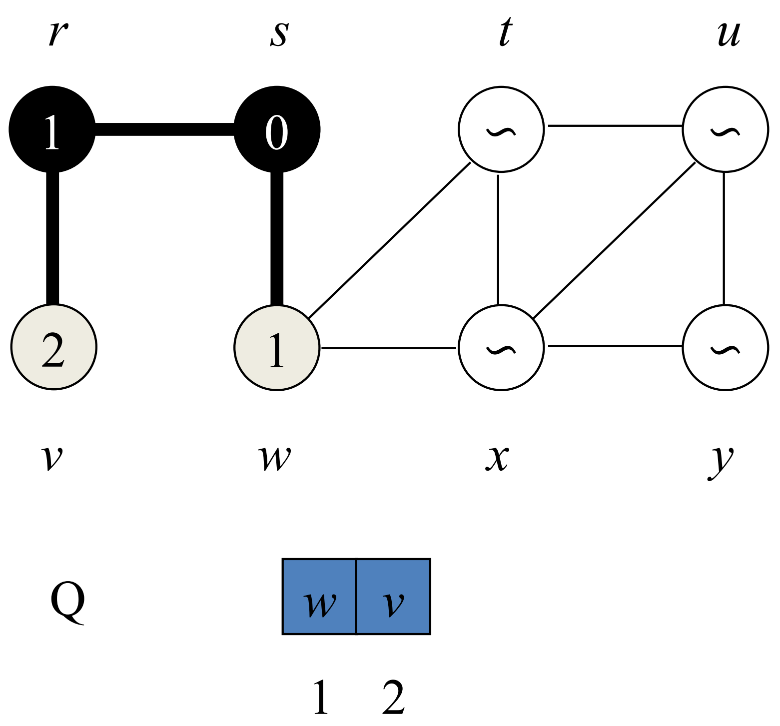

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

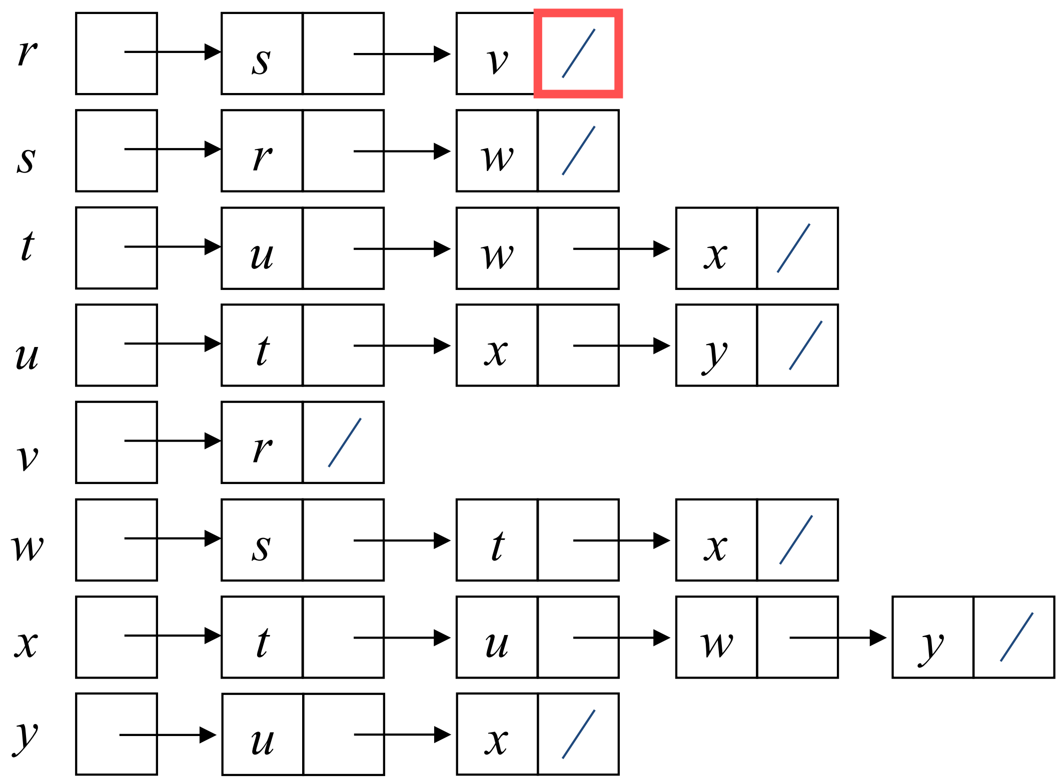

r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
w
v
1
2

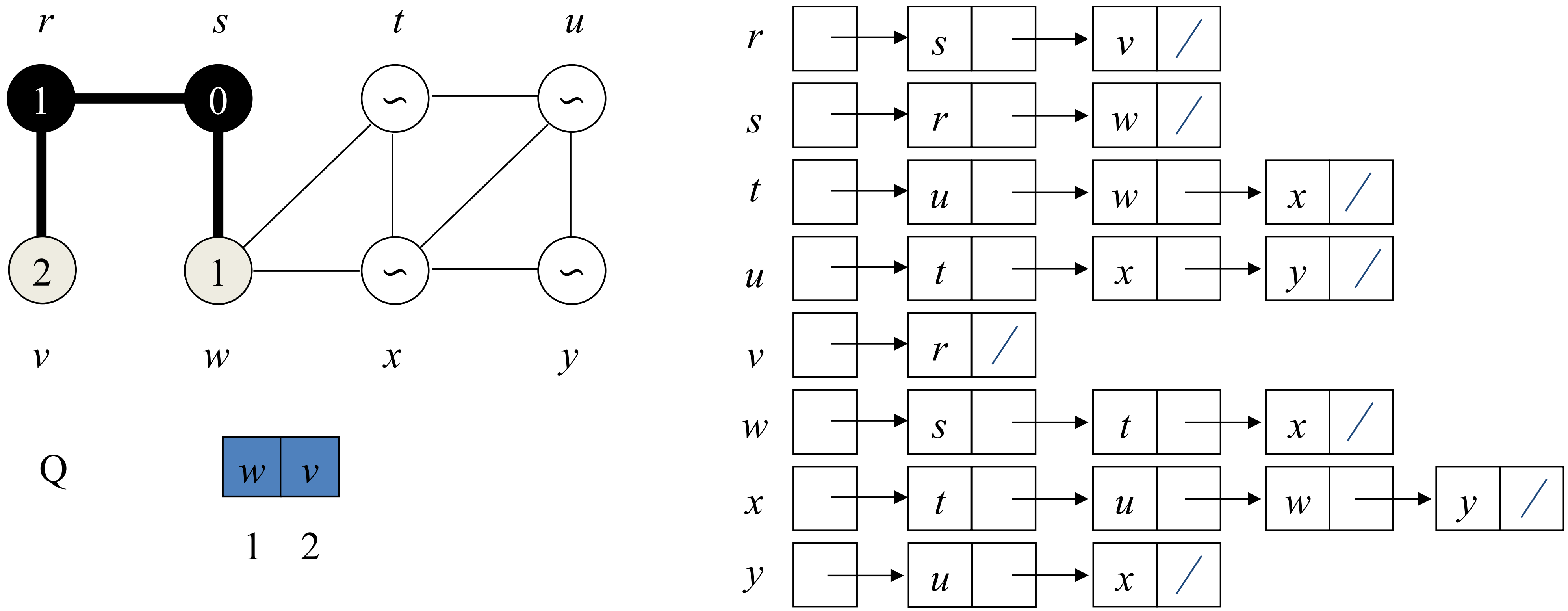

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

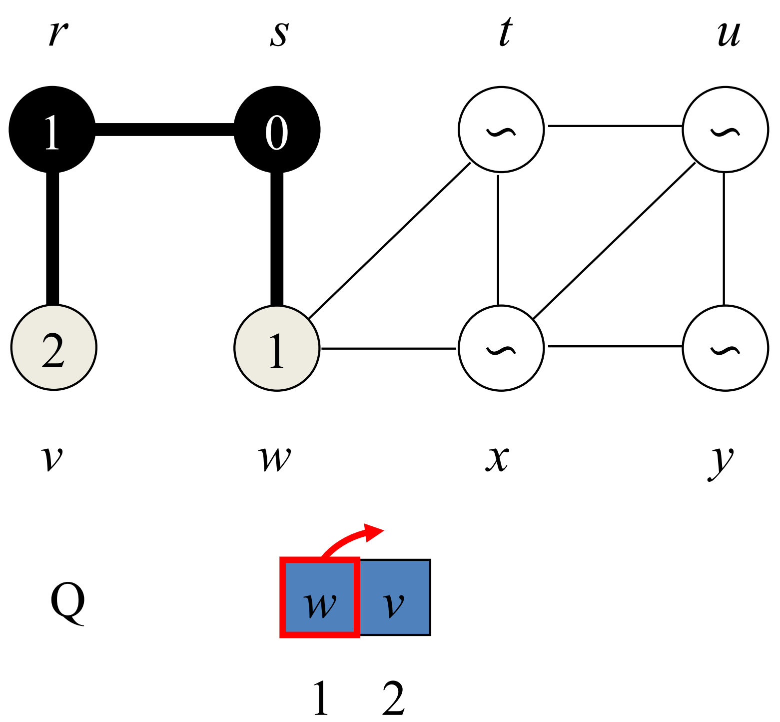
Breadth-first search

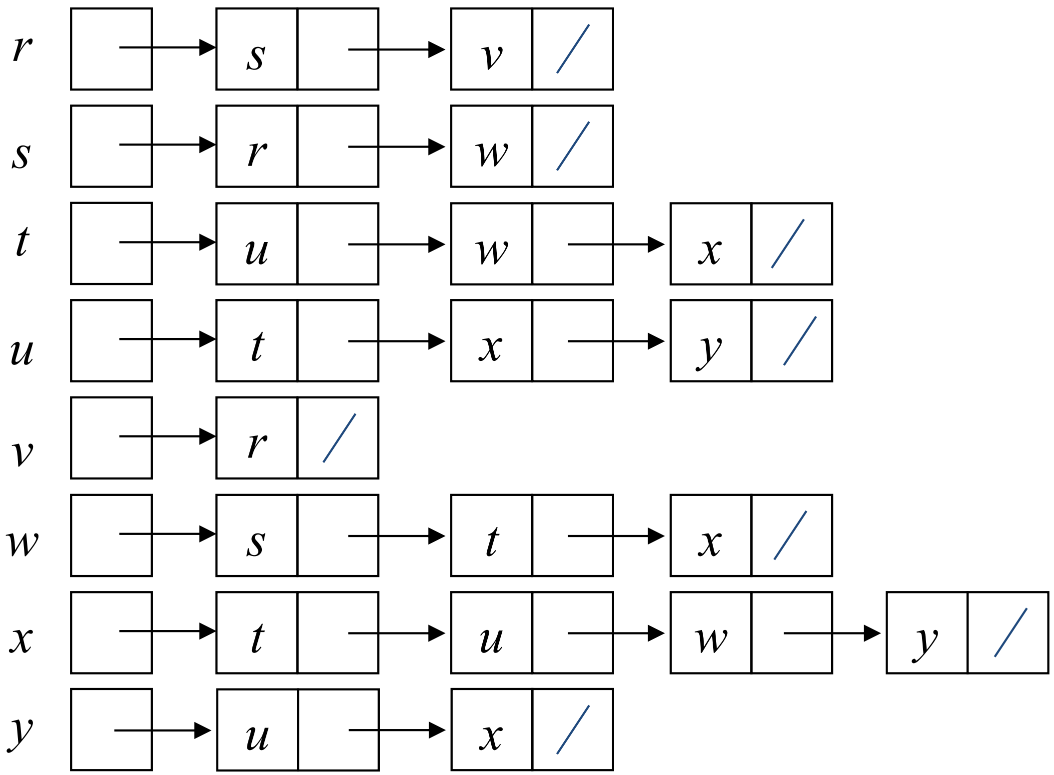


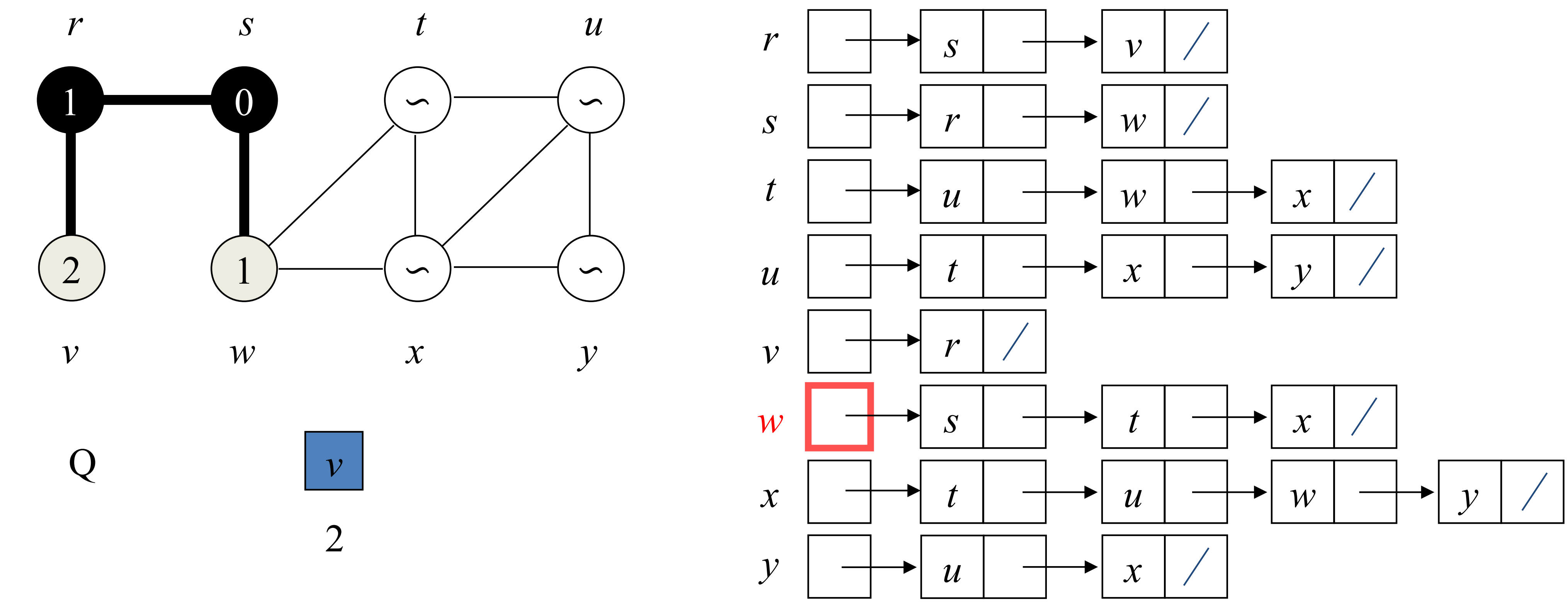

r
s
t
u
1
0
2
1
v
w
x
y
Q
w
v
1
2

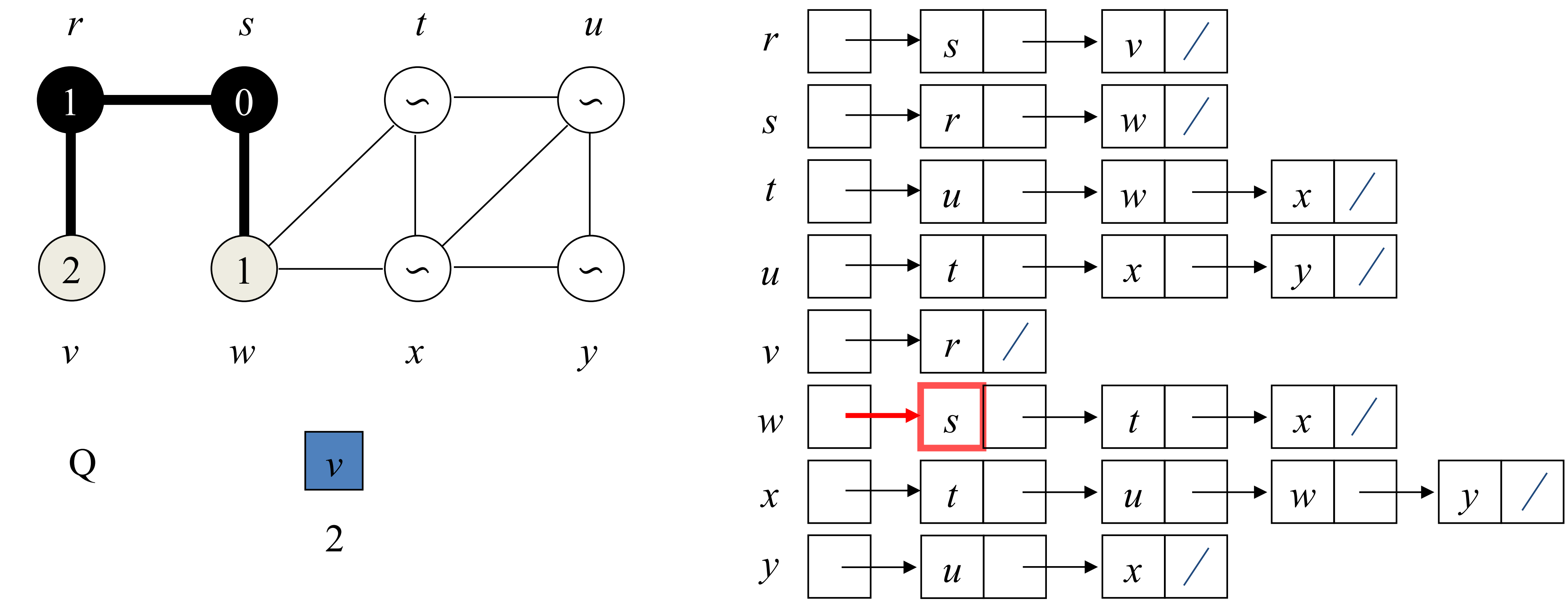

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

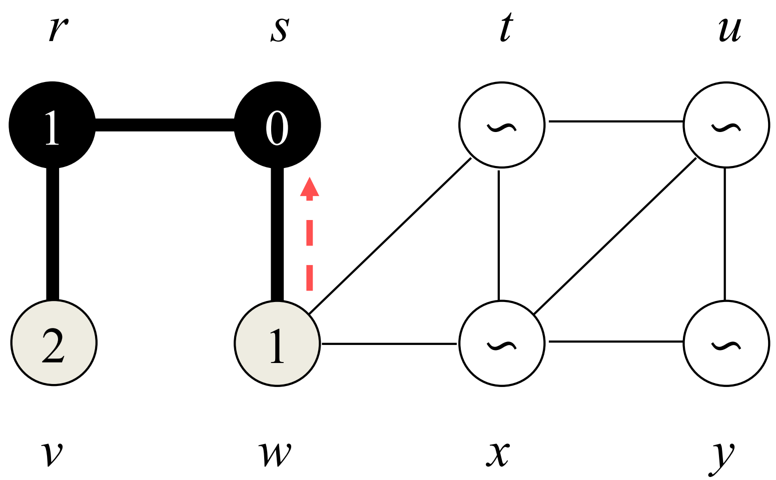

r
s
t
u
v
w
x
y
1
0
2
1
Q
w
v
1 2


r
s
t
u
1
0
2
1
v
w
x
y
Q
w
v
1
2

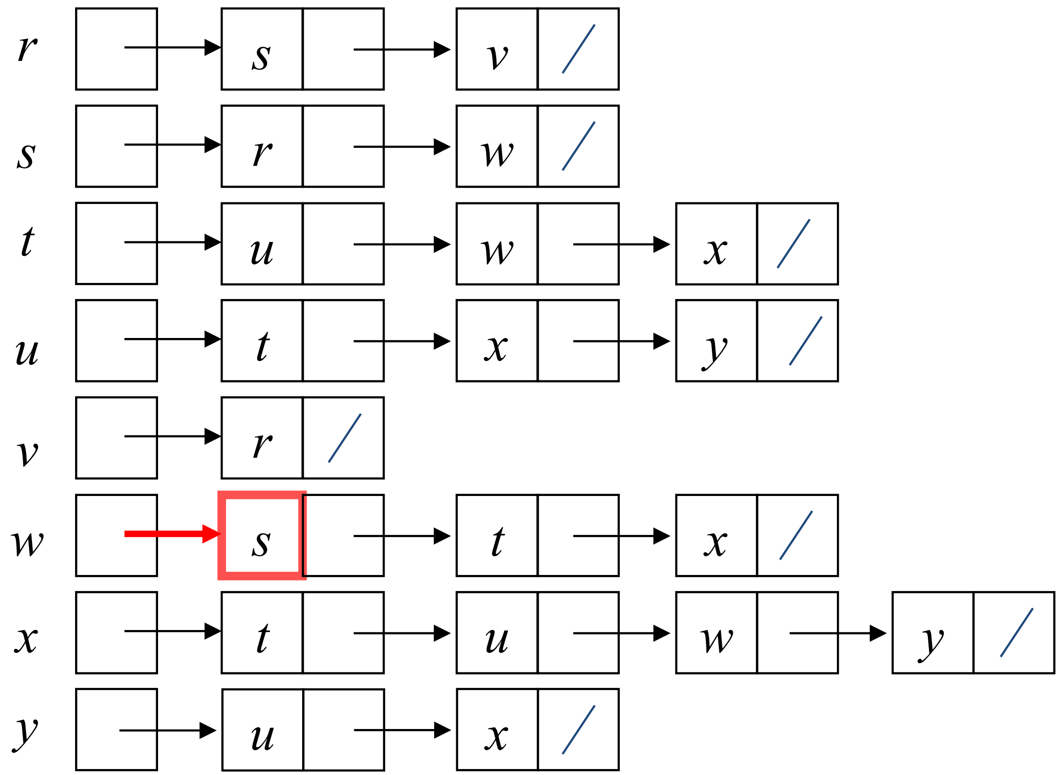

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

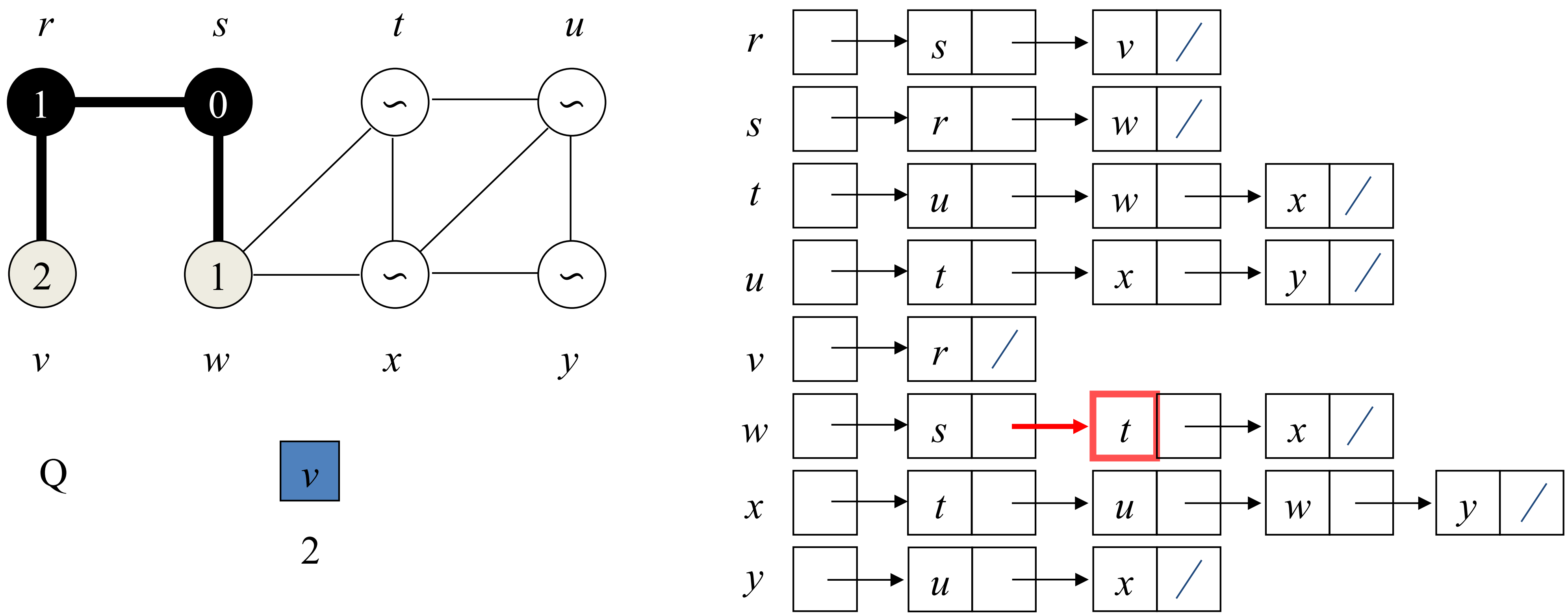

r
s
t
u
1
0
2
1
v
w
x
y
Q
v
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

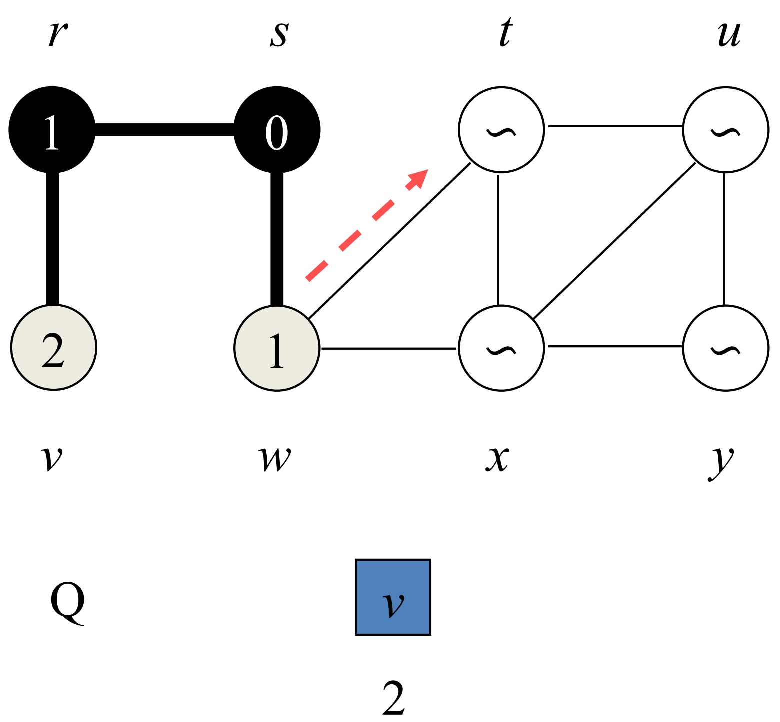

r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
v
2
r: s → v
s: r → w
t: u → w → x
u: t → x → y
v: r
w: s → t → x
x: t → u → w → y
y: u → x


r
s
t
u
1
0
2
1
v
w
x
y

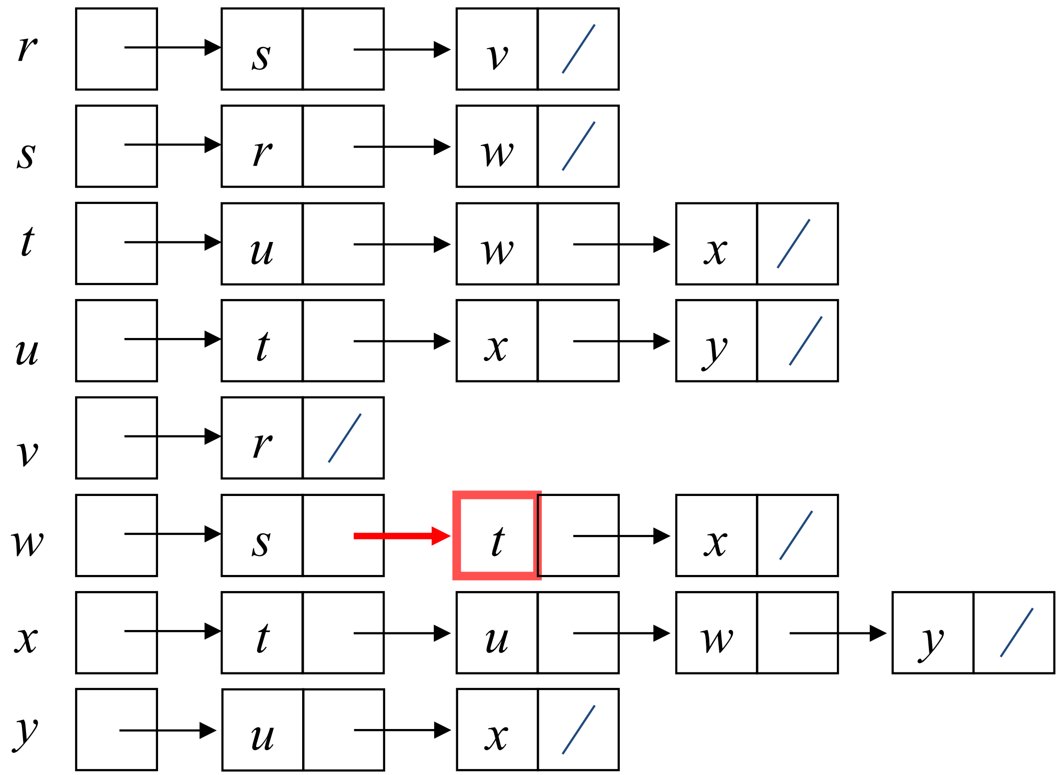

Q
v
2


r
s
t
u
v
w
x
y

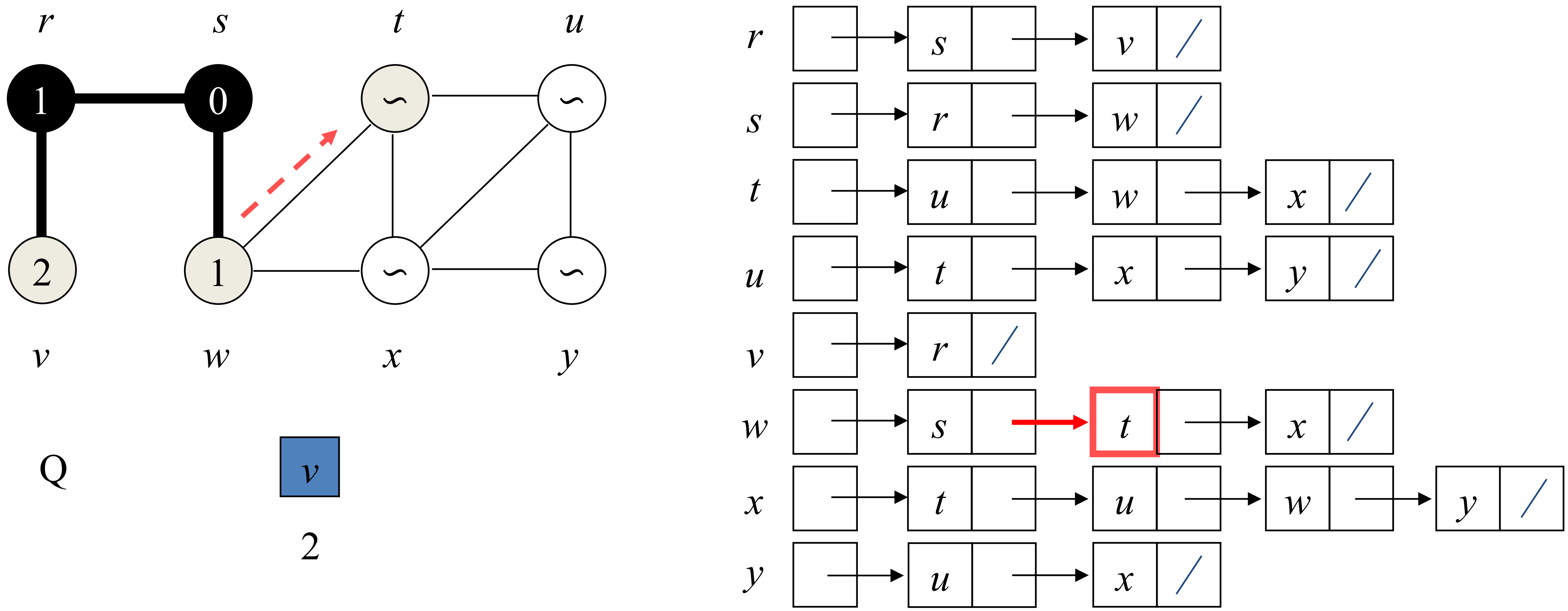

r
s
t
u
1
0
2
1
v
w
x
y
Q
v
2
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

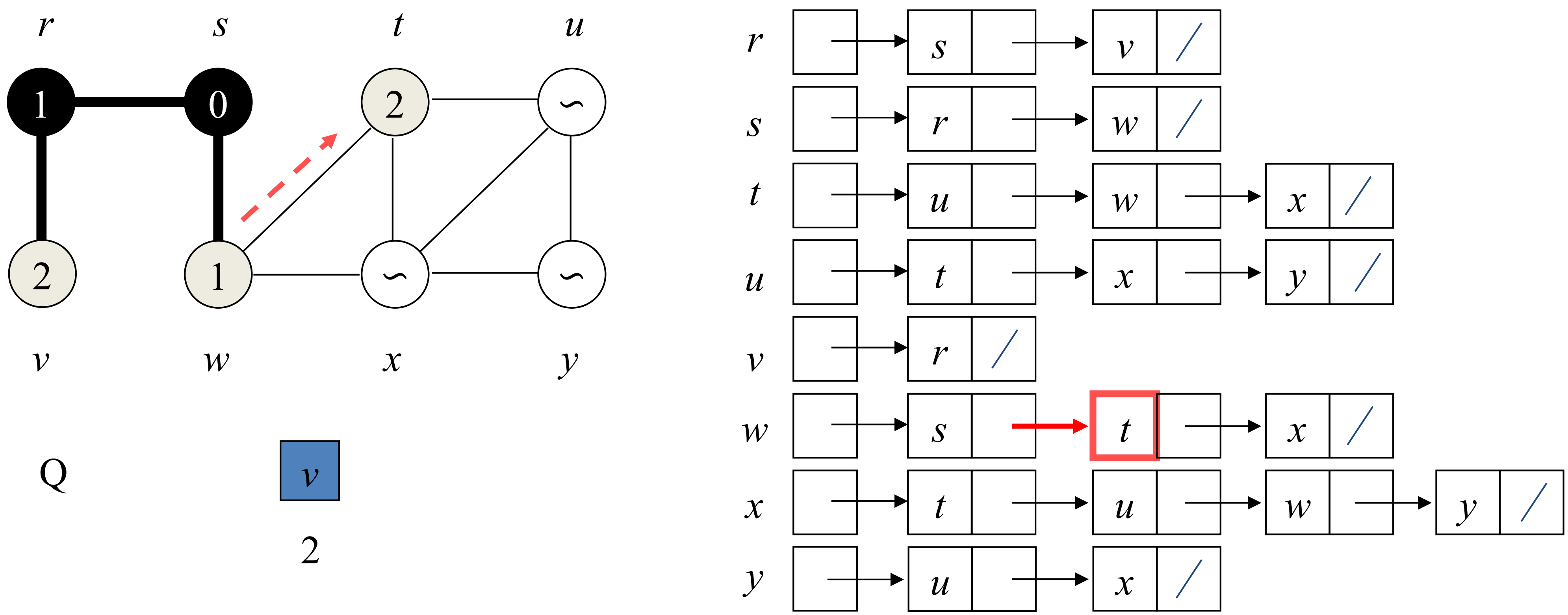


r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
v
2

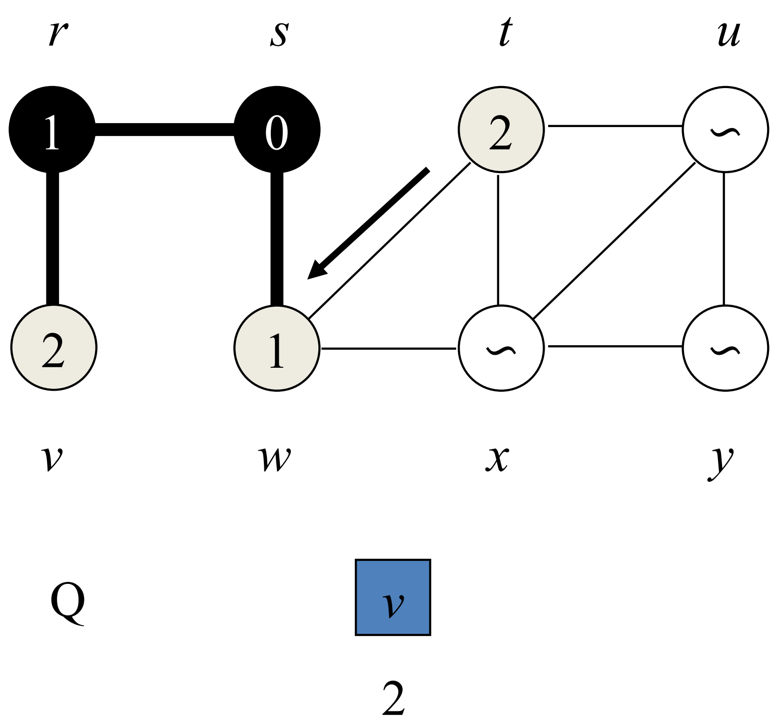


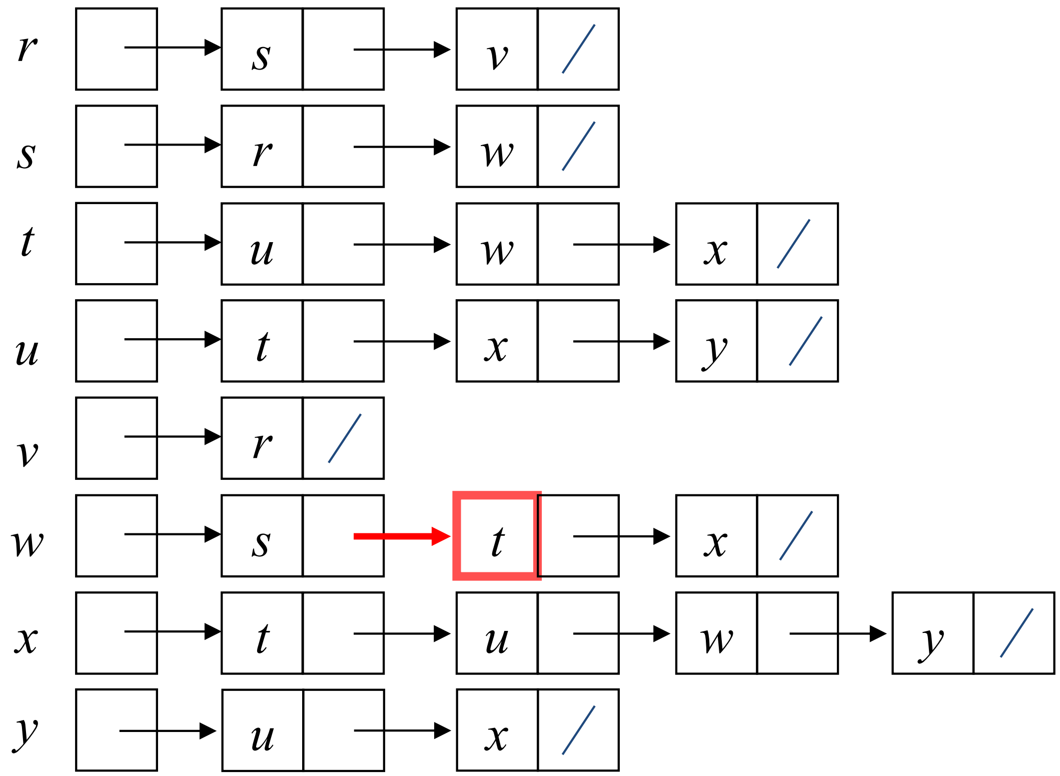
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
~
~
2
1
~
~
v
w
x
y
Q
v
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

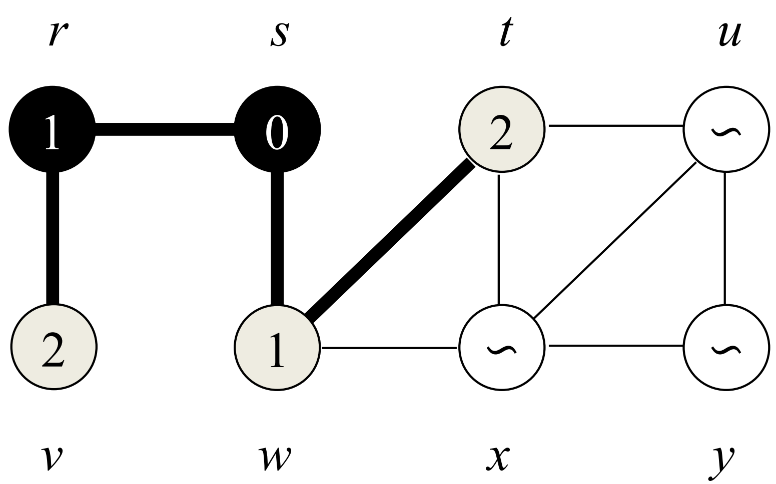

r
s
t
u
1
0
2
~
2
1
~
~
v
w
x
y
Q
v
2
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

Breadth-first search

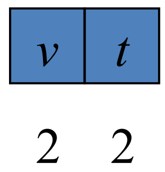


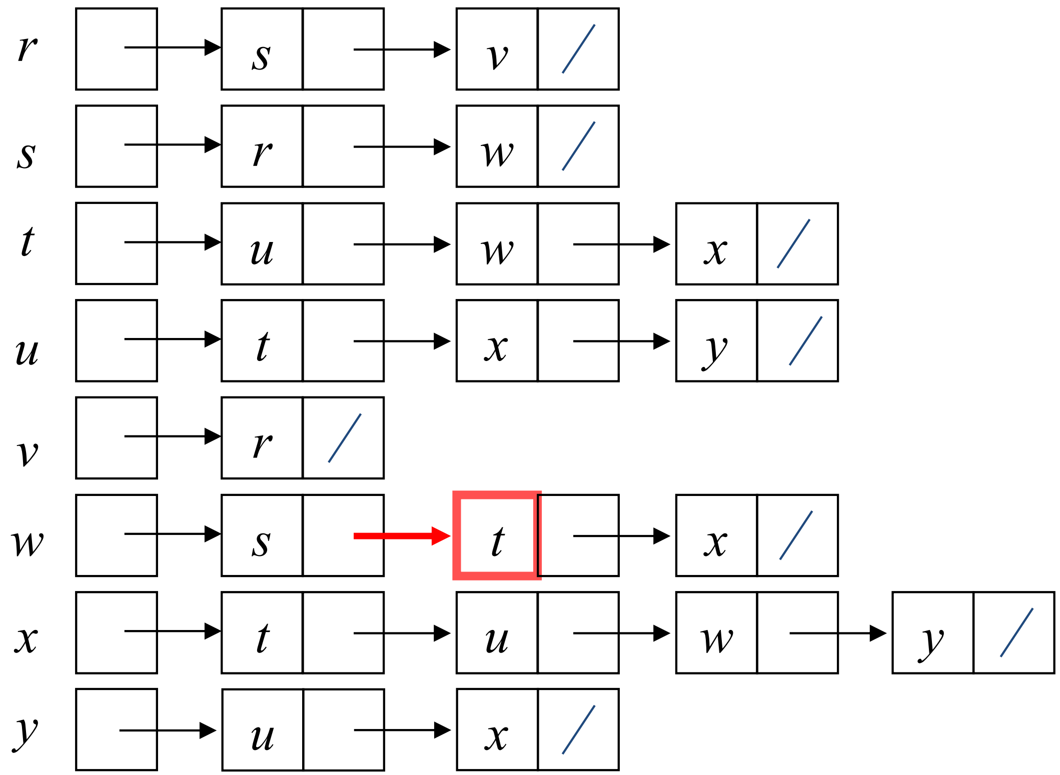


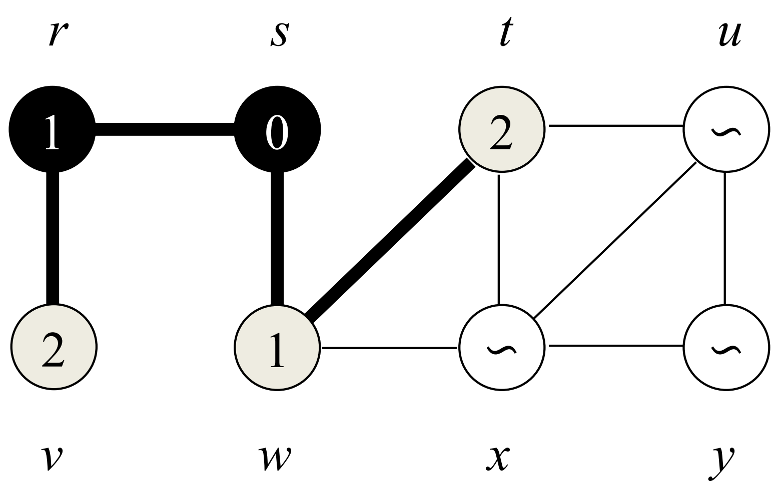
Breadth-first search



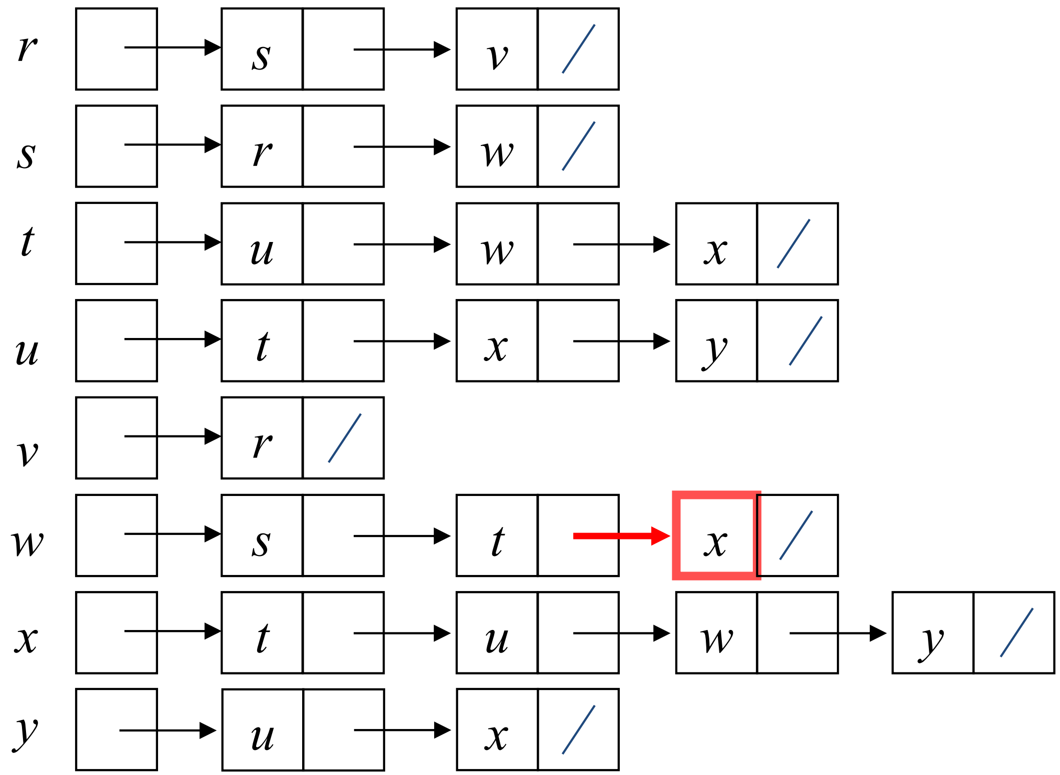
Q

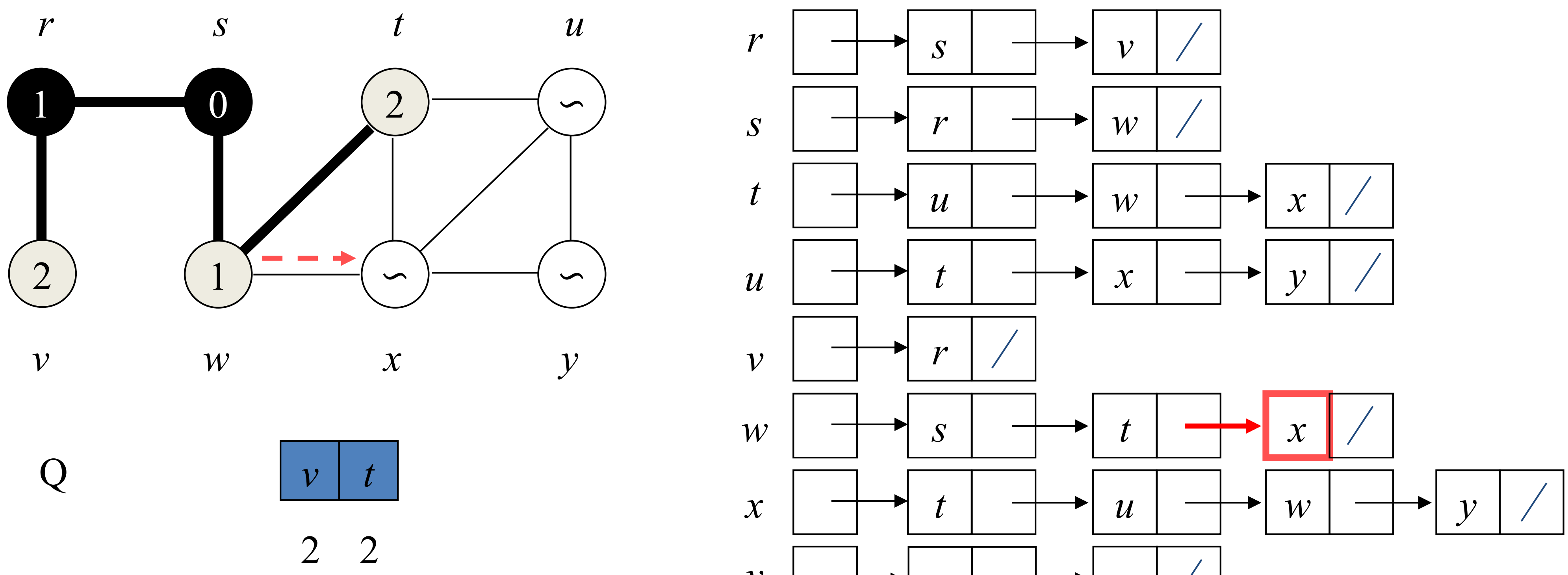


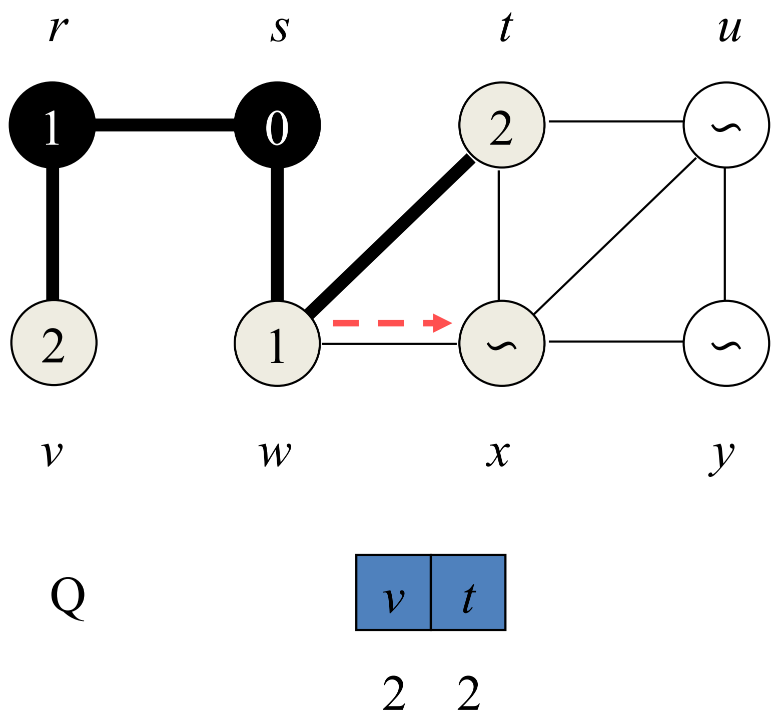


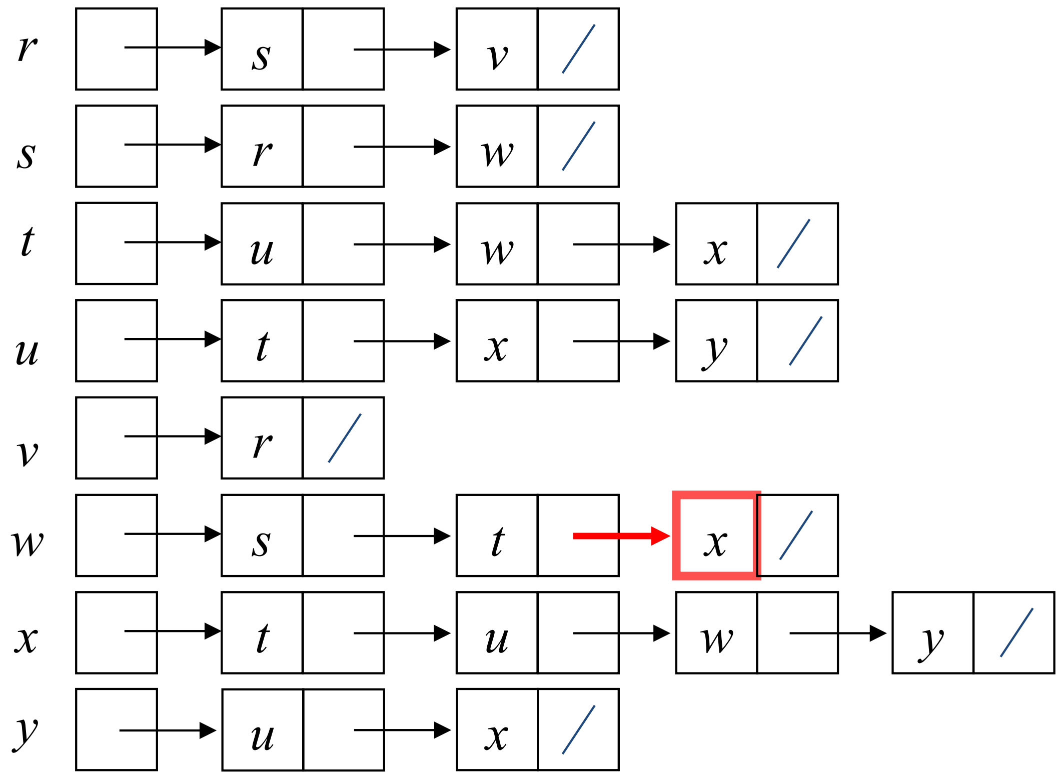


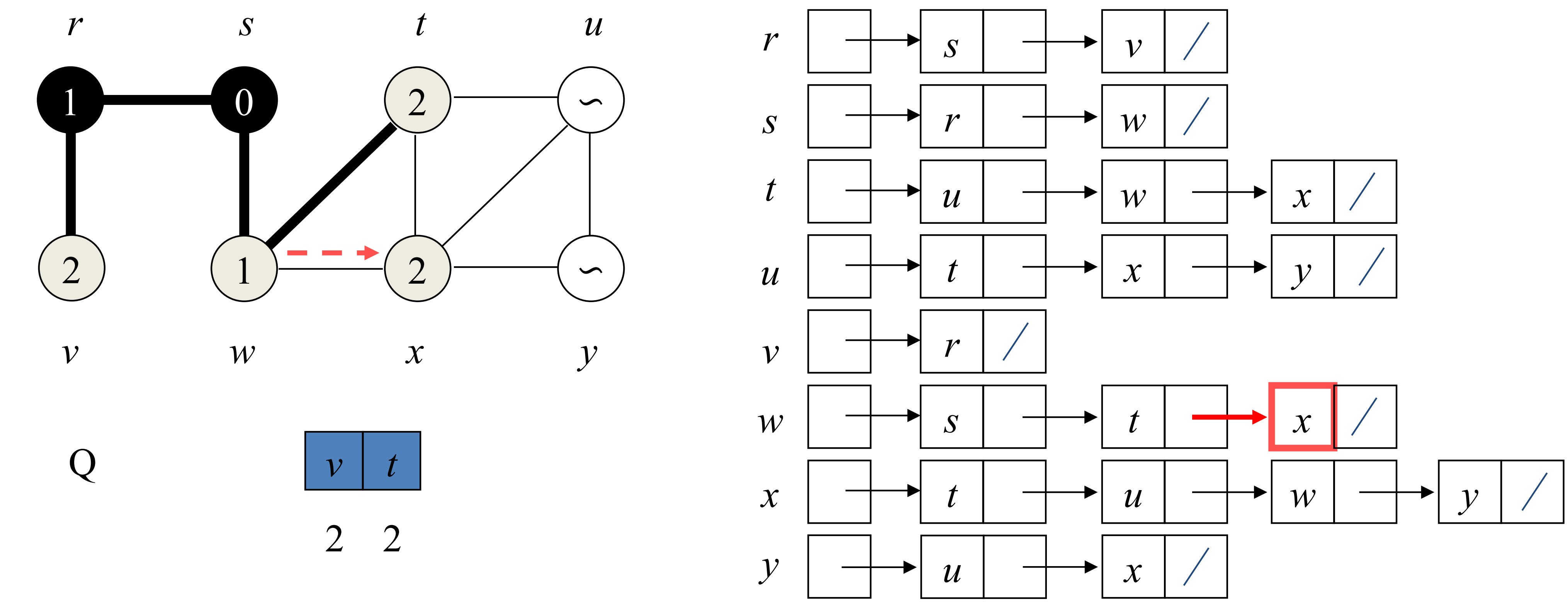
Q	v	t
	2	2

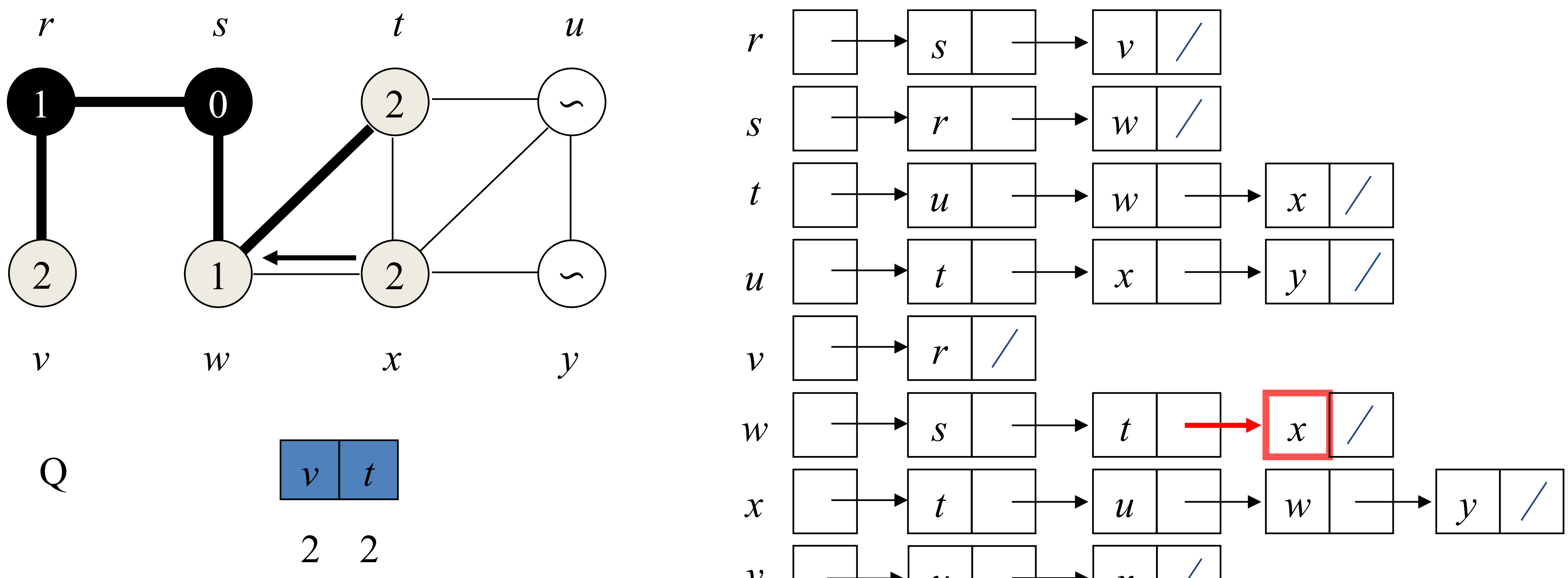


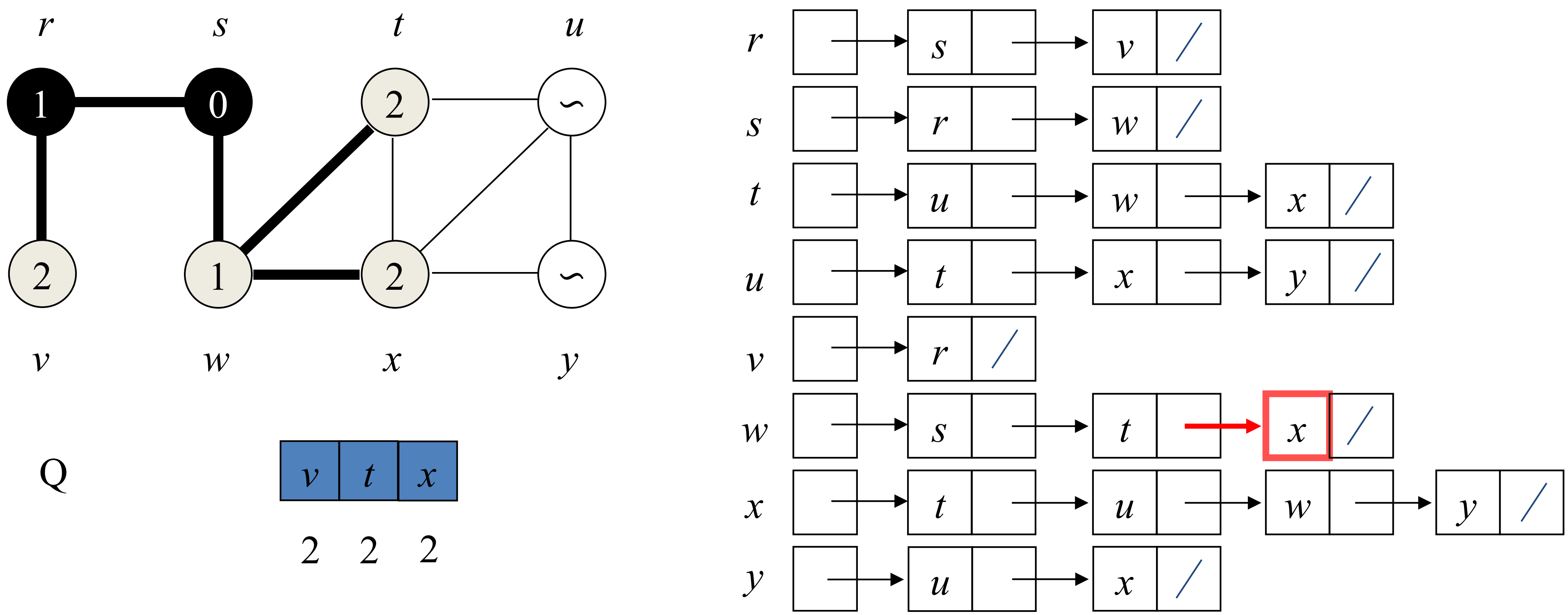

r
s
t
u
1
0
2
2
1
v
w
x
y
Q
v
t
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

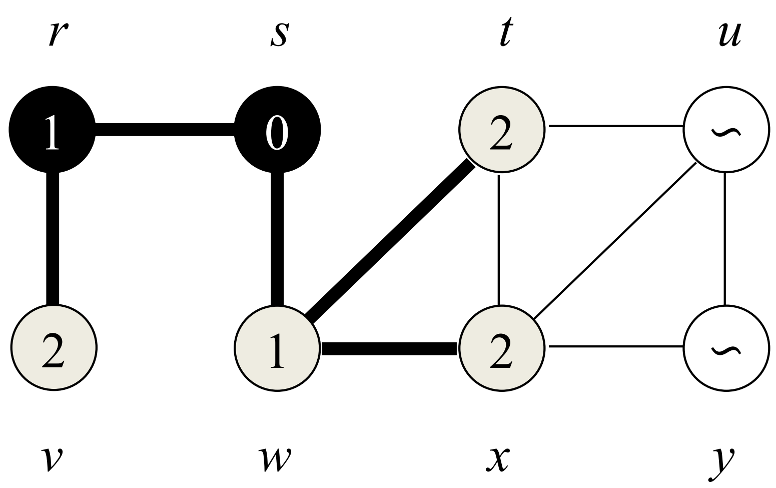

r
s
t
u
1
0
2
~
2
1
~
~
v
w
x
y
Q
v
t
2
2


r
s
t
u
v
w
x
y


r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

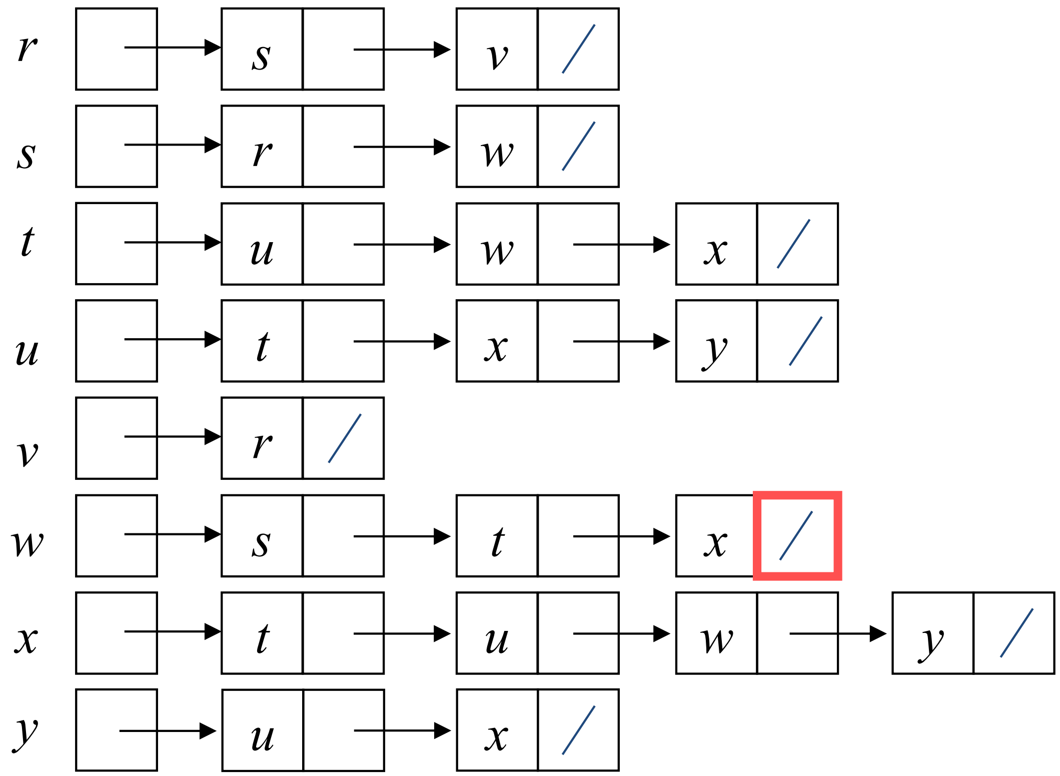

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

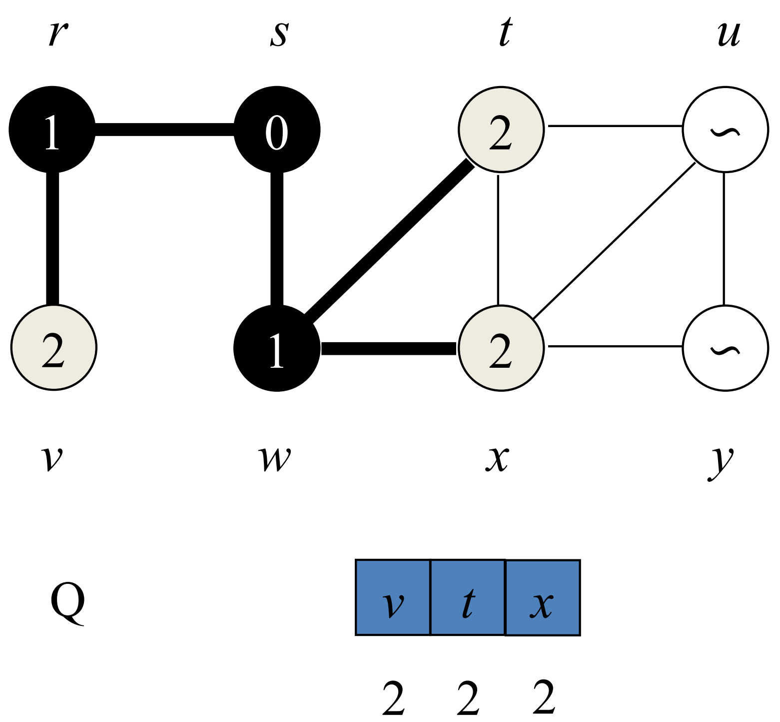

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
x
2
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

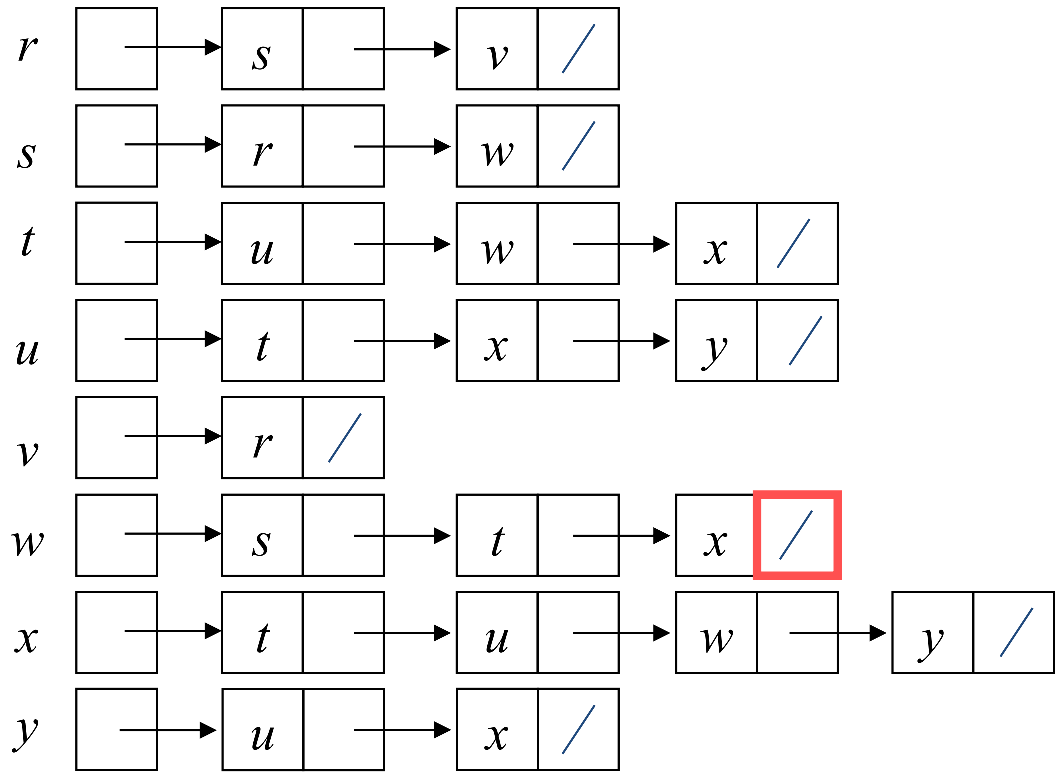


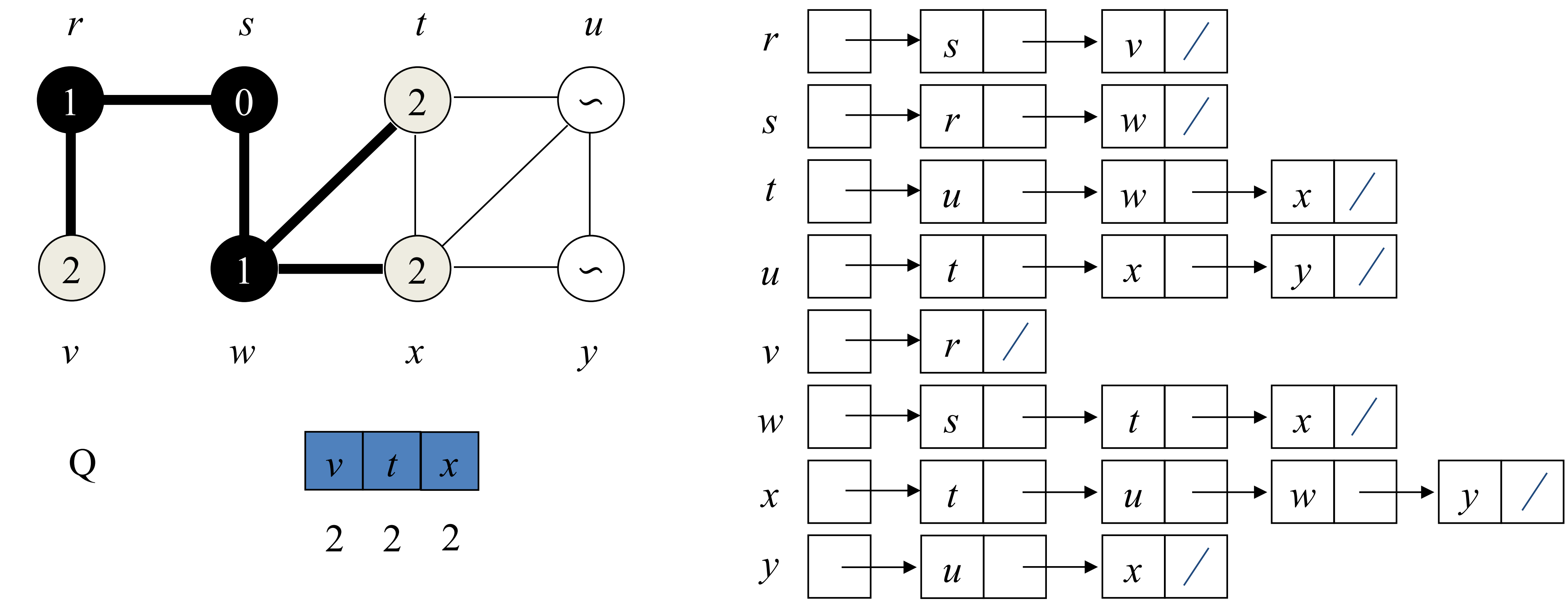
Q

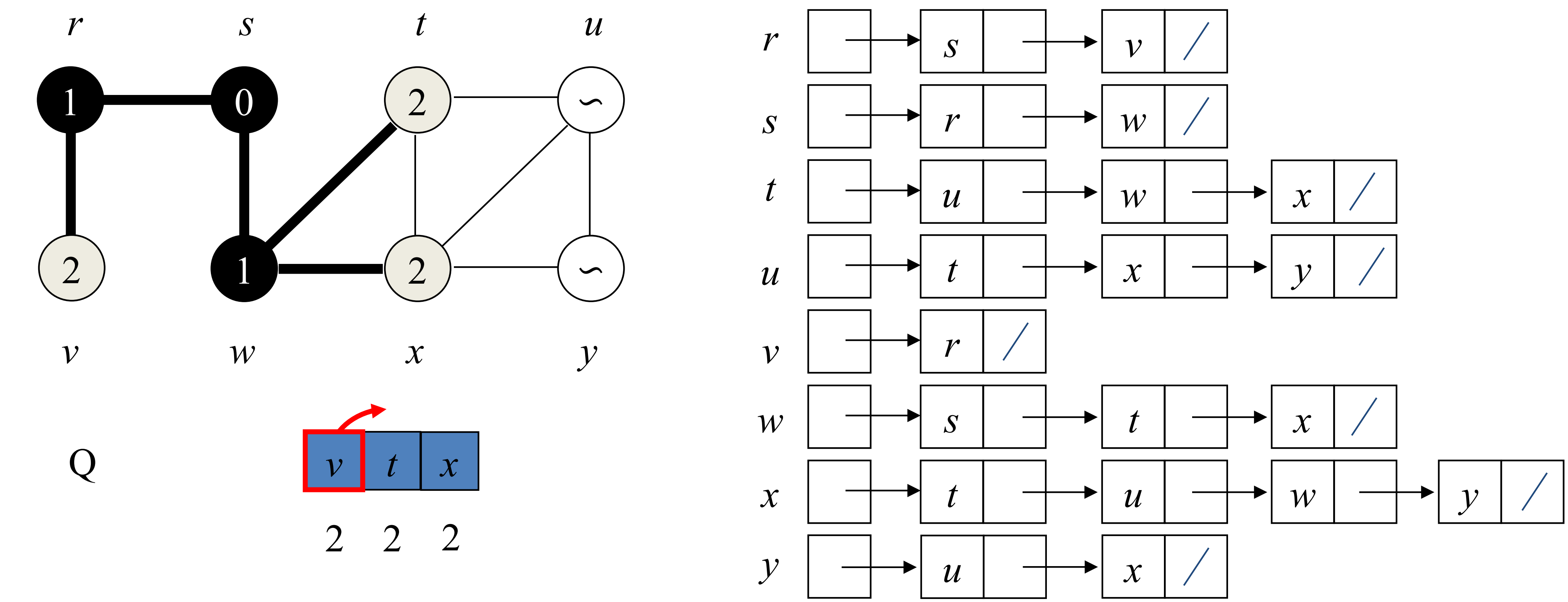
v	t	x
2	2	2

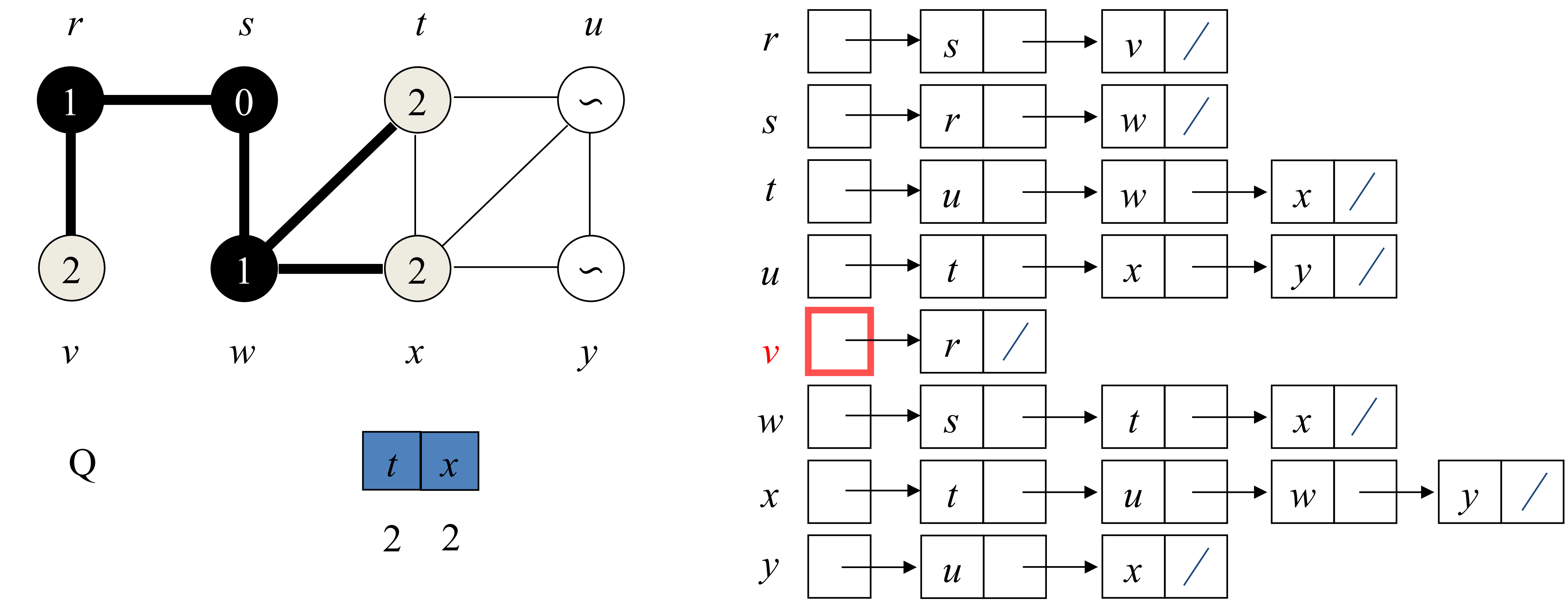



r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
x
2
2
2

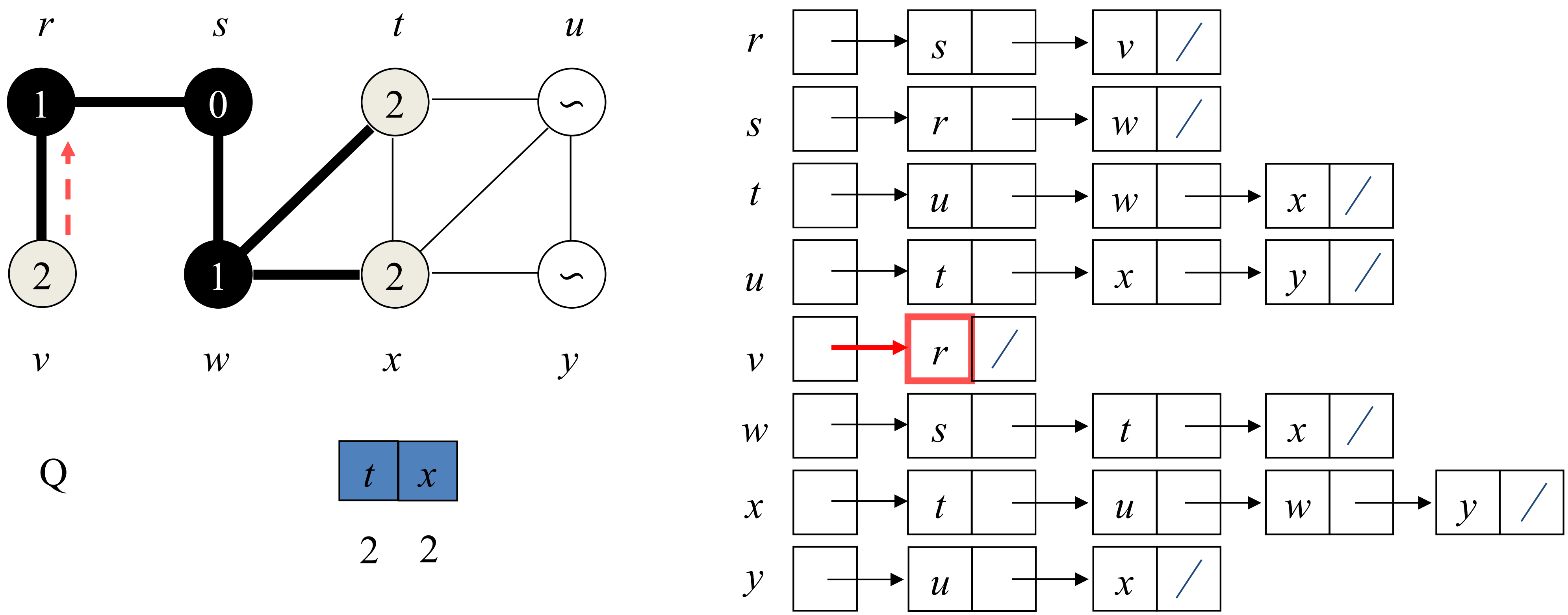

r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

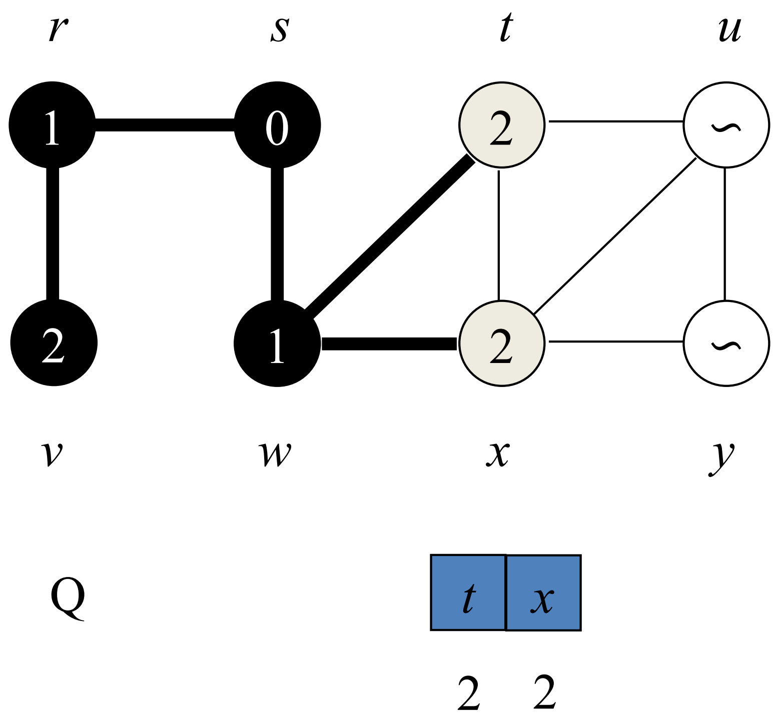

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
x
2 2 2
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

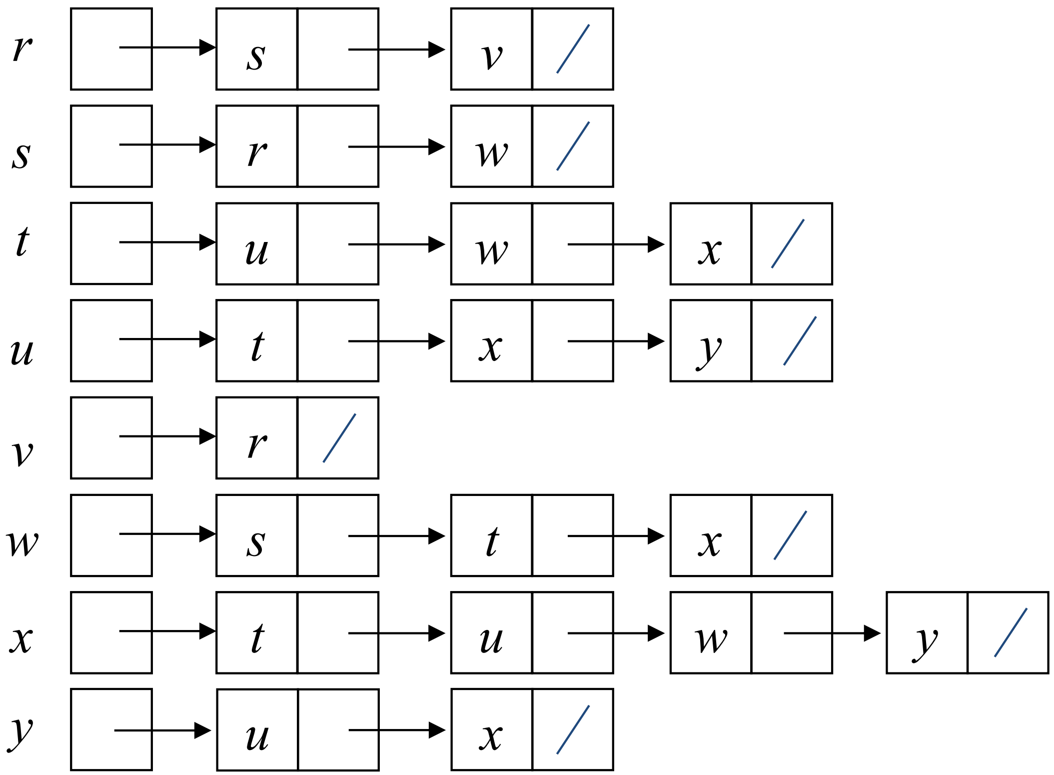

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
v
t
x
2
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

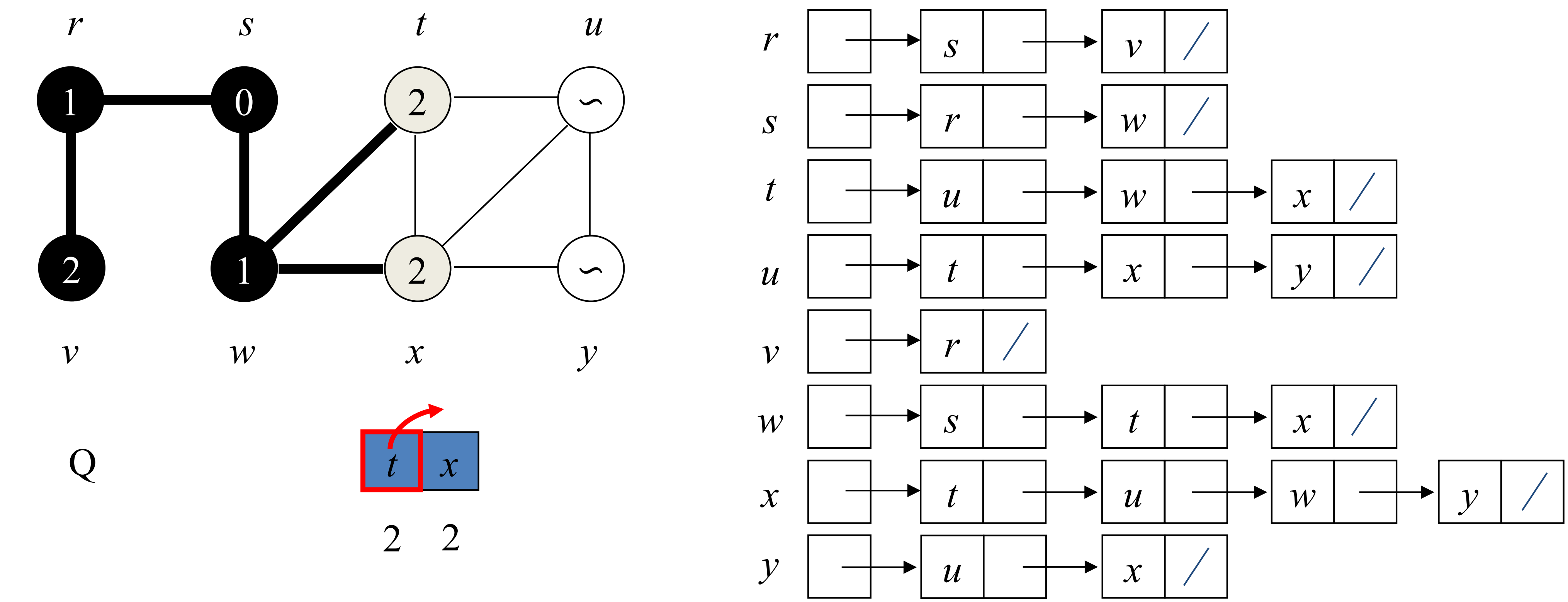

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
t
x
2
2
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

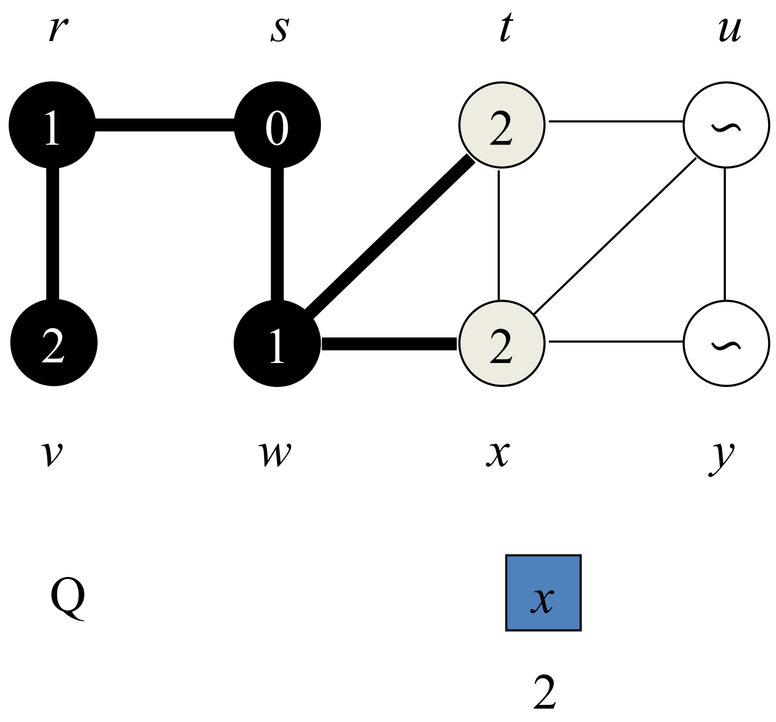
Breadth-first search

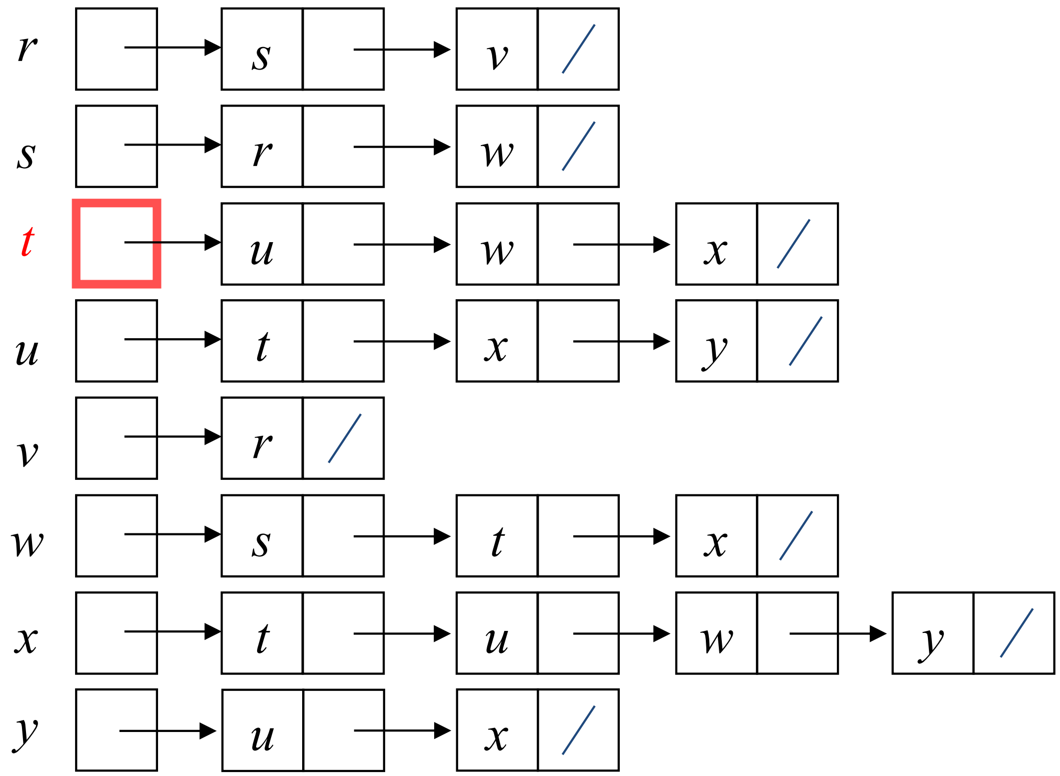


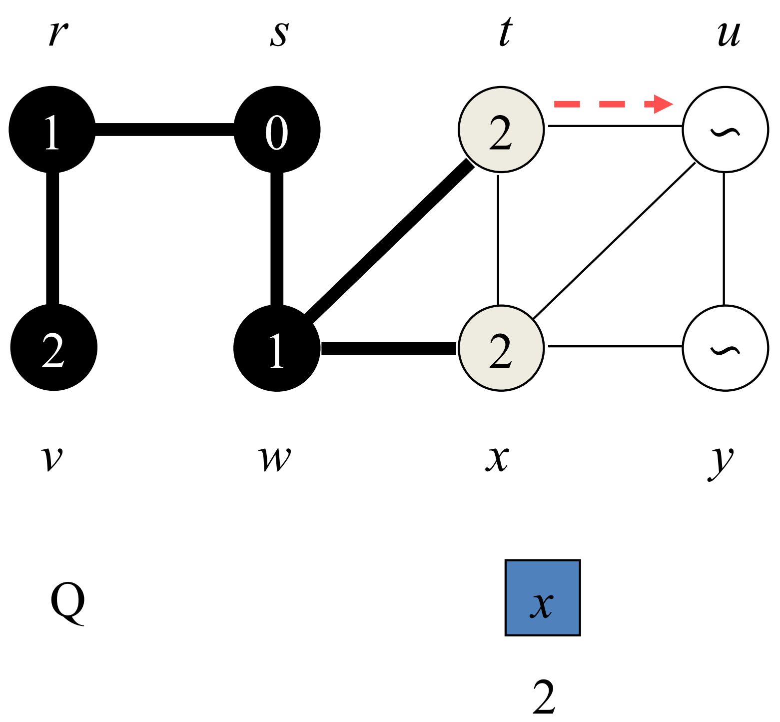

r
s
t
u
1
0
2
∽
2
1
2
∽
v
w
x
y
Q
t
x
2
2

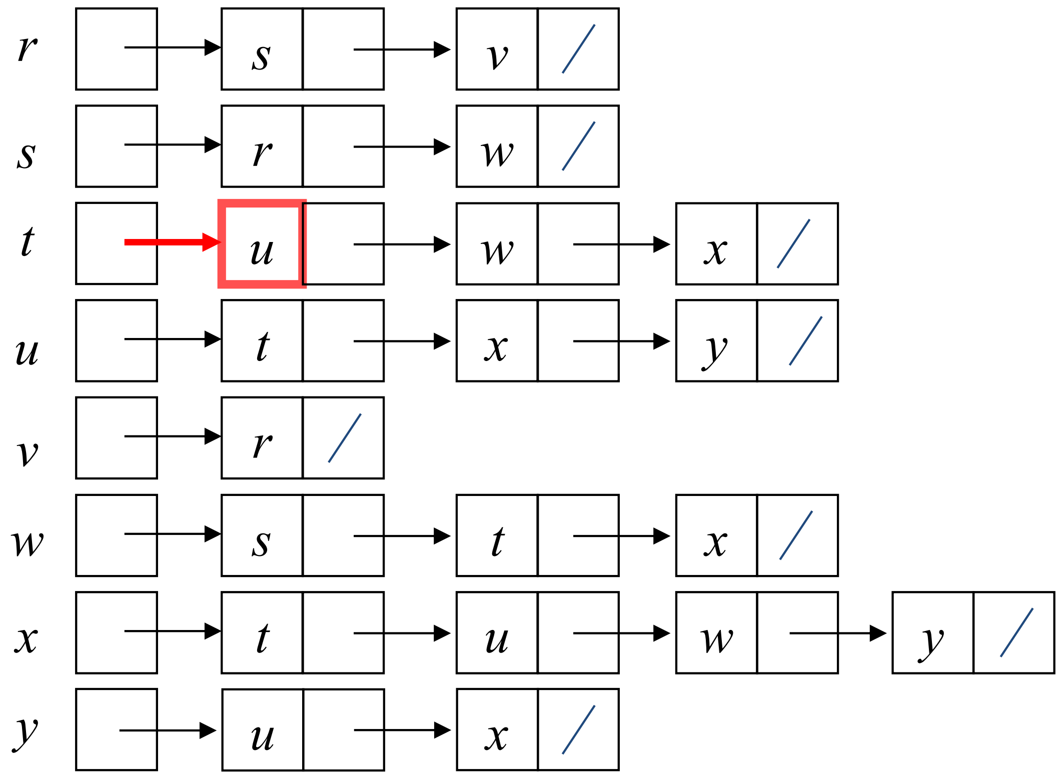

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
2
2
1
2
v
w
x
y
Q
t
x
2
2
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

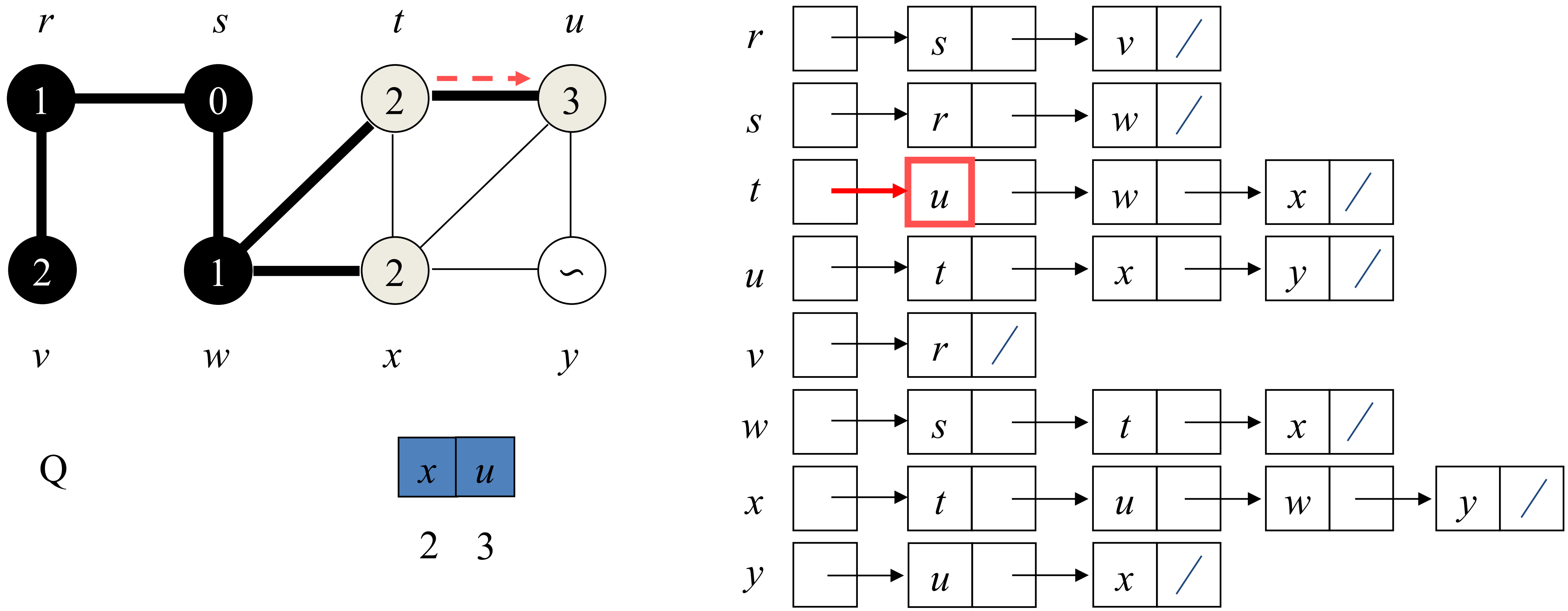

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
x
2


r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

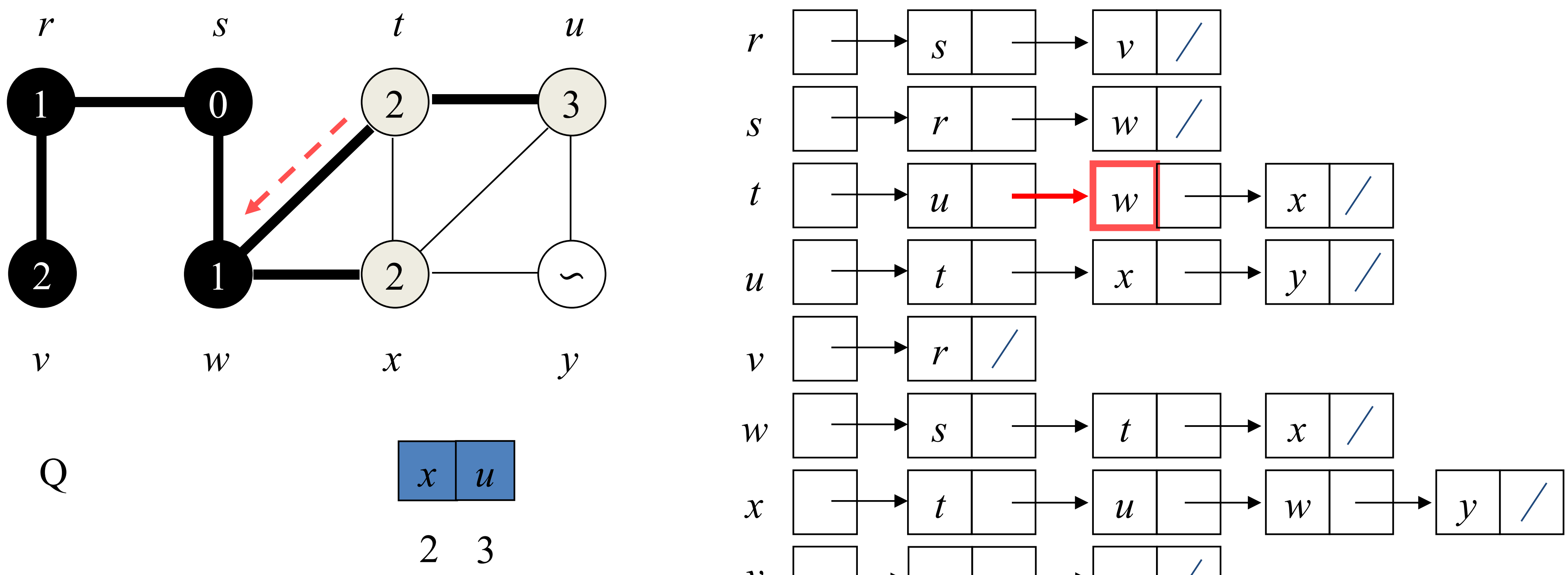

r
s
t
u
1
0
2
~
2
1
2
~
v
w
x
y
Q
x
2


r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

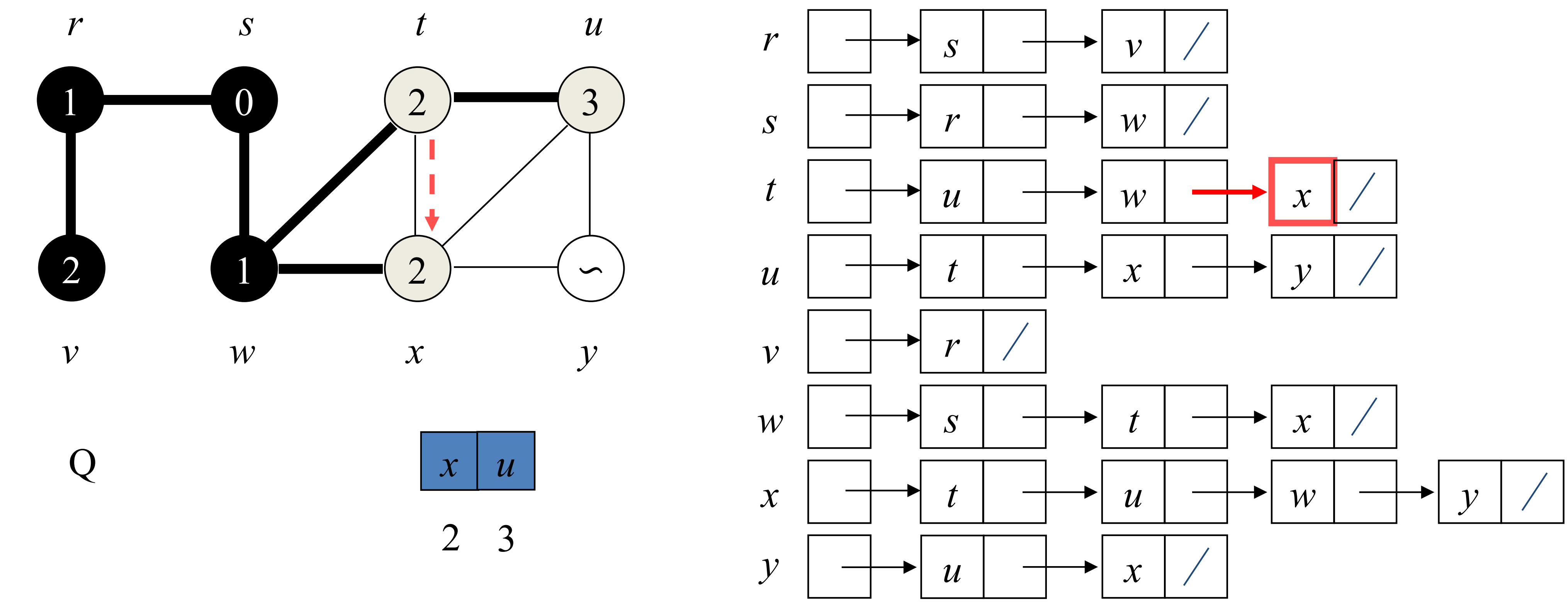
Breadth-first search

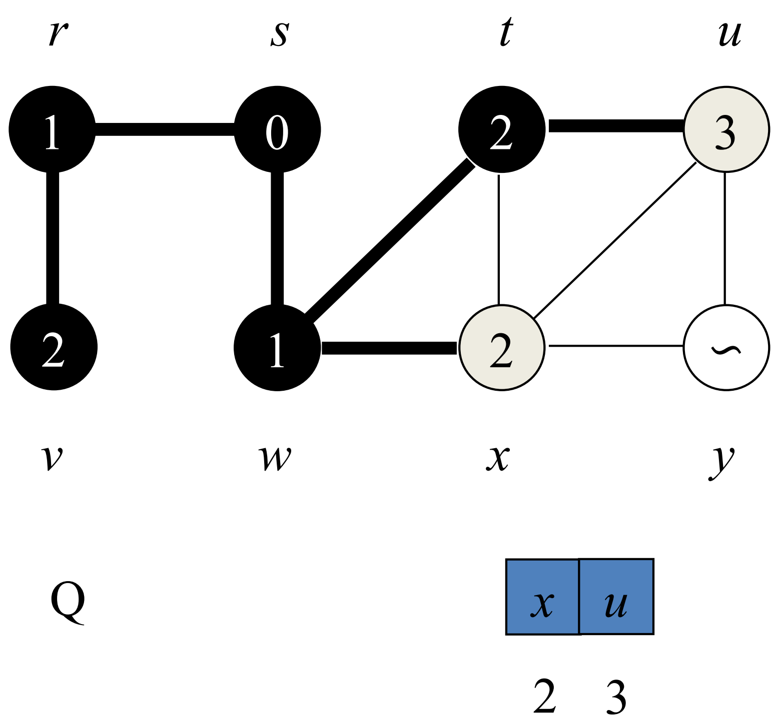


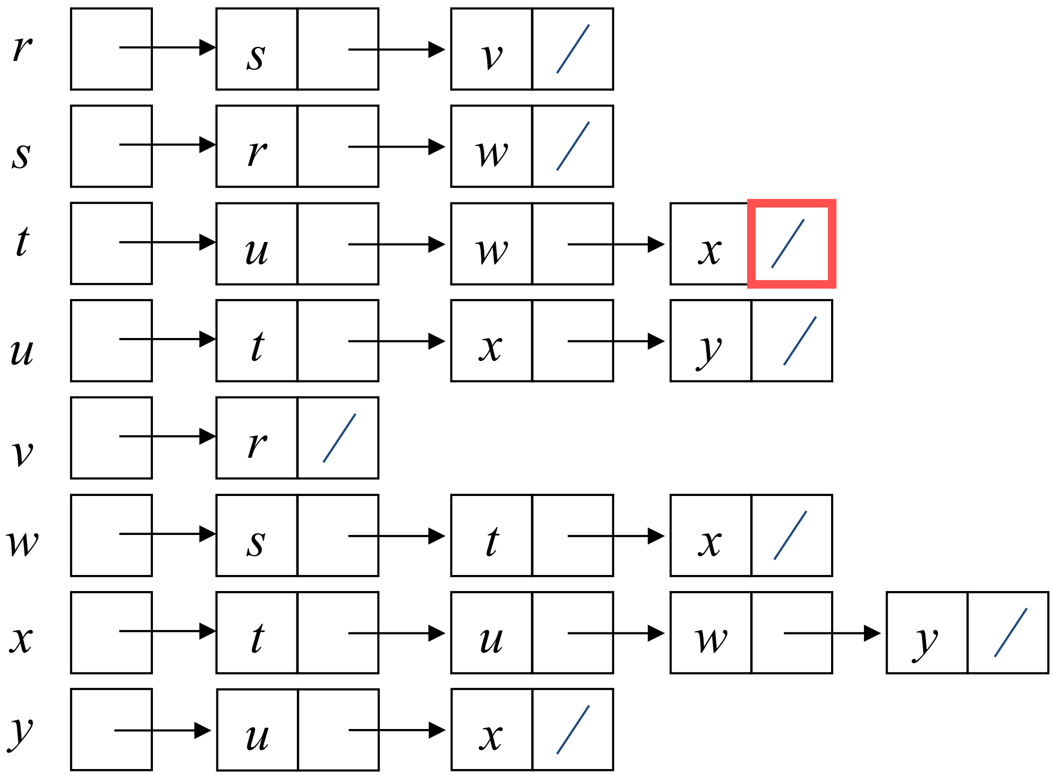
Breadth-first search

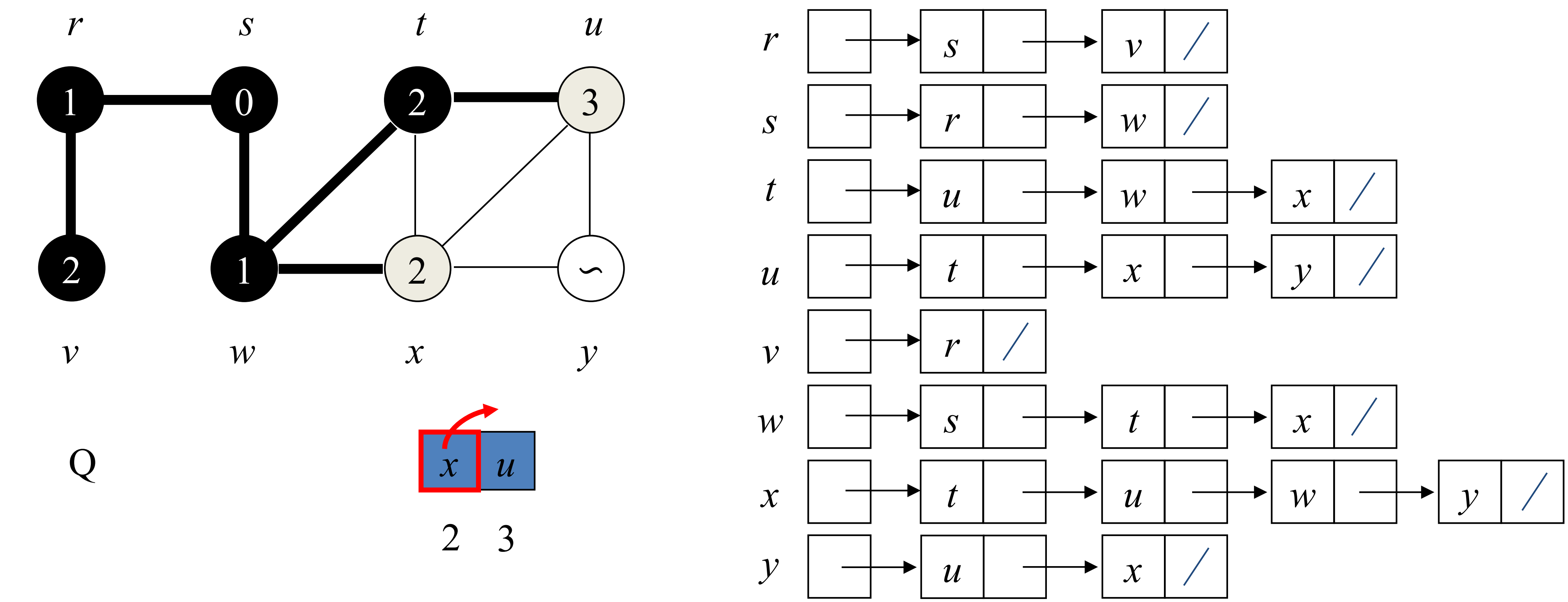


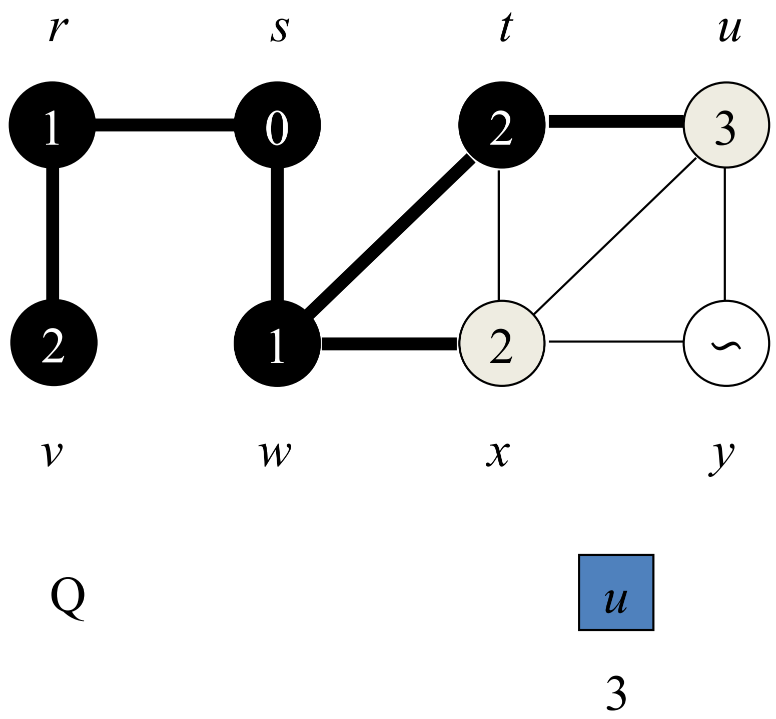
Breadth-first search

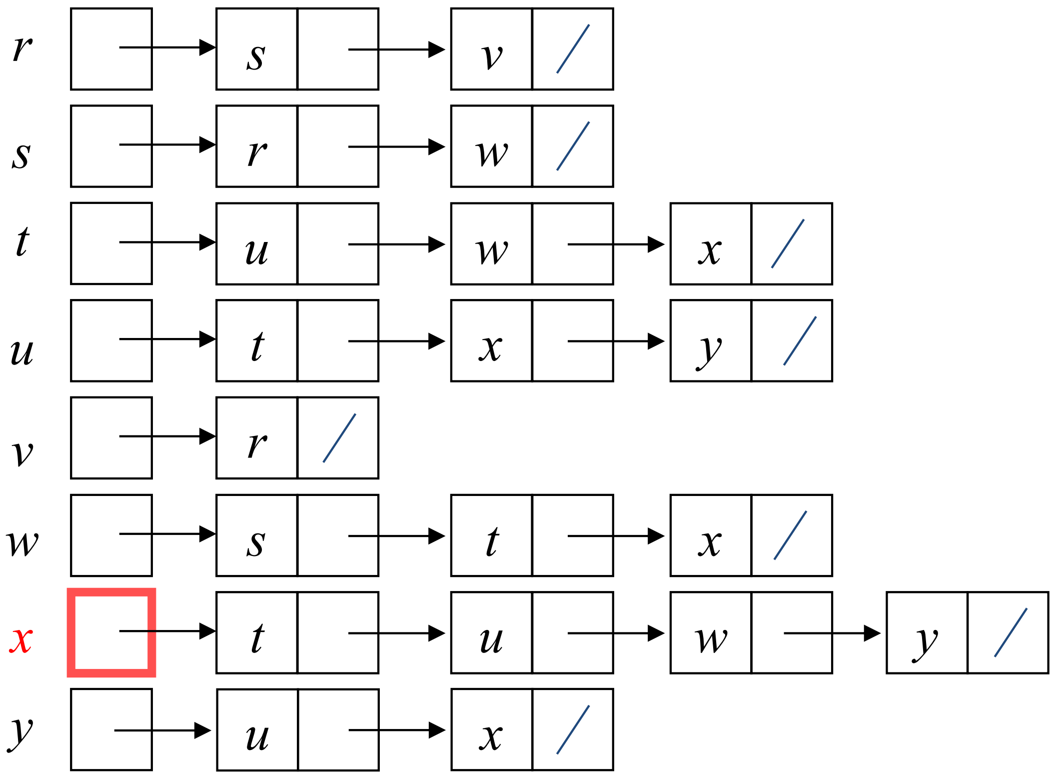


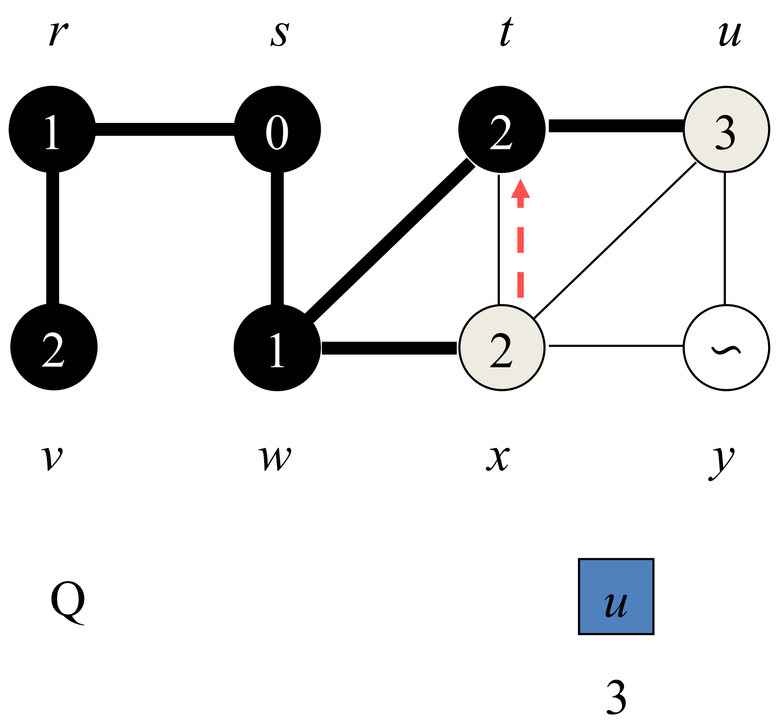

r
s
t
u
1
0
2
3
2
1
2
∽
v
w
x
y
Q
x
u
2
3

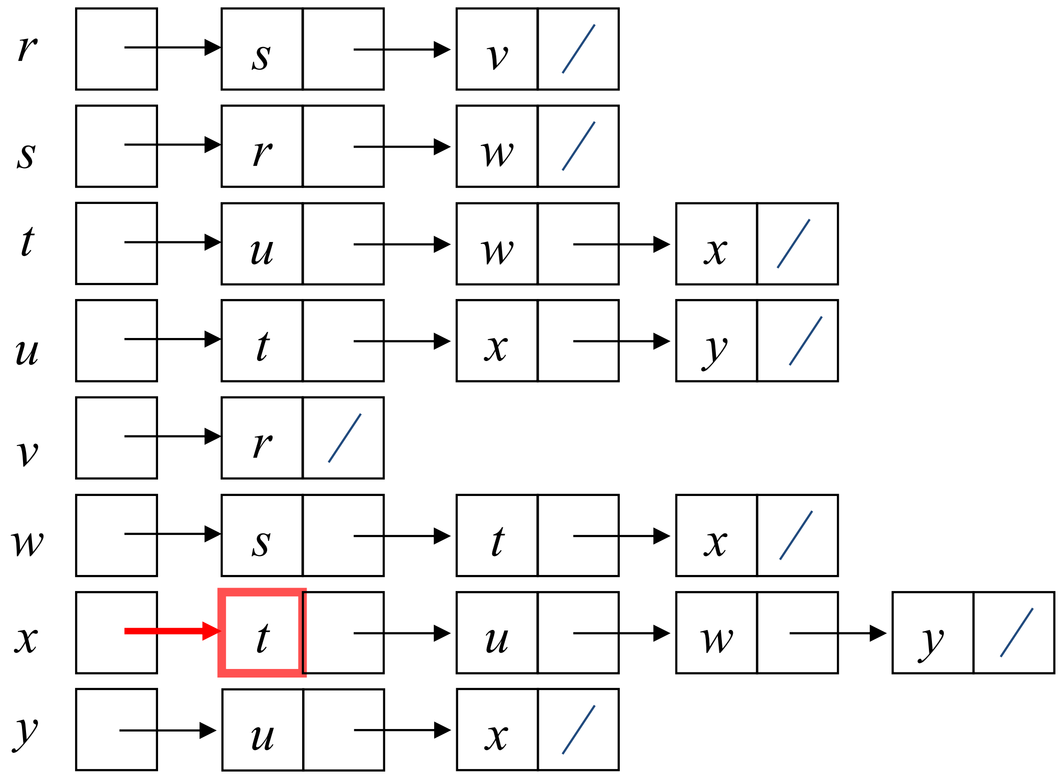

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

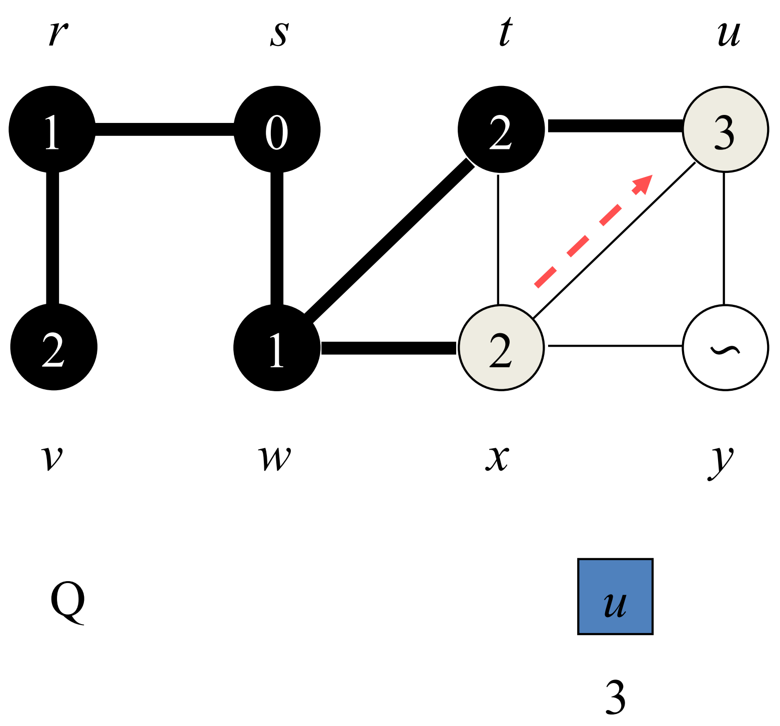

r
s
t
u
1
0
2
3
2
1
2
∽
v
w
x
y
Q
x
u
2
3
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

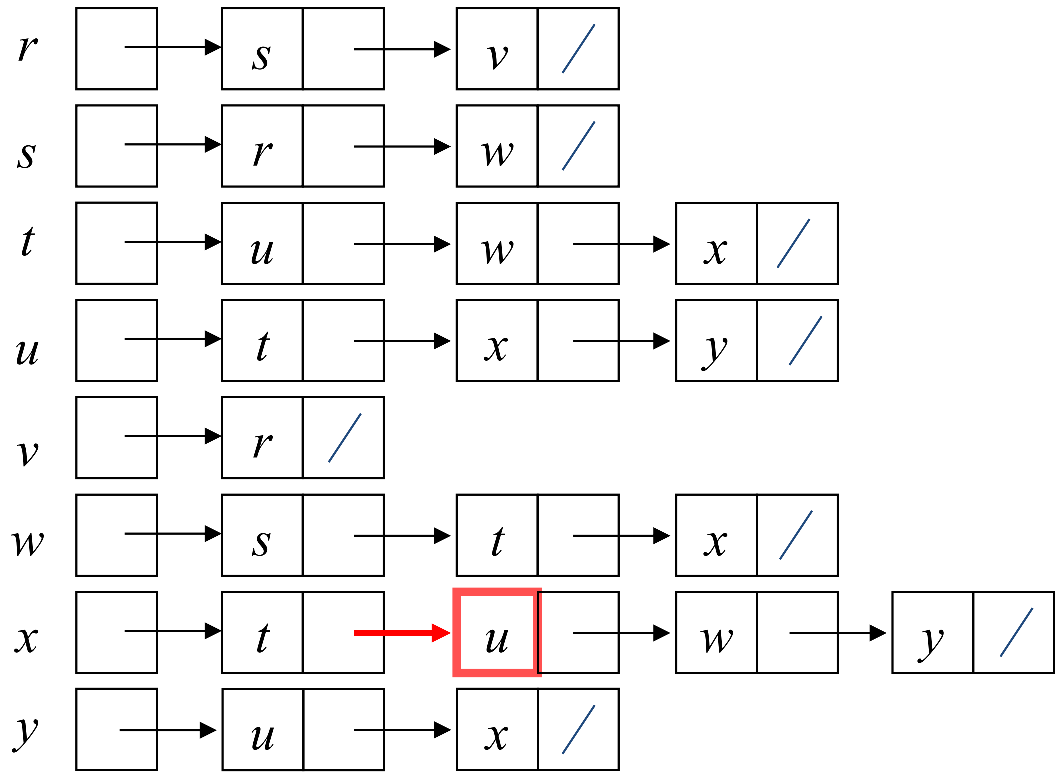

r
s
t
u
1
0
2
3
2
1
2
v
w
x
y
Q
u
3

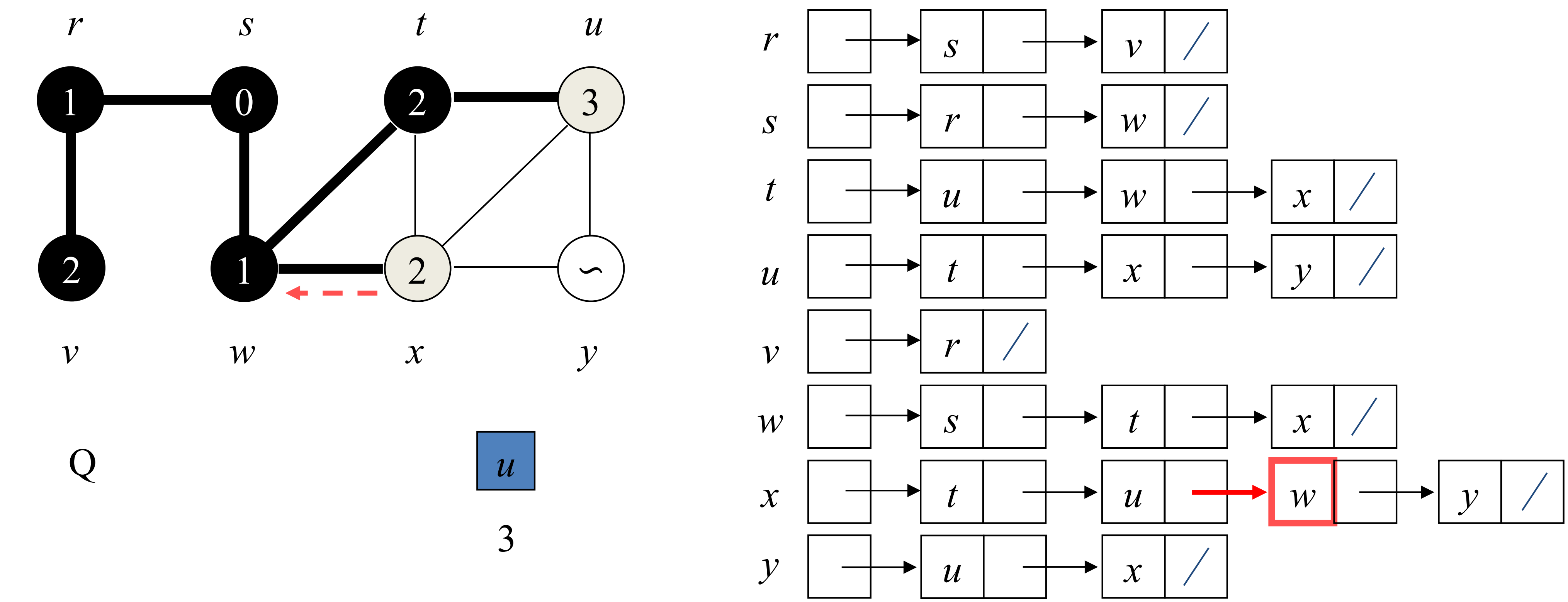

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

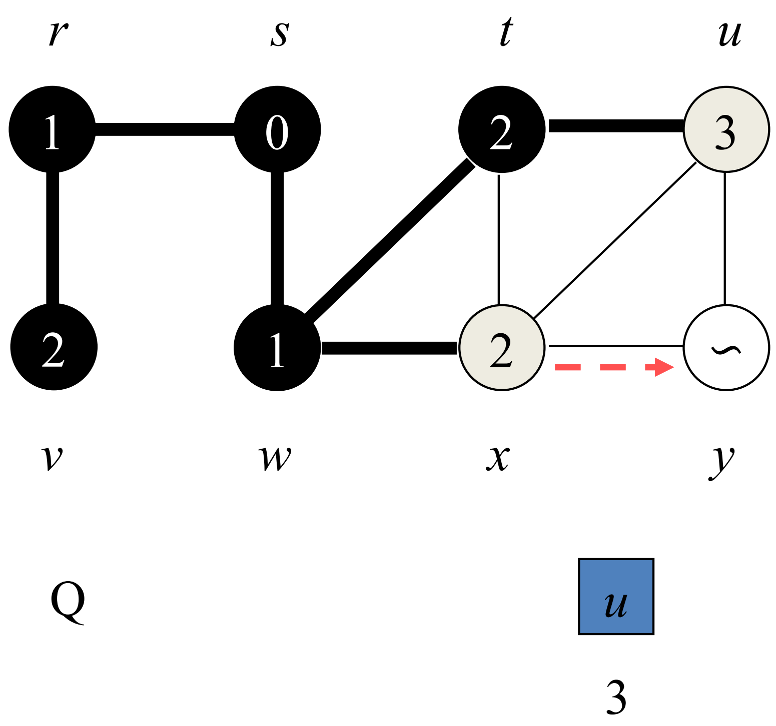

r
s
t
u
1
0
2
3
2
1
2
∽
v
w
x
y
Q
u
3


r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

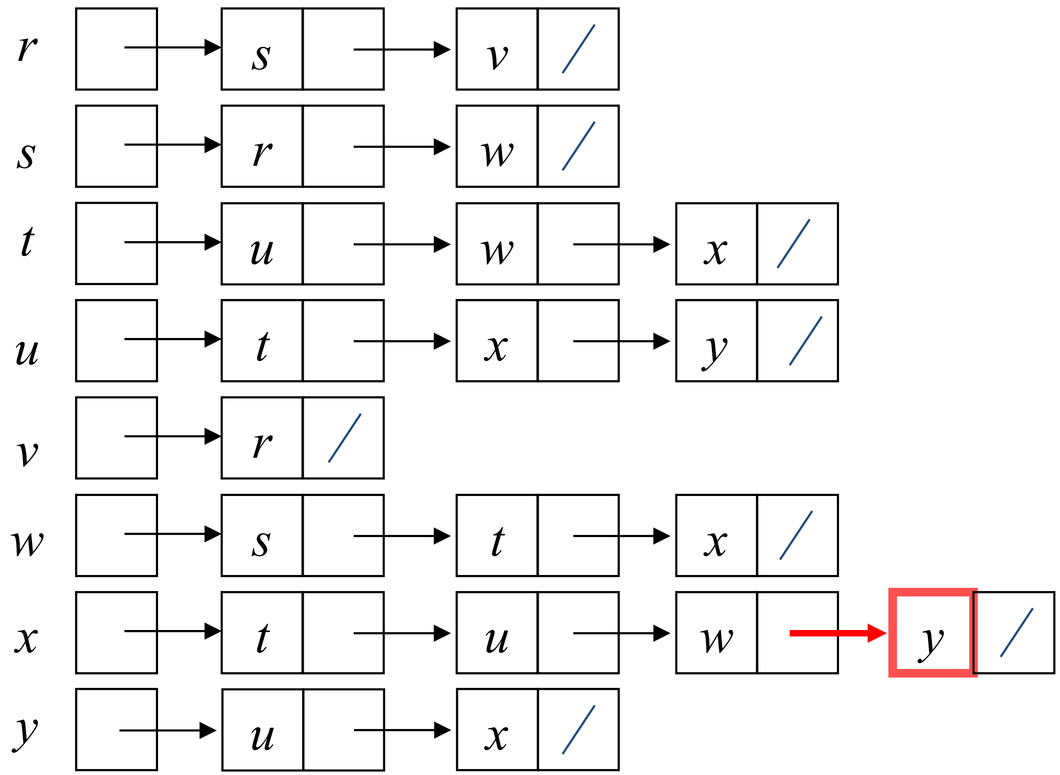

r
s
t
u
1
0
2
3
2
1
2
~
v
w
x
y
Q
u
3


r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

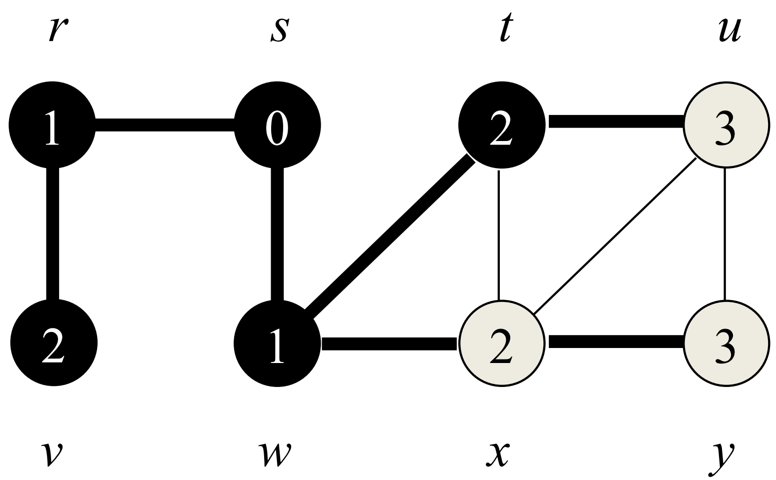

r
s
t
u
1
0
2
3
2
1
2
∽
v
w
x
y
Q
u
3
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

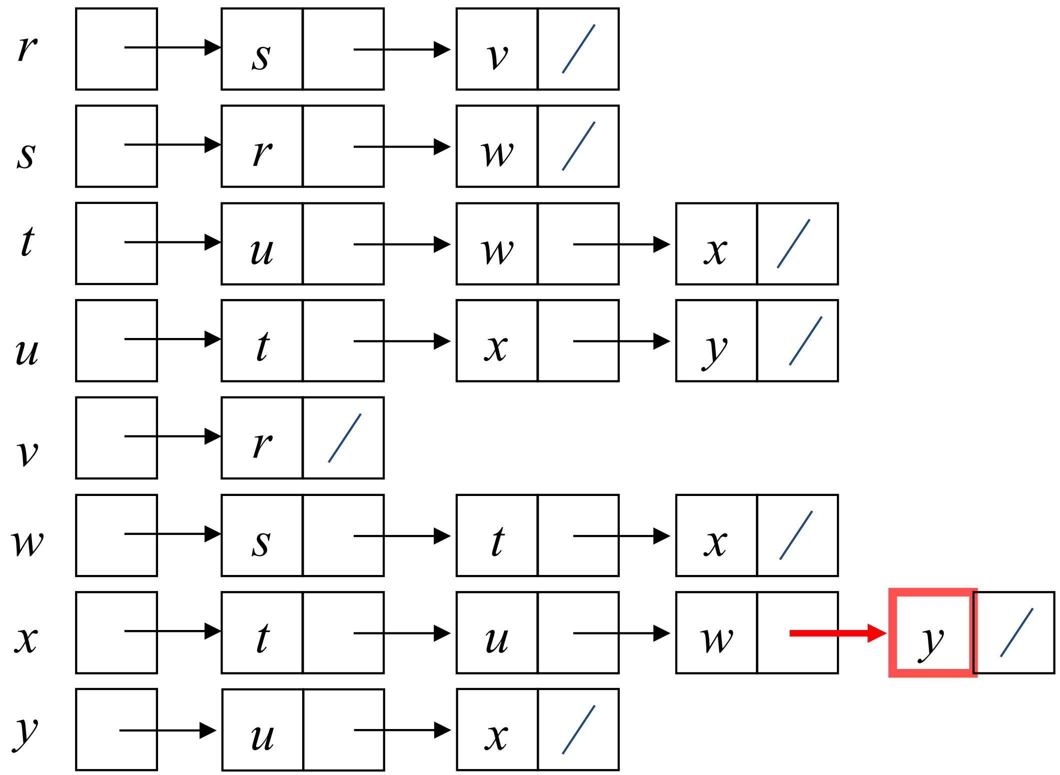


r
s
t
u
1
0
2
3
2
1
2
∽
v
w
x
y
Q
u
3



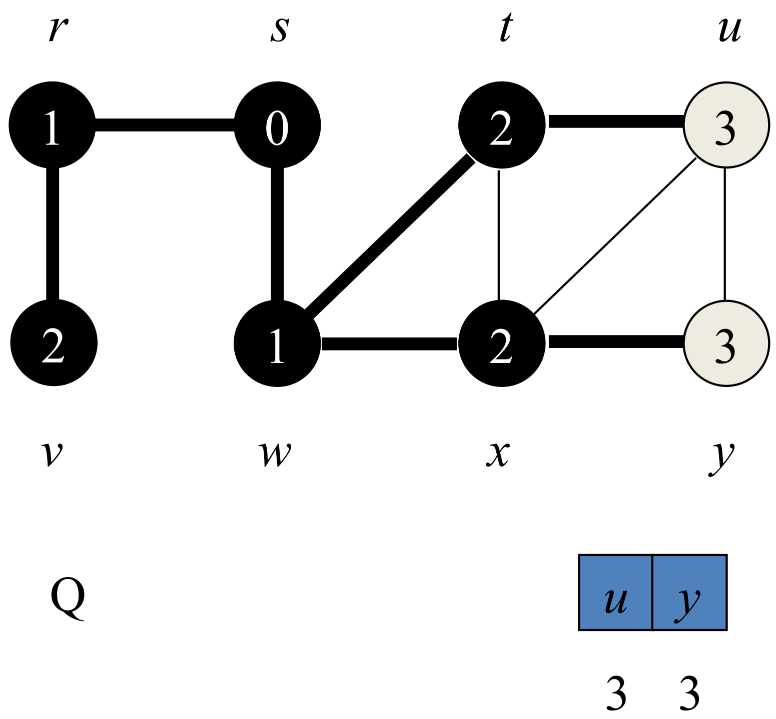
r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

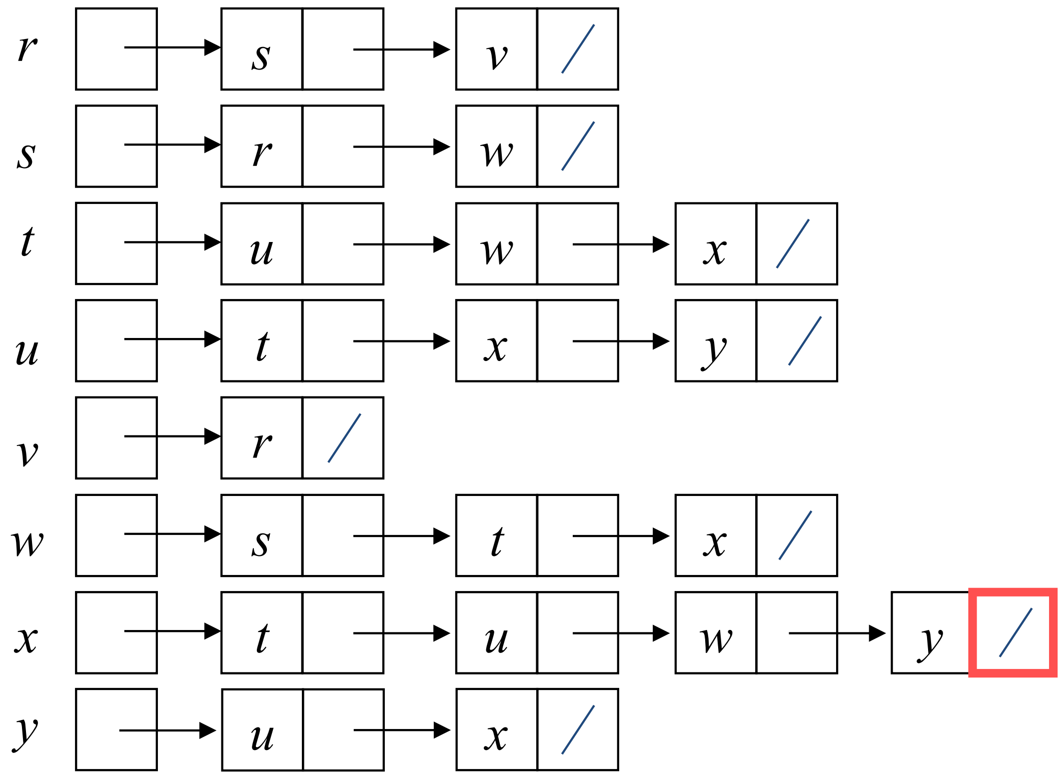


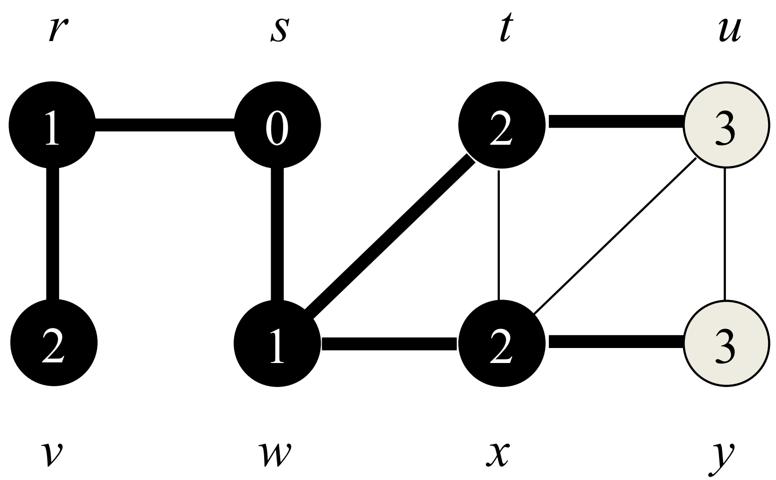


Q

u	y
3	3

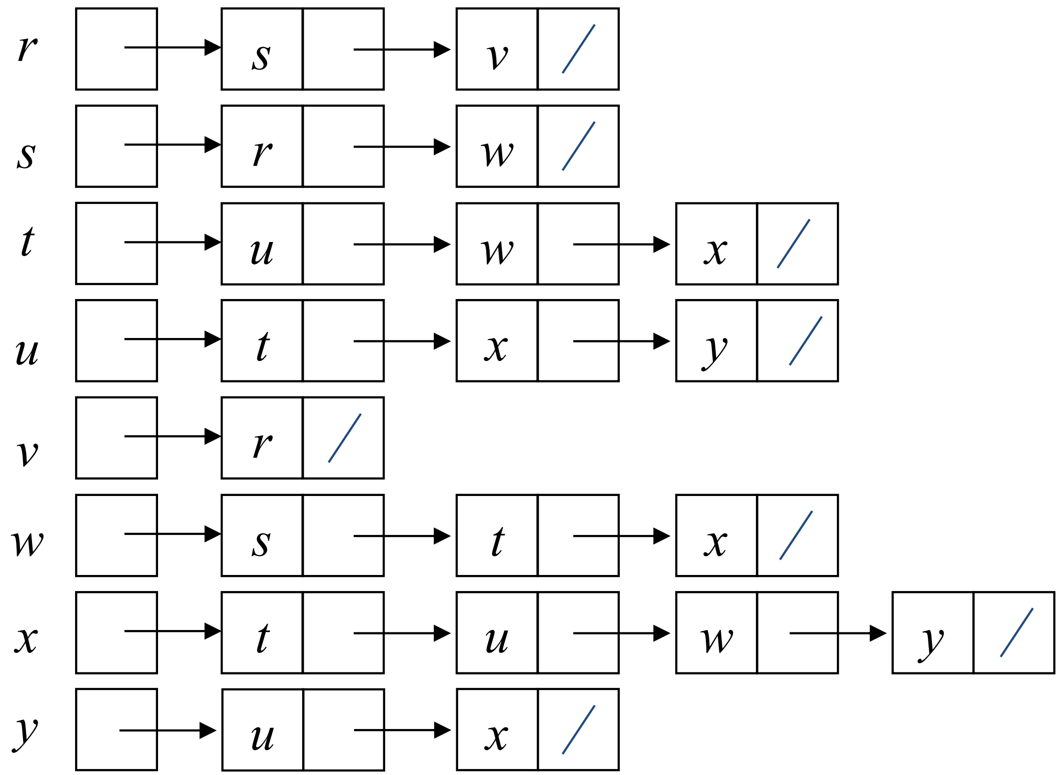

r s t u
1 0 2 3
2 1 2 3
v w x y
Q
u y
3 3

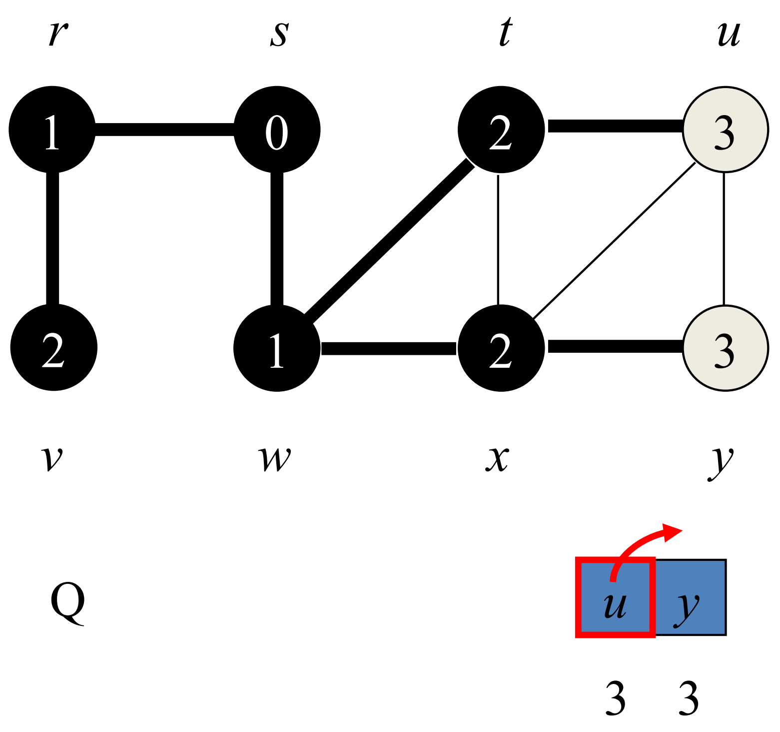

r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

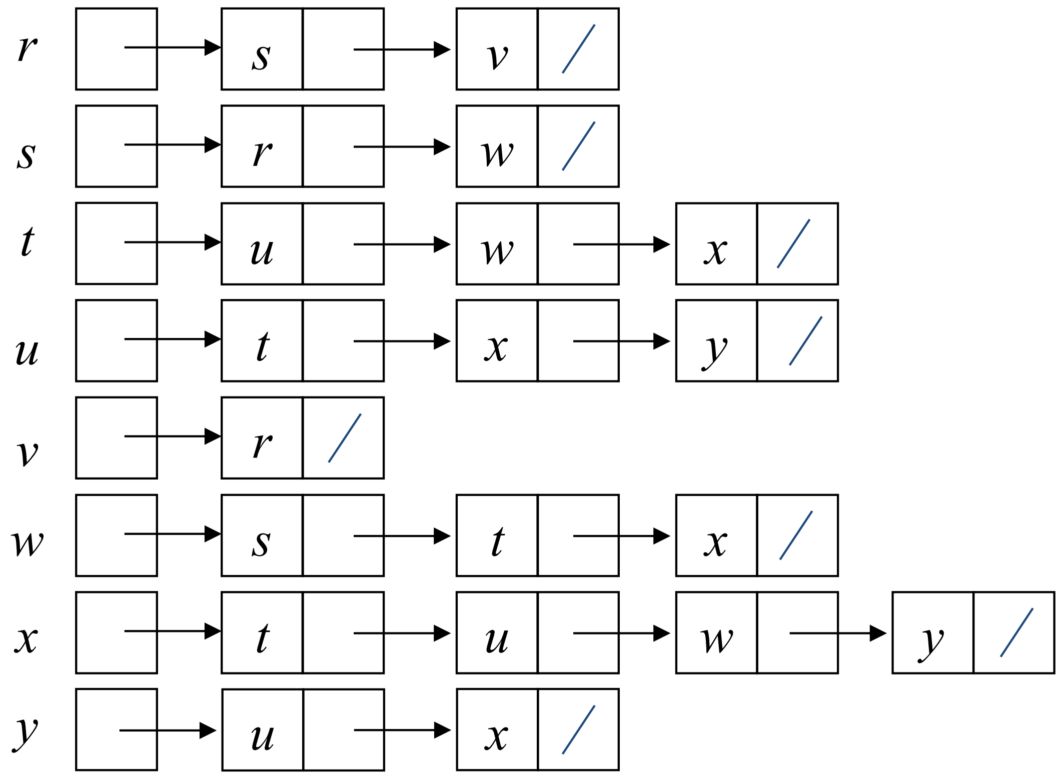


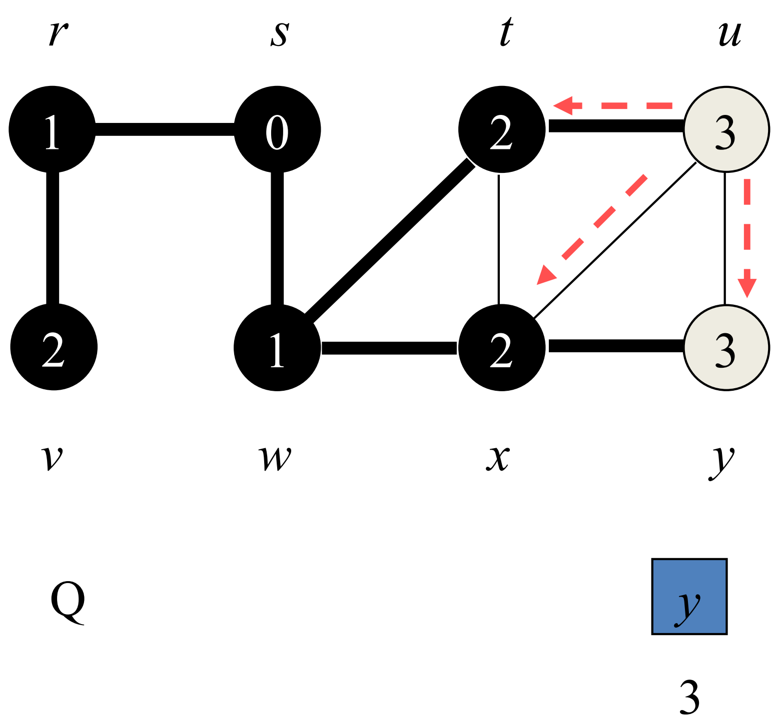
Q

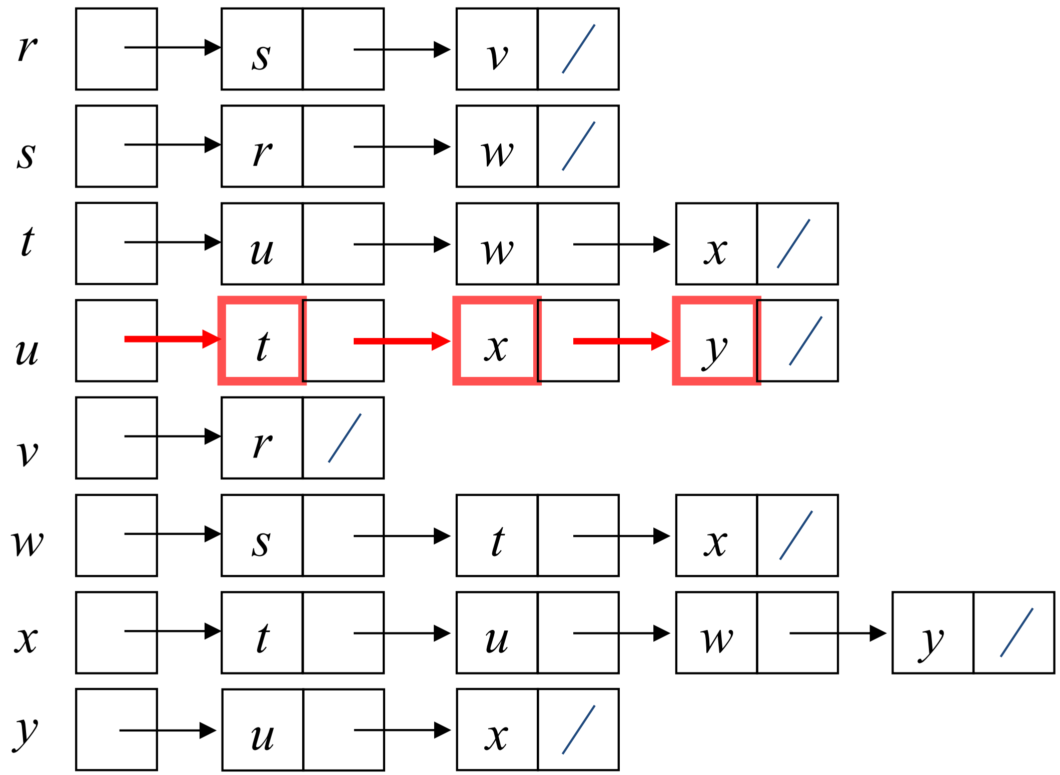
u	y
3	3

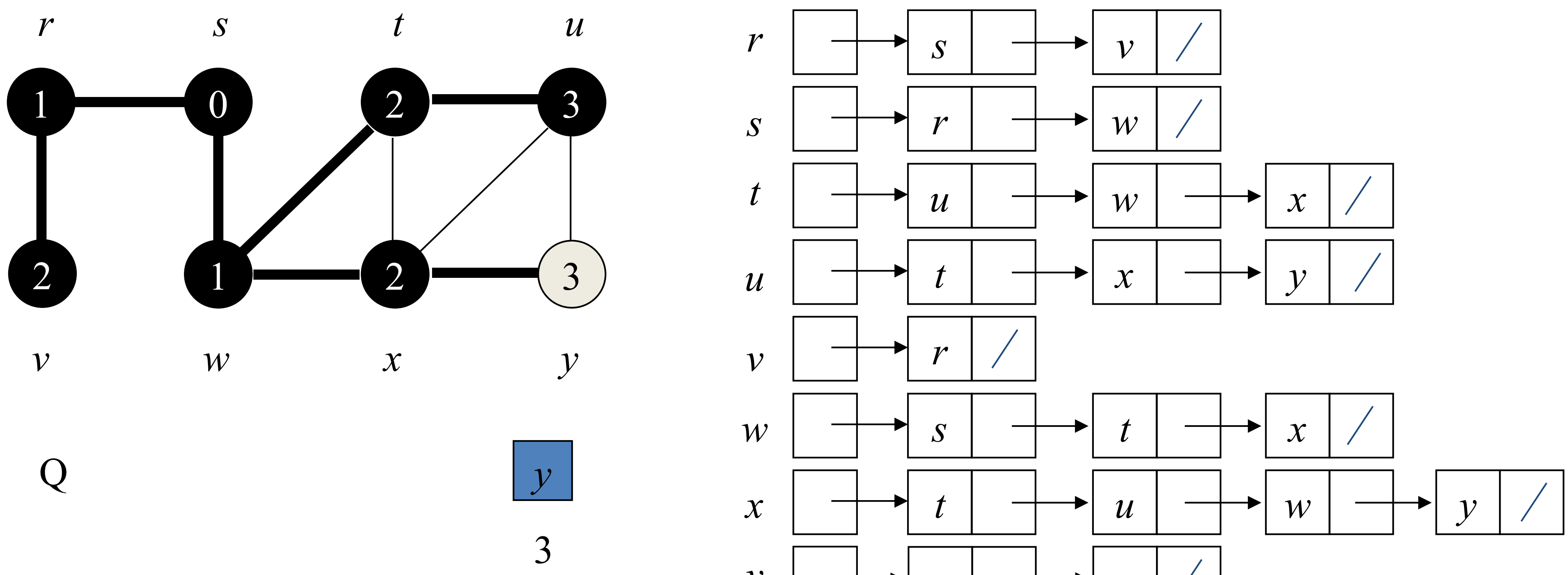


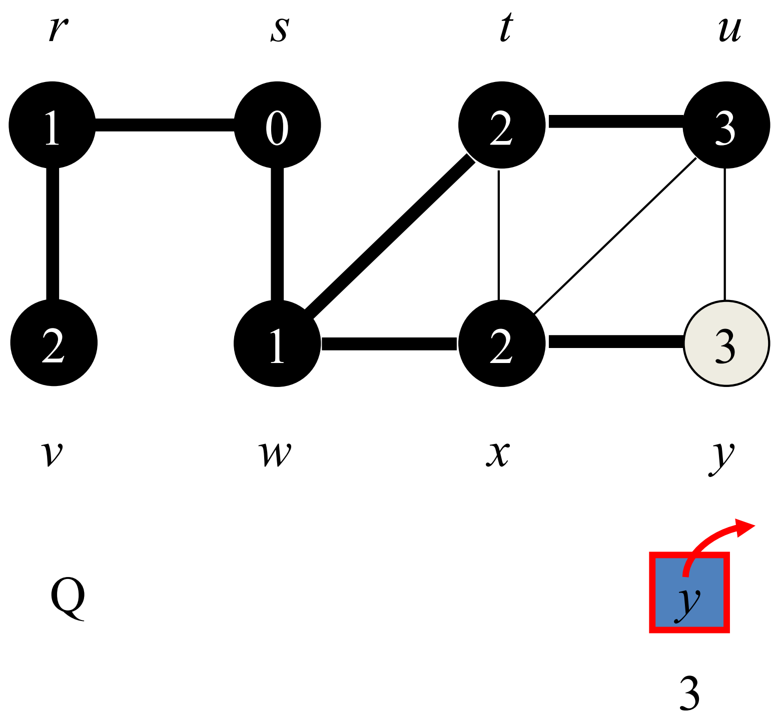

r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
u
y
3
3

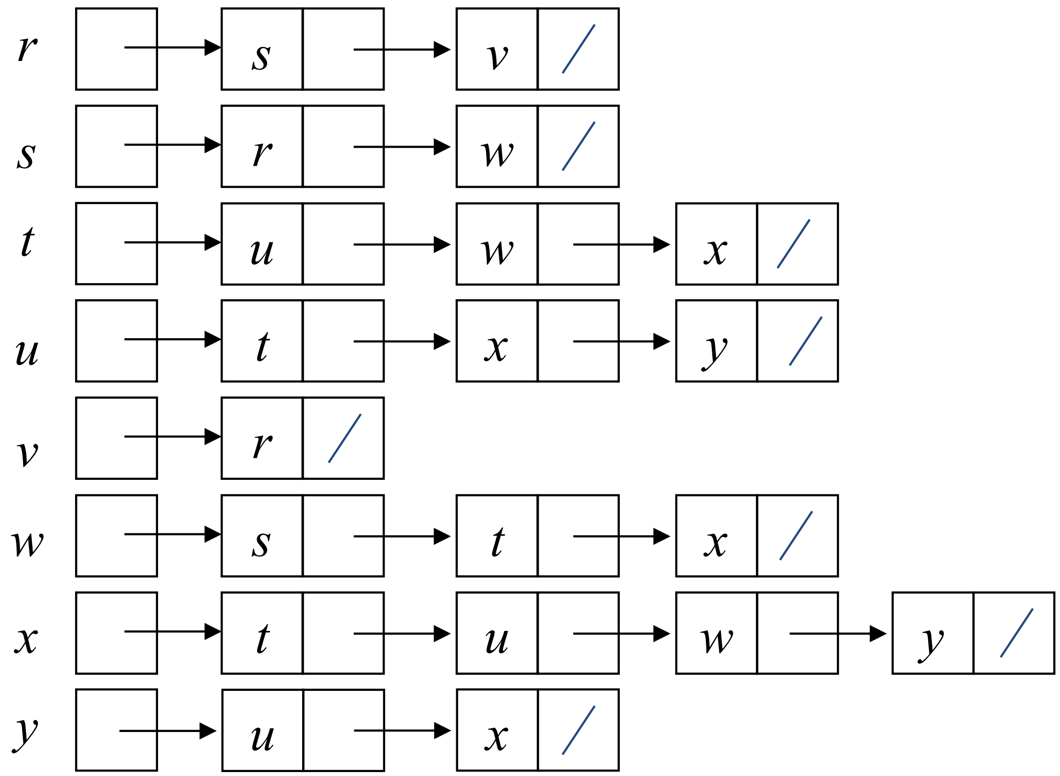

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
y
3

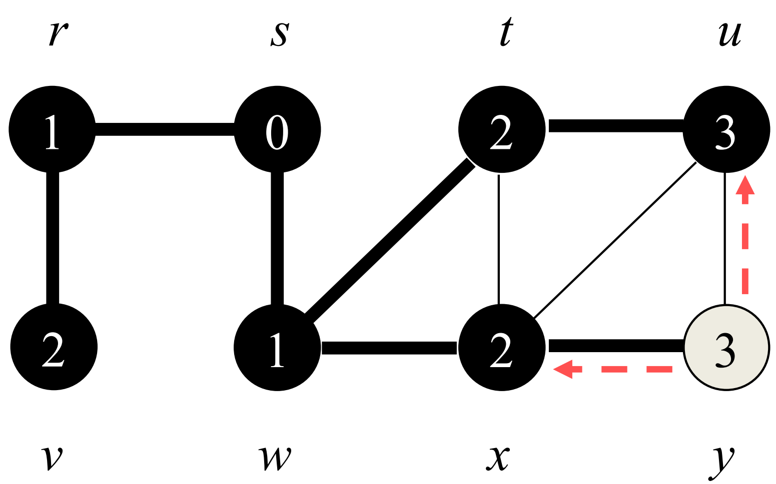

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x


r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
y
3
r s v
s r w
t u w x
u t x y
v r
w s t x
x t u w y
y u x

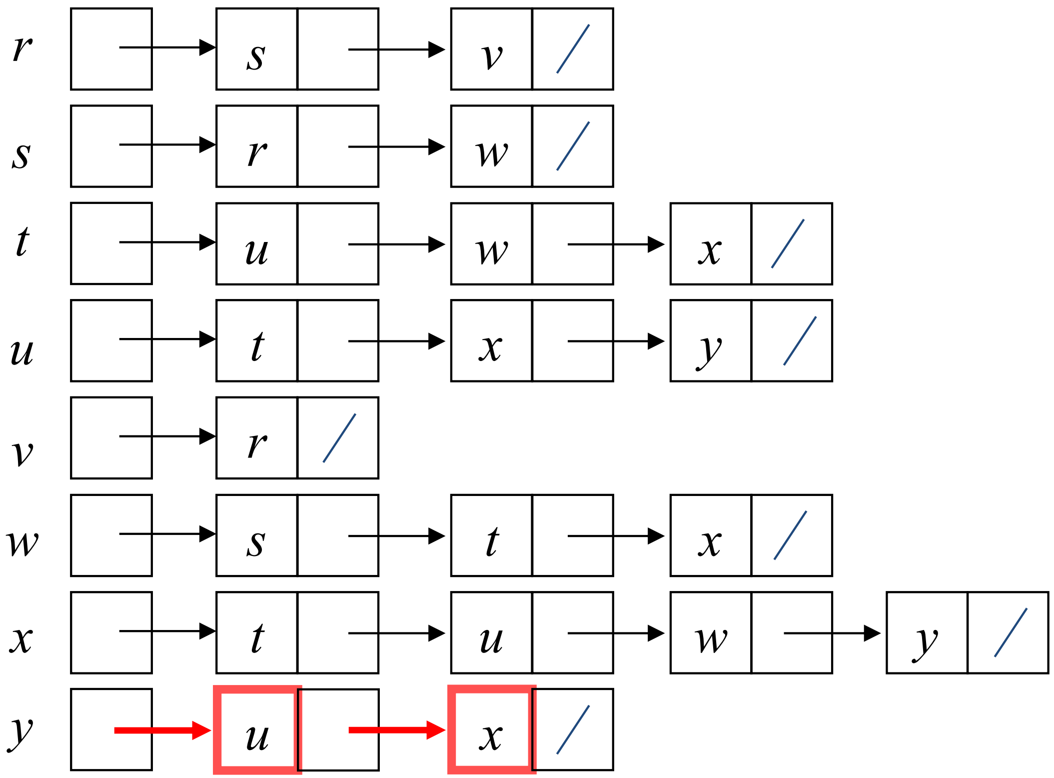

r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q
y
3

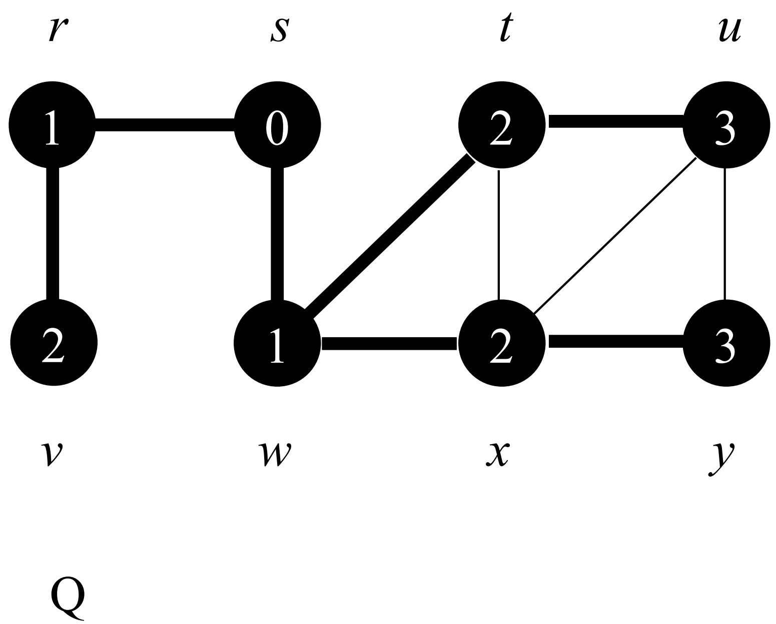

r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

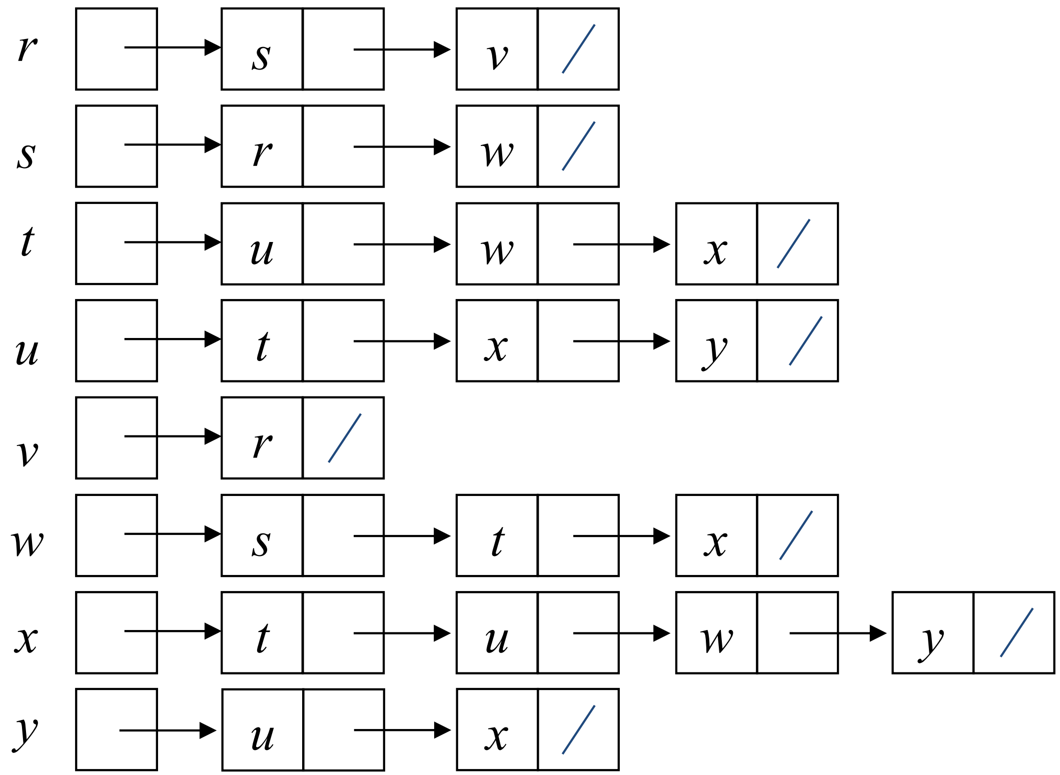
Breadth-first search



Q

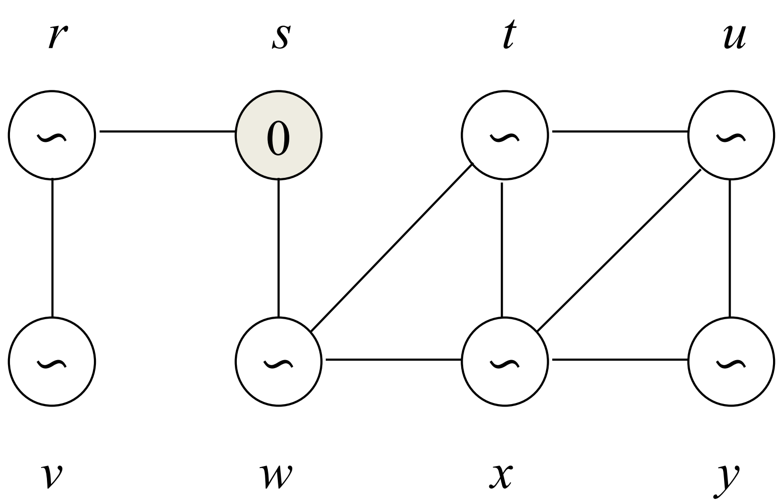


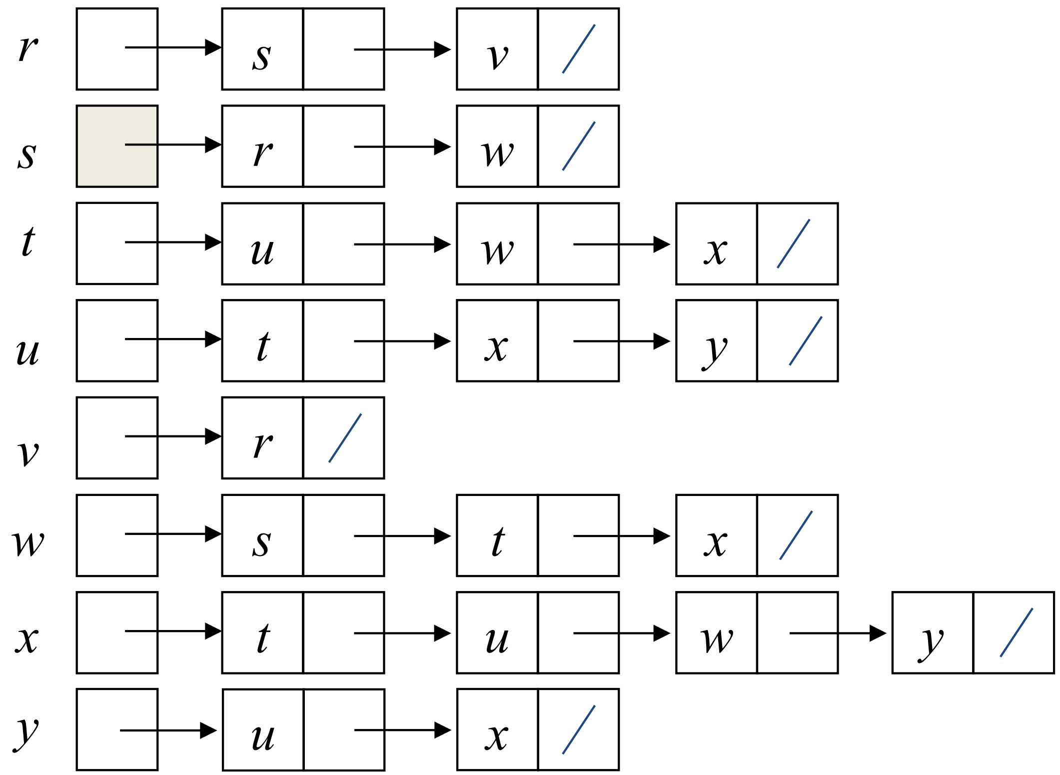

r
s
t
u
1
0
2
3
2
1
2
3
v
w
x
y
Q


r
s
v
s
r
w
t
u
w
x
u
t
x
y
v
r
w
s
t
x
x
t
u
w
y
y
u
x

Breadth-first search

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```





Q s

0

Breadth-first search

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

Initialization (lines 1–9)

$\Theta(V)$

Graph Exploration (lines 10–18)

$O(V + E)$

- **Running time**
  - Initialization: $\Theta(V)$
  - Exploring the graph: $O(V + E)$
    - A vertex is examined at most once.
    - An edge is explored at most twice.
  - Overall: $O(V + E)$

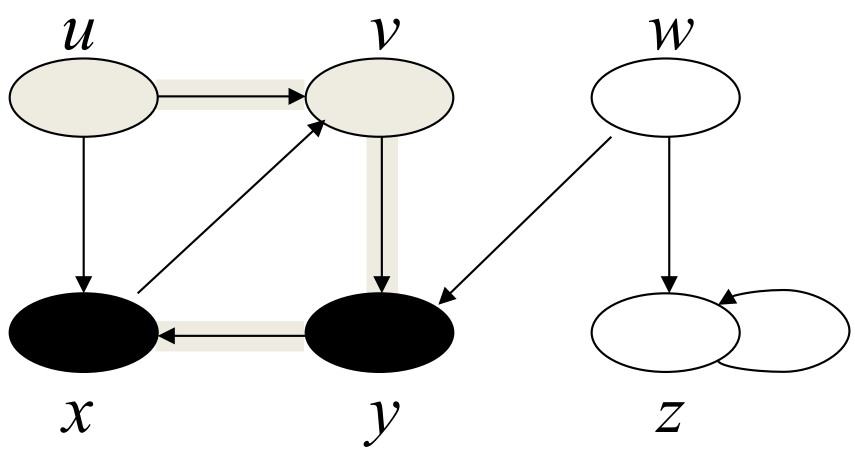
- **Exercise 22.2-4 (22.2-3 in the 2nd ed.)**
  - The running time of BFS
    with adjacency matrix representation.

- **Exercise 22.2-6 (22.2-5 in the 2nd ed.)**
  - Impossible breadth-first trees.

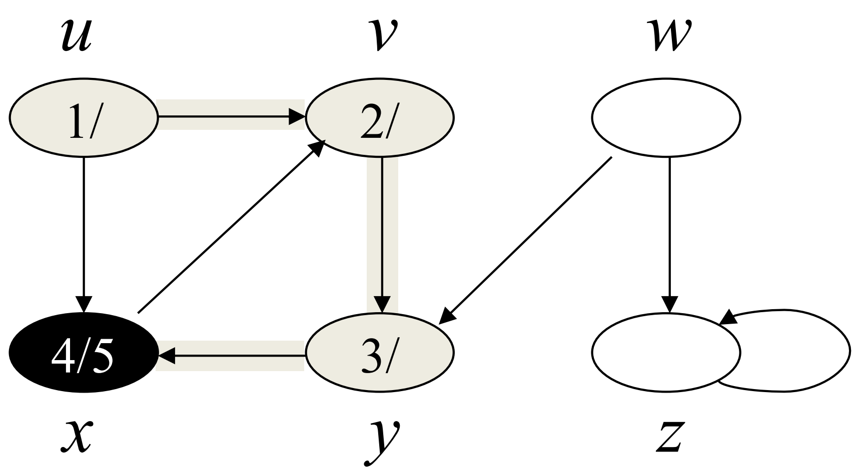
- **Exercise 22.2-7 (22.2-6 in the 2nd ed.)**
  - Rivalry

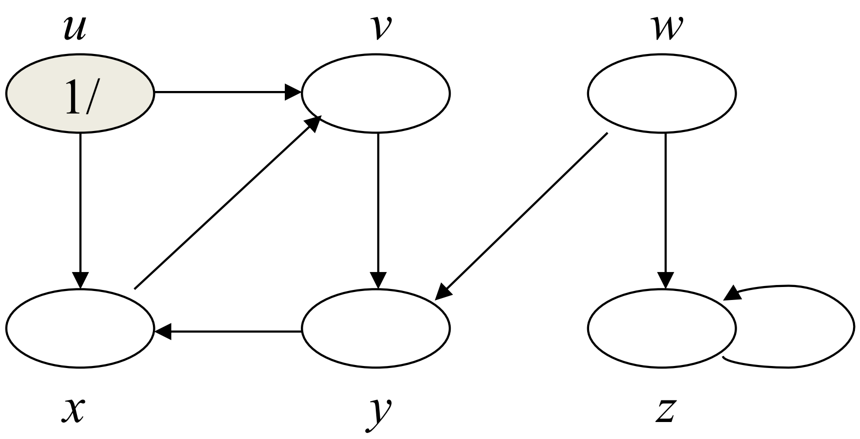
Contents

- ***Colors of vertices***
  - Each vertex is initially ***white***. (not discovered)
  - The vertex is ***grayed*** when it is ***discovered.***
  - The vertex is ***blackened*** when it is ***finished***, that is, when its adjacency list has been examined completely.

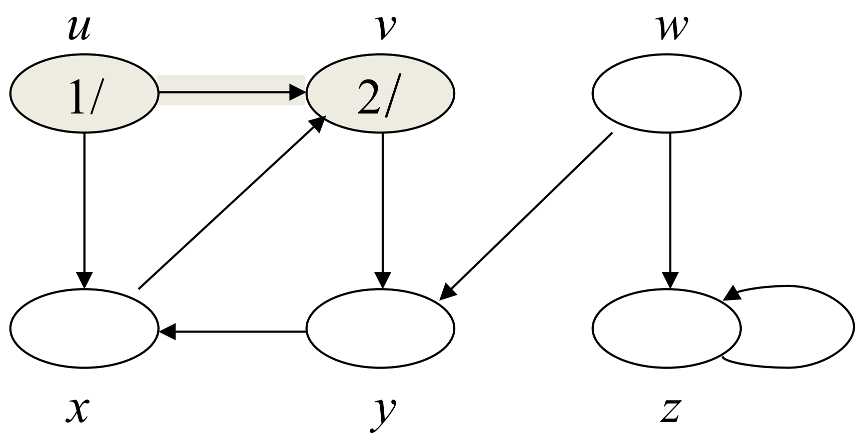


- ***Timestamps***
  - Each vertex v has two timestamps.
    - $v.d$: ***discovery time*** (when v is grayed)
    - $v.f$: ***finishing time*** (when v is blacken)

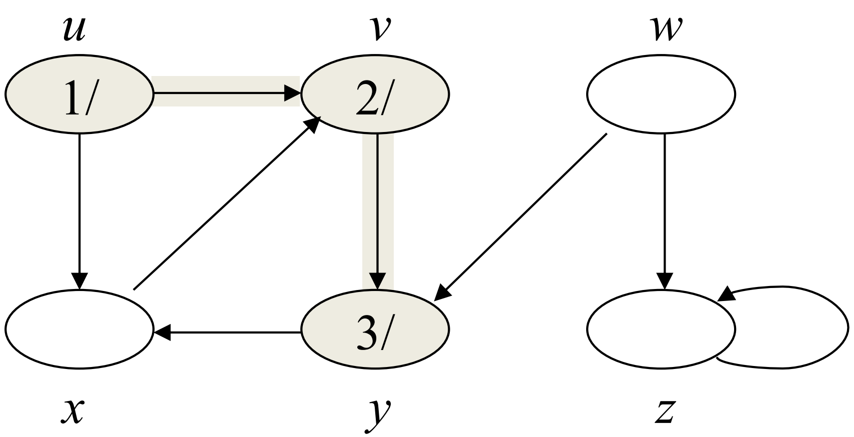




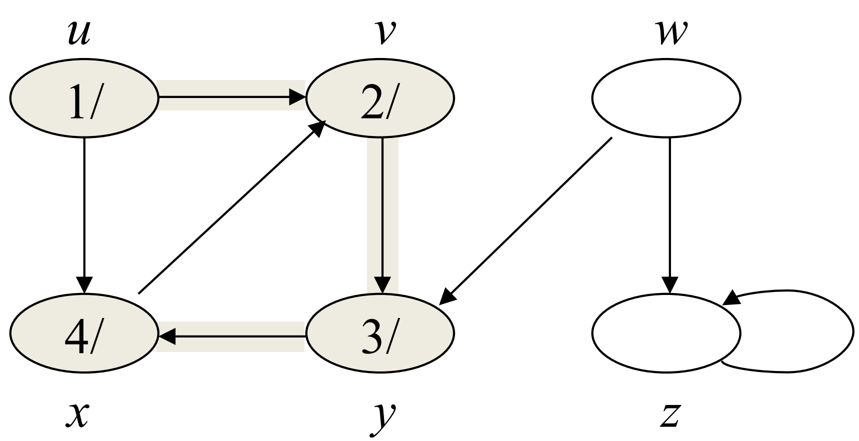
(a)



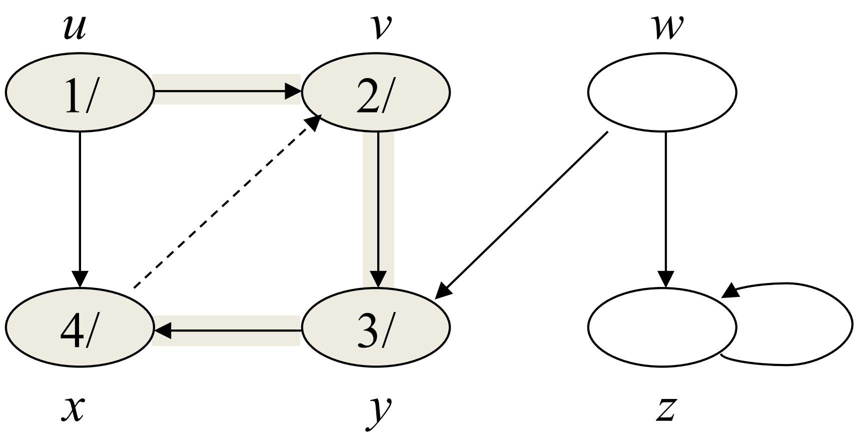
(b)



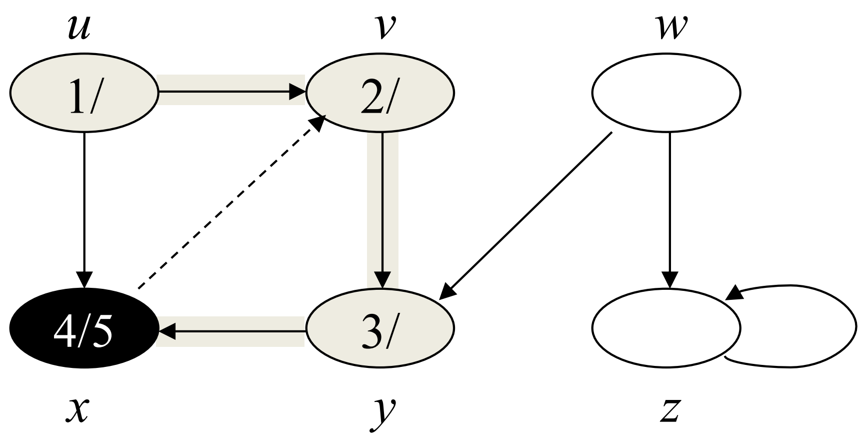
(c)



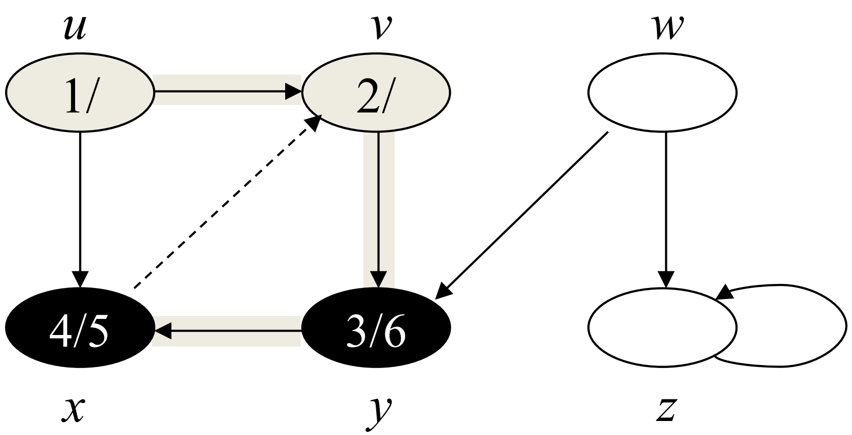
(d)



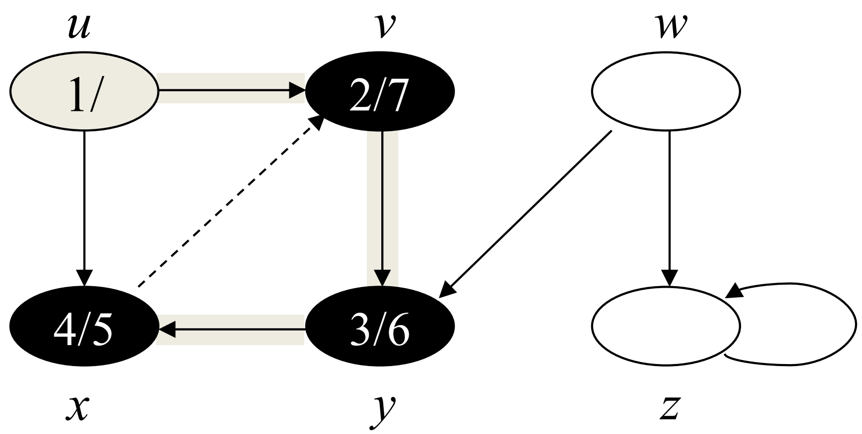
(e)



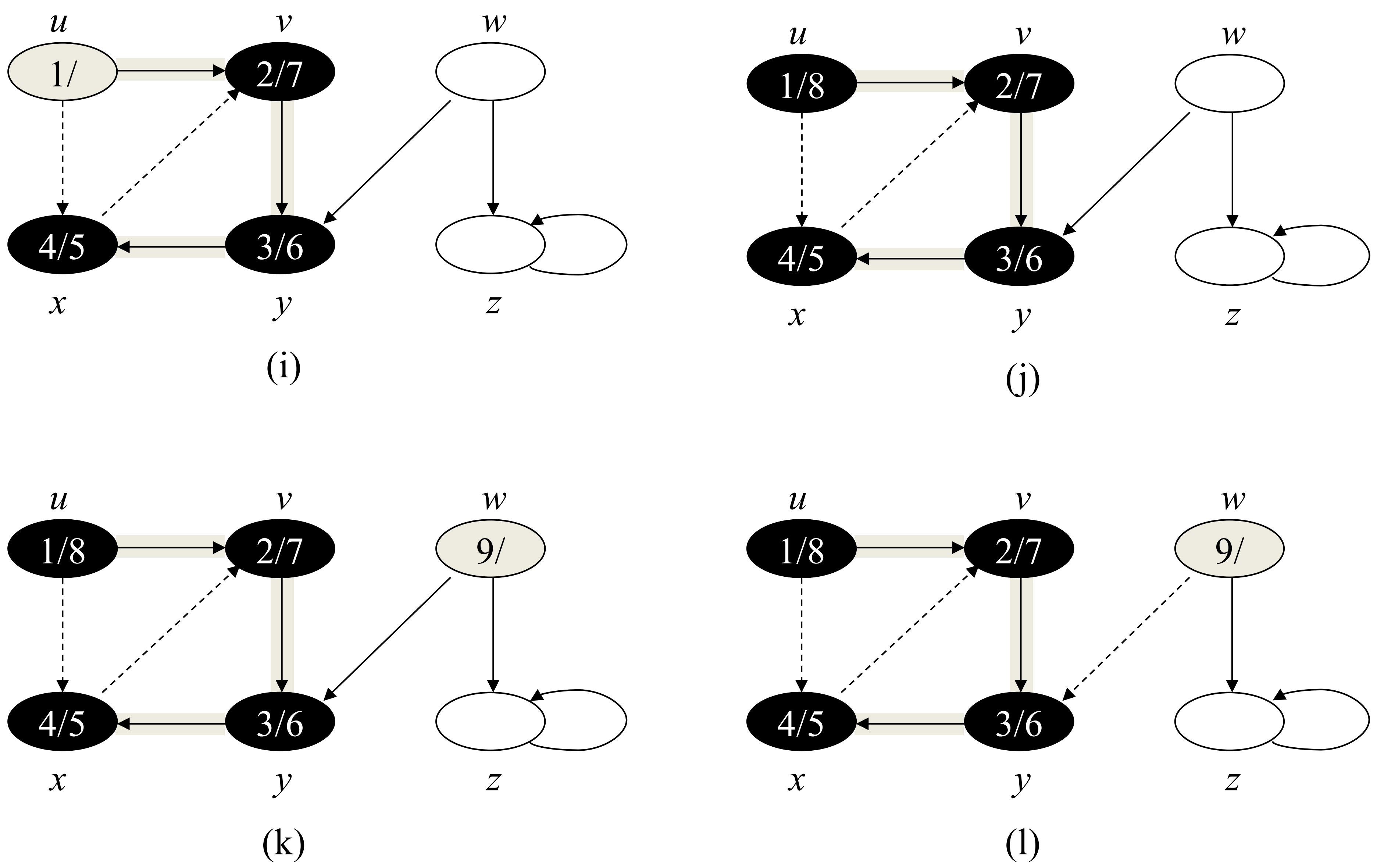
(f)

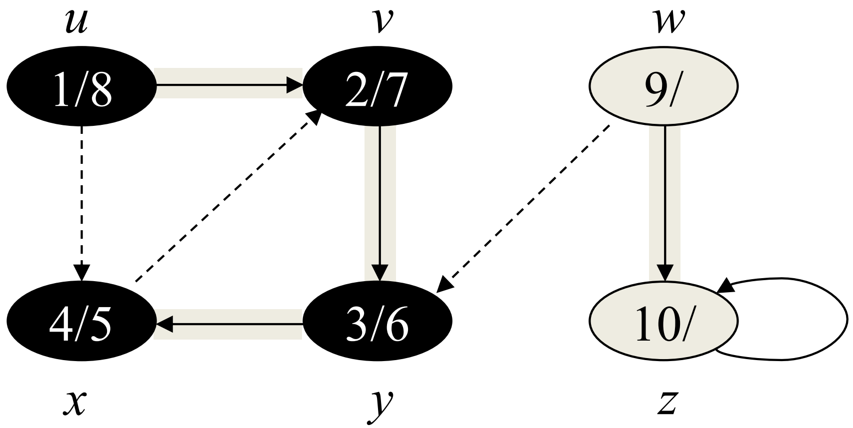


(g)

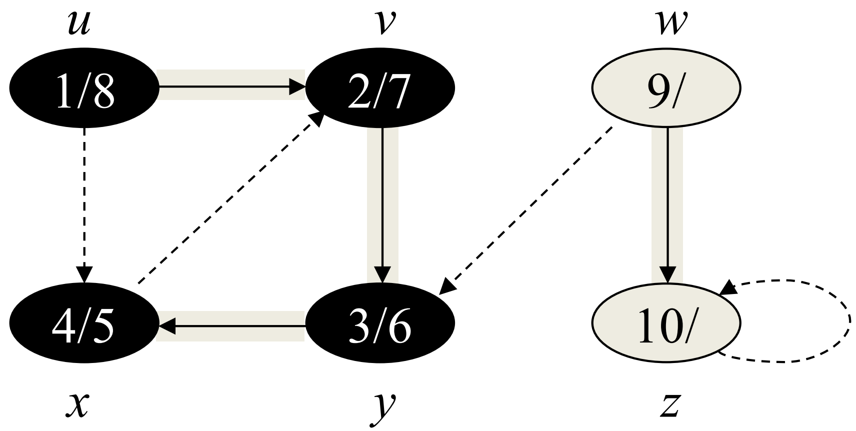


(h)

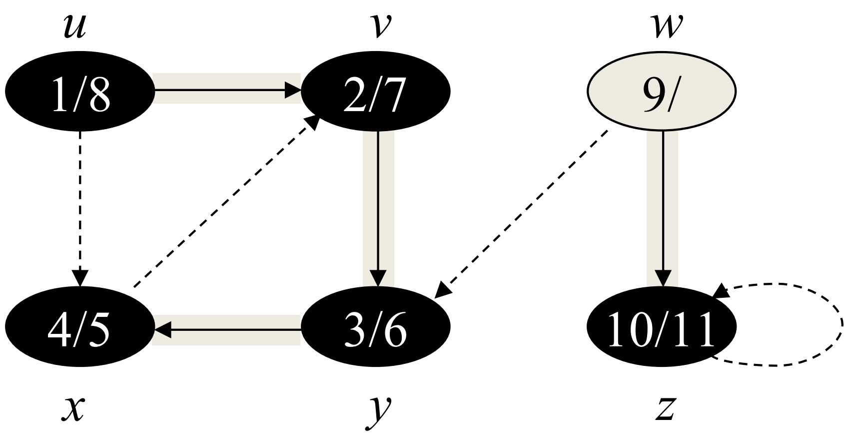

u
v
w
1/
2/7
4/5
3/6
x
y
z
(i)
u
v
w
1/8
2/7
4/5
3/6
x
y
z
(j)
u
v
w
1/8
2/7
9/
4/5
3/6
x
y
z
(k)
u
v
w
1/8
2/7
9/
4/5
3/6
x
y
z
(l)


u
v
w
1/8
2/7
9/
4/5
3/6
10/
x
y
z

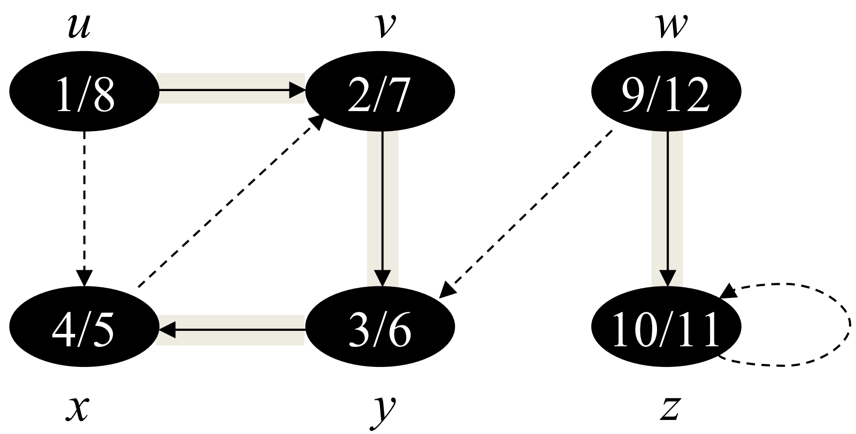
(m)


u
v
w
1/8
2/7
9/
4/5
3/6
10/
x
y
z

(n)


u
v
w
1/8
2/7
9/
4/5
3/6
10/11
x
y
z

(o)


u
v
w
1/8
2/7
9/12
4/5
3/6
10/11
x
y
z

(p)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G,u)

DFS-VISIT(G,u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G,v)
u.color = BLACK
time = time + 1
u.f = time
```

```
DFS(G)
    for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
    time = 0
    for each vertex u ∈ G.V
        if u.color == WHITE
            DFS-VISIT(G,u)

DFS-VISIT(G,u)
    time = time + 1
    u.d = time
    u.color = GRAY
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.π = u
            DFS-VISIT(G,v)
    u.color = BLACK
    time = time + 1
    u.f = time
```

Initialization

$\Theta(V)$

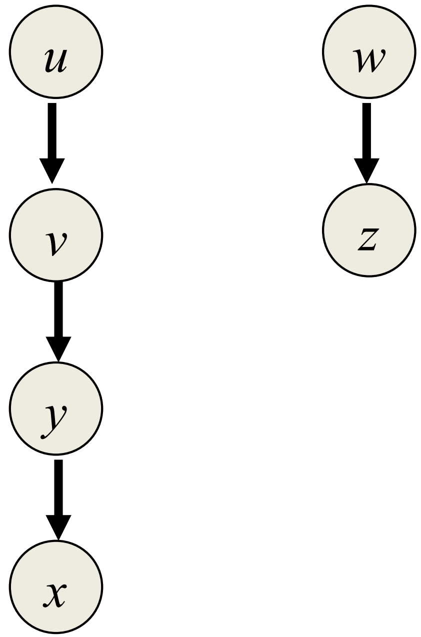
Graph Exploration

$\Theta(V + E)$

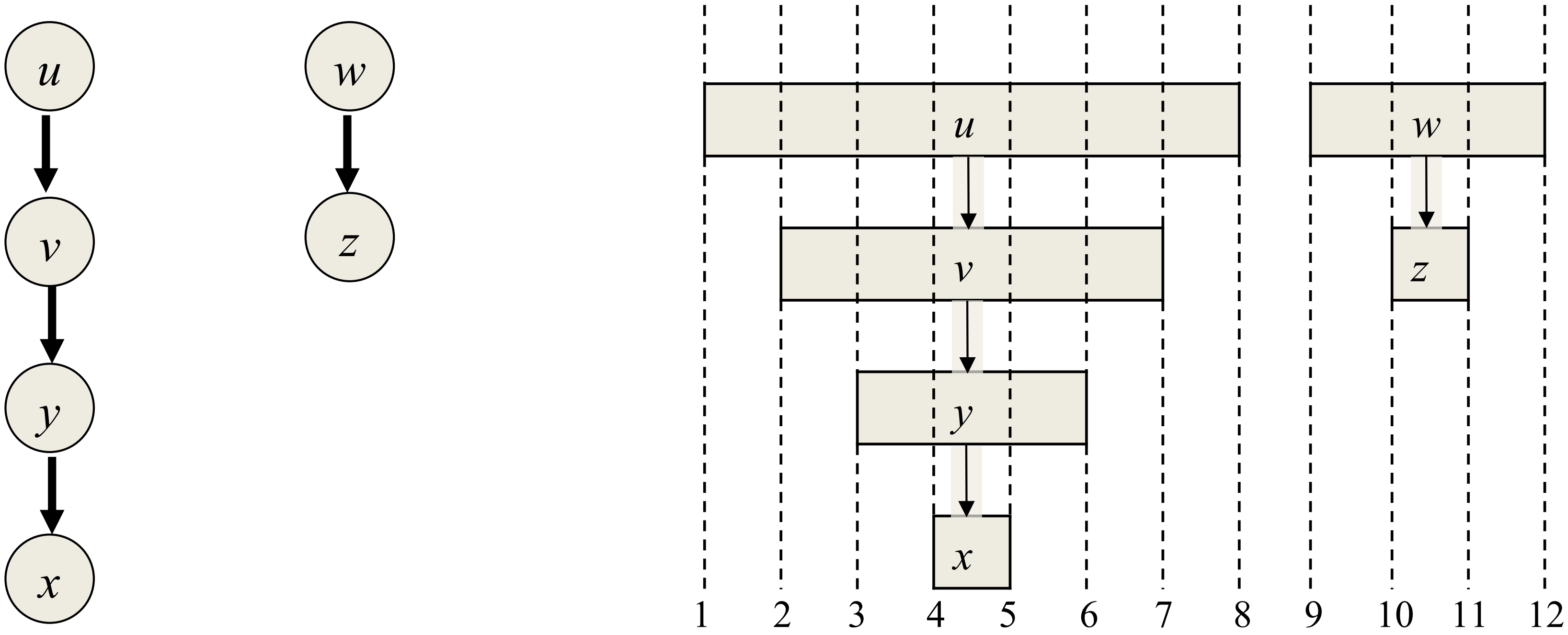
DFS-VISIT

$\Theta(E)$

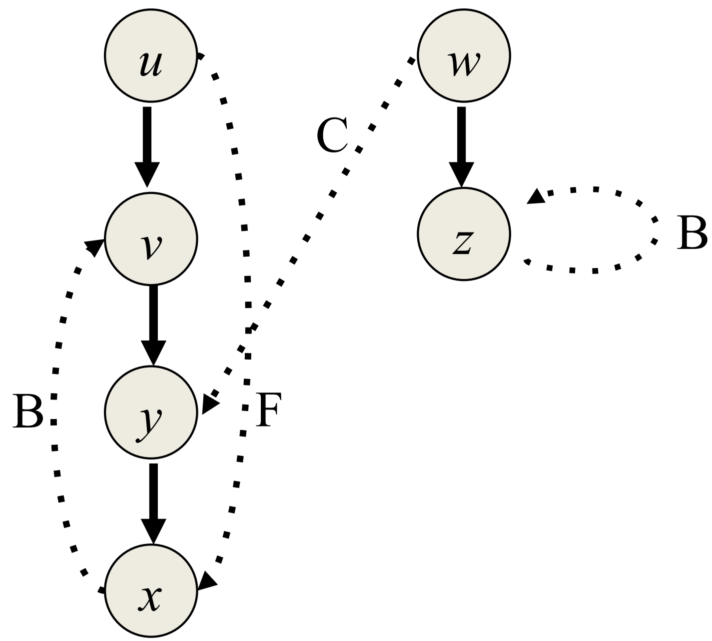
- The ***predecessor subgraph*** is a ***depth-first forest***.



- ***Parenthesis theorem* (*for gray interval*)**
  - ***Inclusion***: The ancestor's includes the descendants'.
  - ***Disjoint***: Otherwise.



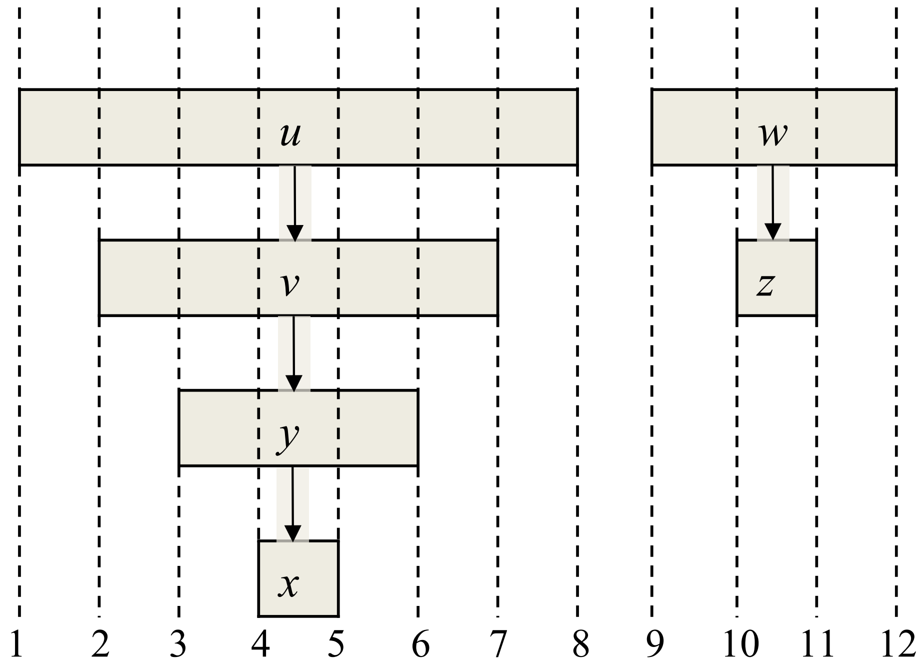
- **Classification of edges**
  - ***Tree edges***
  - ***Back edges***
  - ***Forward edges***
  - ***Cross edges***

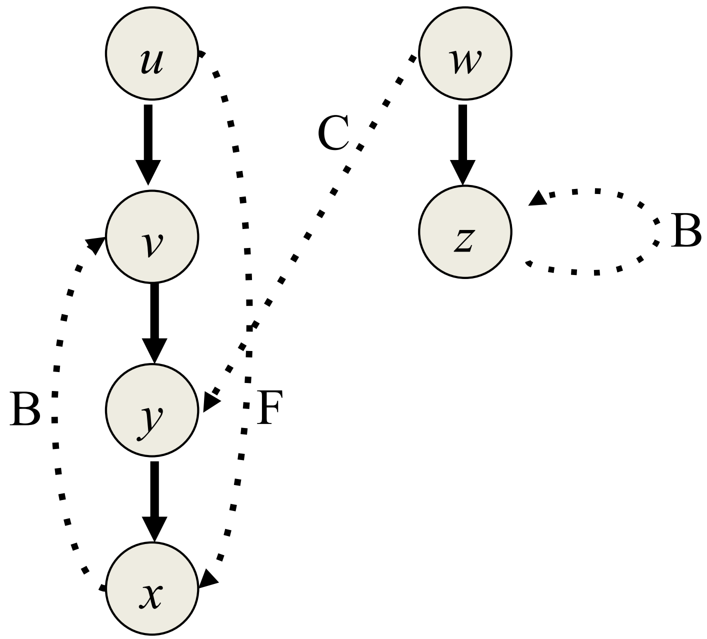


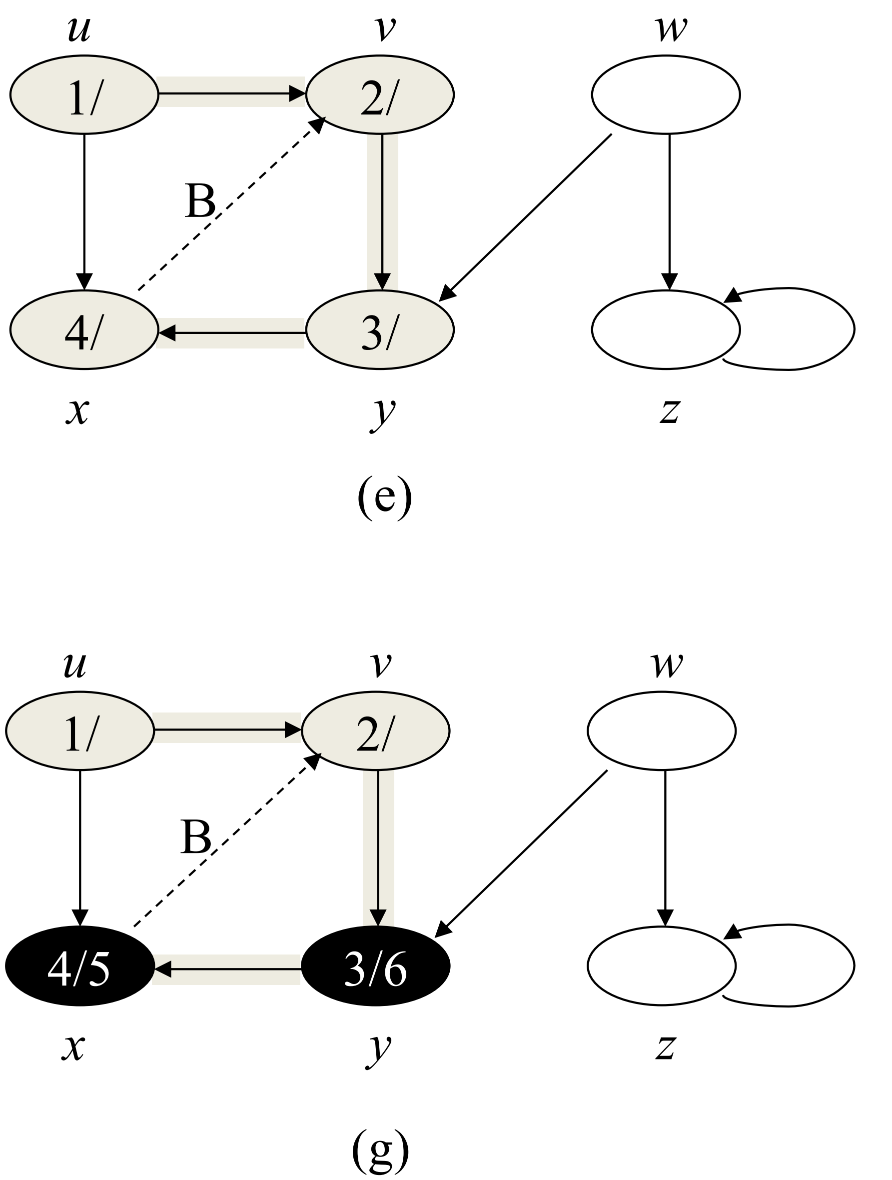
- ***Tree edges***
  - Edges in a depth-first tree.
- ***Back edges* (*cycle*)**
  - Edges from descendants to ancestors or self-loops
- ***Forward edges***:
  - Non-tree edges from ancestors to descendants
- ***Cross edges***:
  - All other edges, which are between vertices such that one vertex is not an ancestor of the other.

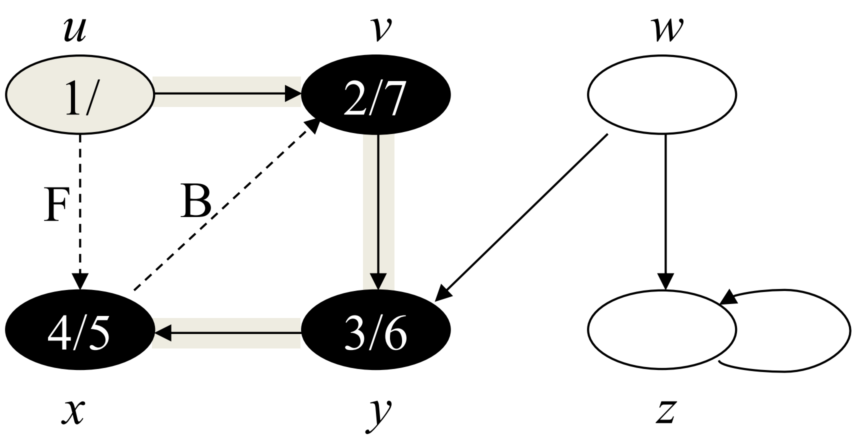
- **Classification by the DFS algorithm**
  - Each edge (u, v) can be classified by the color of the vertex v that is reached when the edge is first explored:
    - ***white*** indicates a tree edge,
    - ***gray*** indicates a back edge, and
    - ***black*** indicates a forward or cross edge.
  - Forward and cross edges are classified by ***the inclusion of gray intervals of u and v***.

Depth-first search

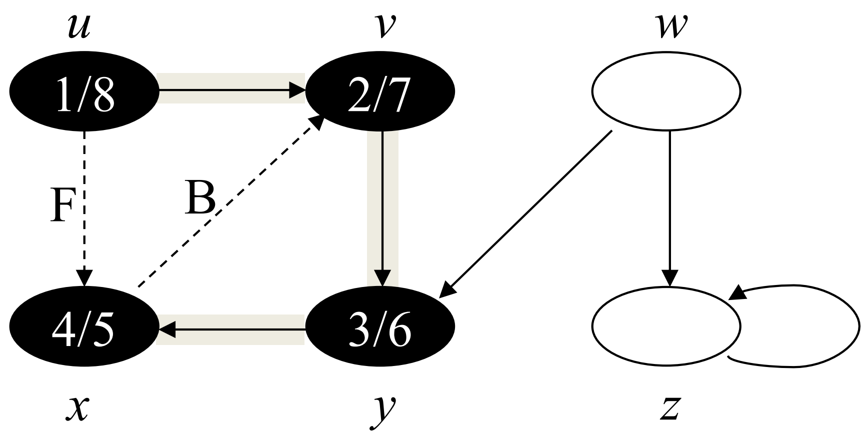




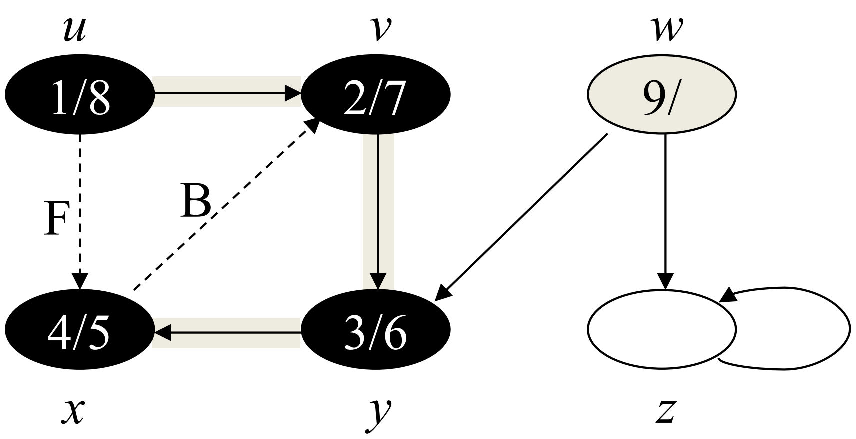

u
v
w
1/
2/
B
4/
3/
x
y
z
(e)
u
v
w
1/
2/
B
4/5
3/
x
y
z
(f)
u
v
w
1/
2/
B
4/5
3/6
x
y
z
(g)
u
v
w
1/
2/7
B
4/5
3/6
x
y
z
(h)


u
v
w
1/
2/7
F
B
4/5
3/6
x
y
z

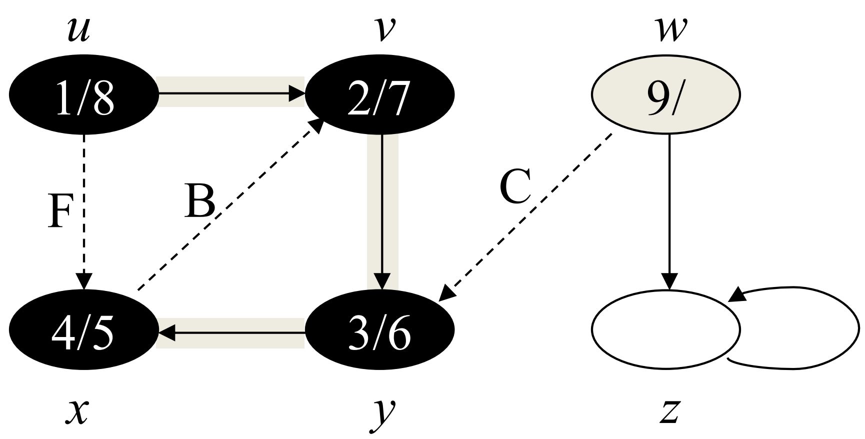
(i)


u
v
w
1/8
2/7
F
B
4/5
3/6
x
y
z

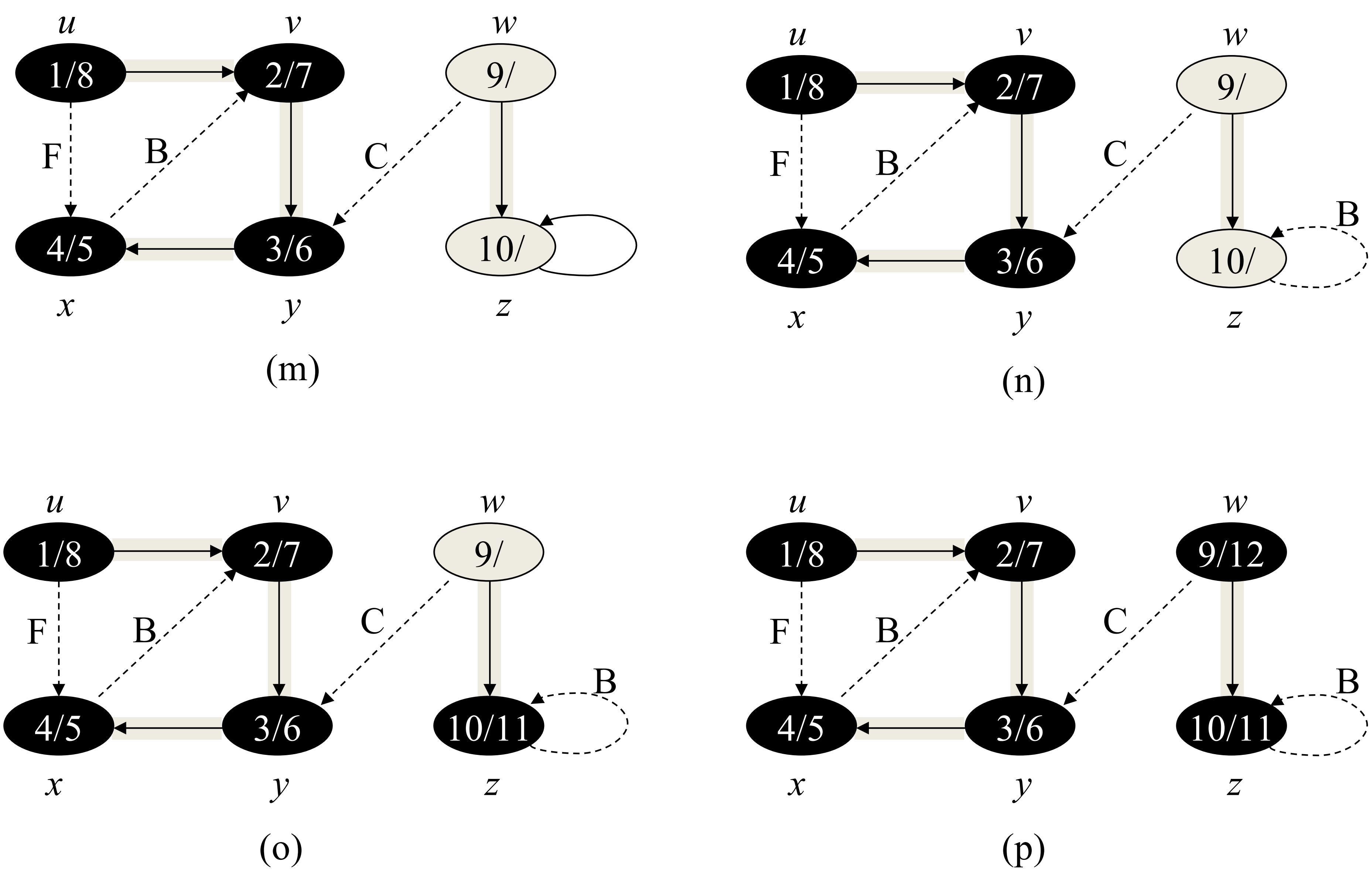
(j)


u
v
w
1/8
2/7
9/
F
B
4/5
3/6
x
y
z

(k)


u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
x
y
z

(l)


u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
10/
x
y
z
(m)
u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/
x
y
z
(n)
u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/11
x
y
z
(o)
u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z
(p)

- In a depth-first search of an ***undirected graph***, every edge of G is either a ***tree edge*** or a ***back edge***.
  - Forward edge?
  - Cross edge?

- Running Time
  - $\Theta(V+E)$

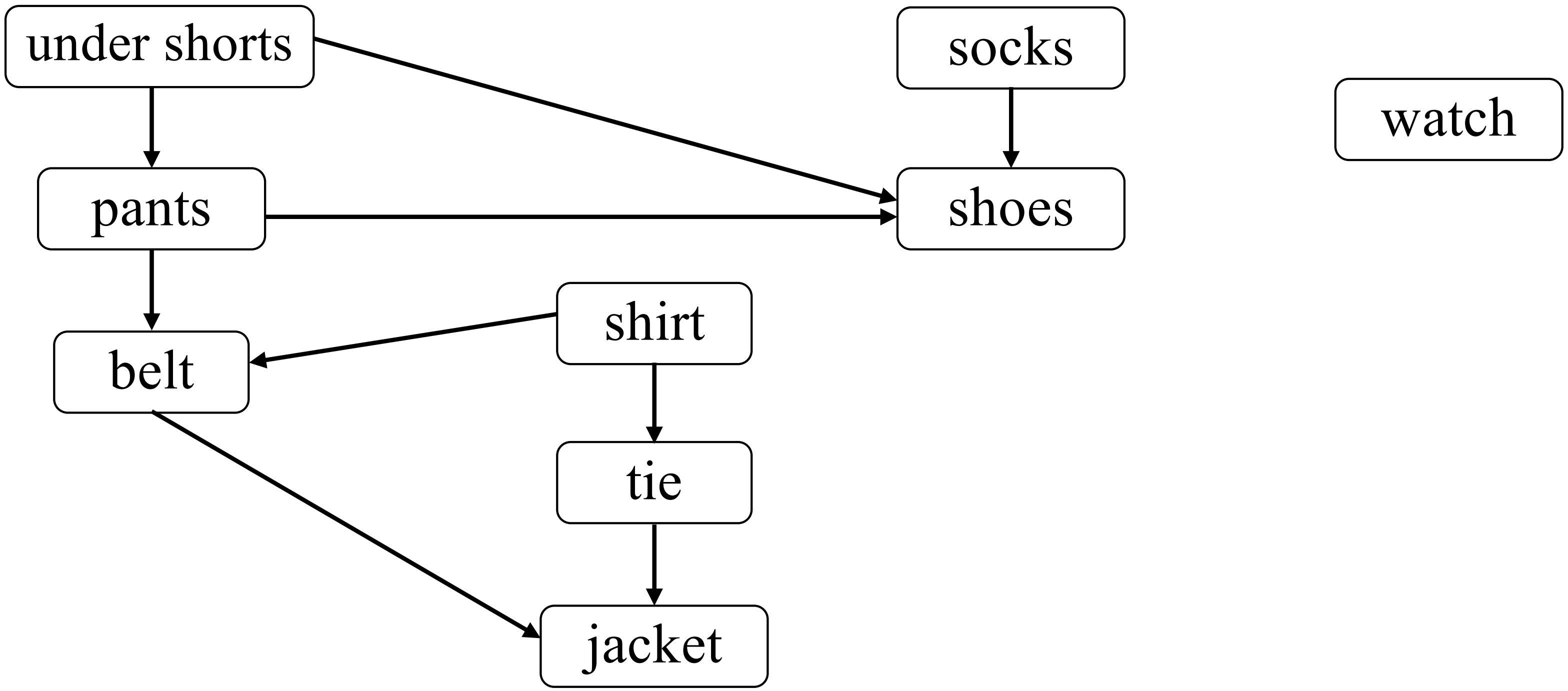
- **Exercise 22.3-5 (22.3-4 in the 2nd ed.)**
  - Edge classification

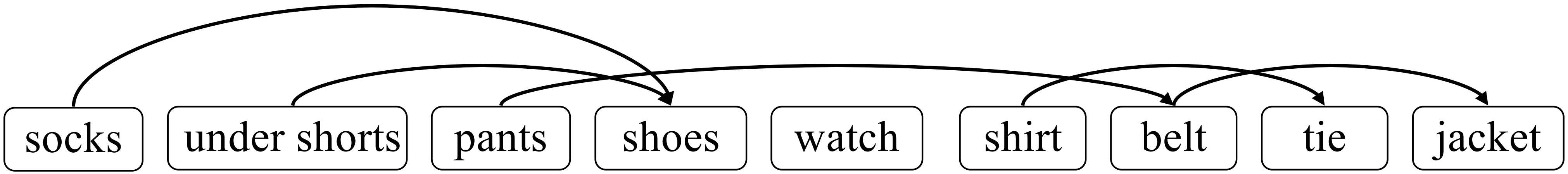
- **Problem 22-2 *a*-*d***
  - Articulation points

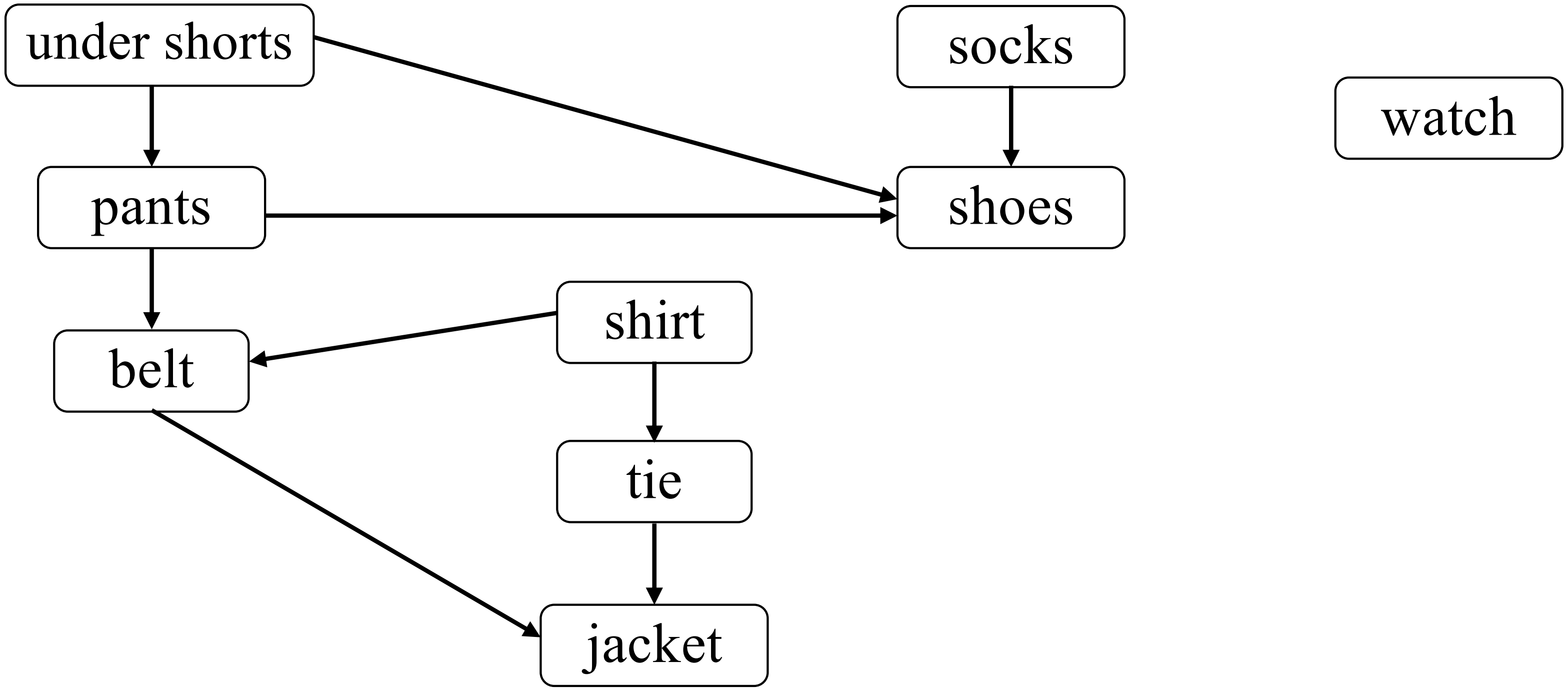
Contents

- ***Graphs***
  - *Graphs basics*
  - *Graph representation*
- ***Searching a graph***
  - *Breadth-first search*
  - *Depth-first search*
- **Applications of depth-first search**
  - Topological sort

- ***Definition***
  - Given a DAG (directed acyclic graph), generate a linear ordering of all its vertices such that all edges go from left to right.

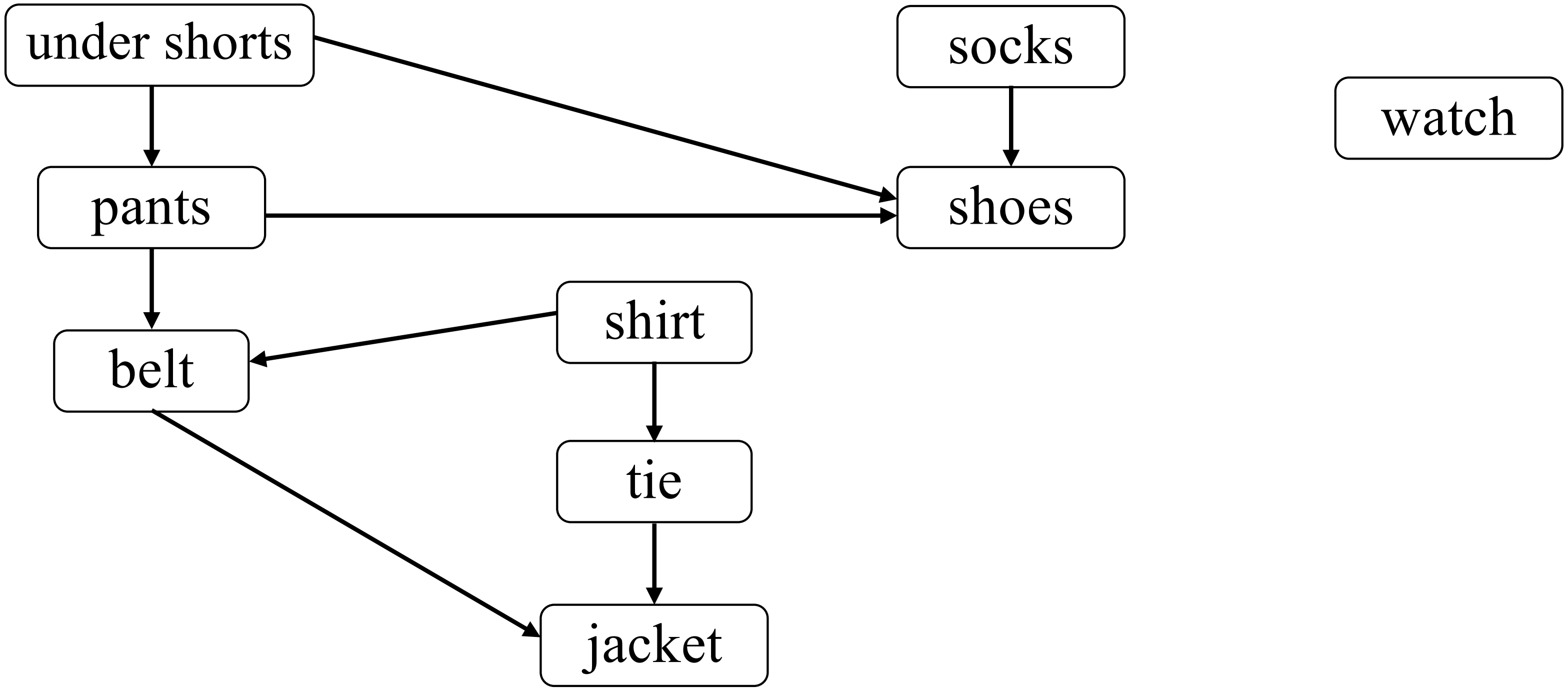

under shorts
socks
watch
pants
shoes
shirt
belt
tie
jacket


socks
under shorts
pants
shoes
watch
shirt
belt
tie
jacket



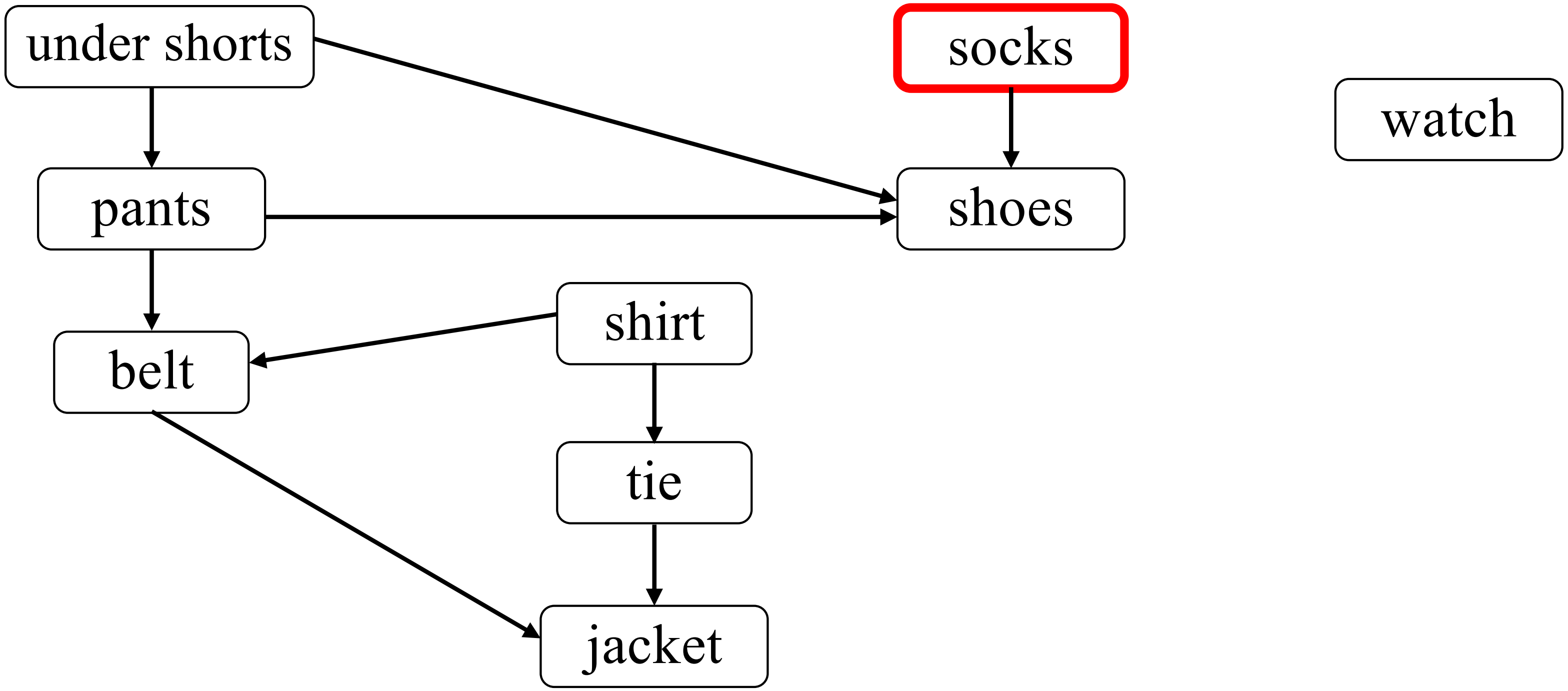
Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
0	0	1	3	0	0	2	1	2



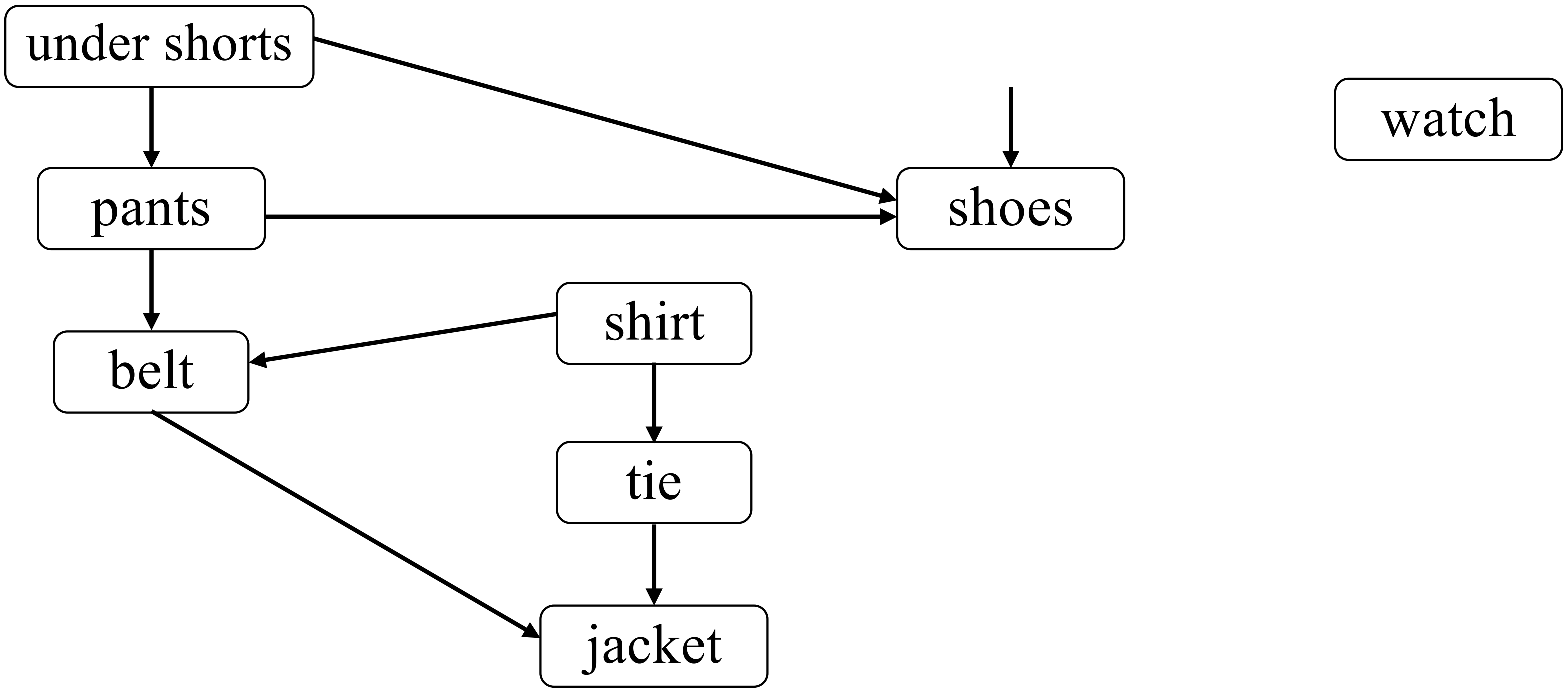
Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
0	0	1	3	0	0	2	1	2



Indegree Array

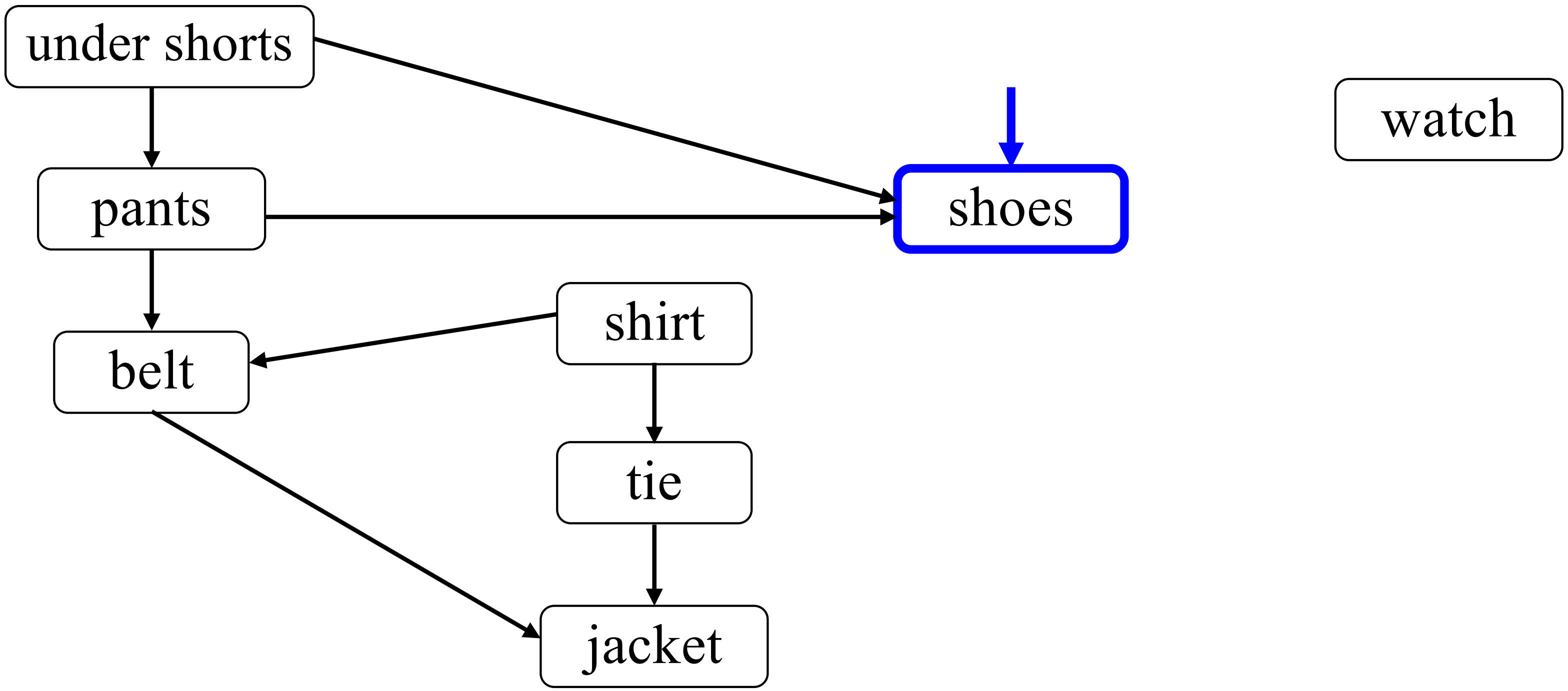
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
0	0	1	3	0	0	2	1	2



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	3	0	0	2	1	2

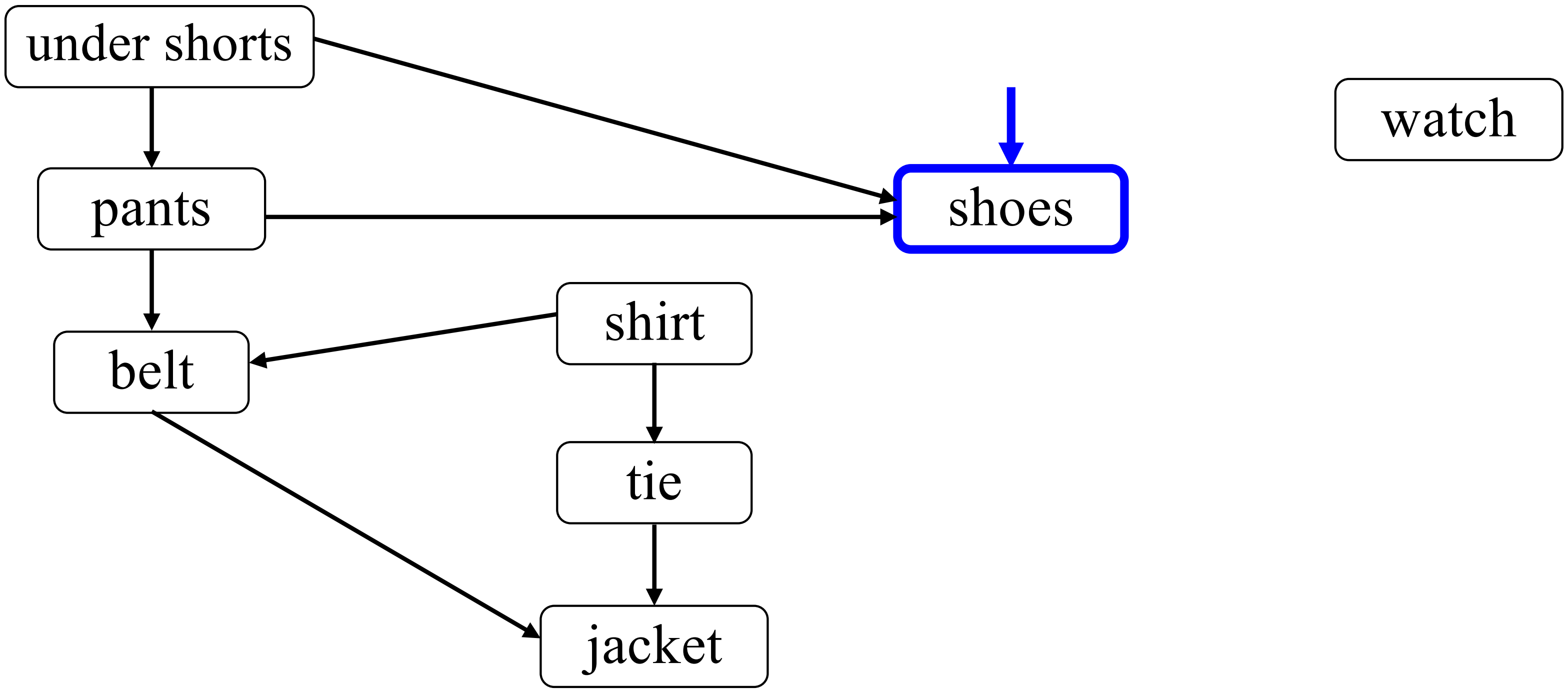
socks



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	3	0	0	2	1	2

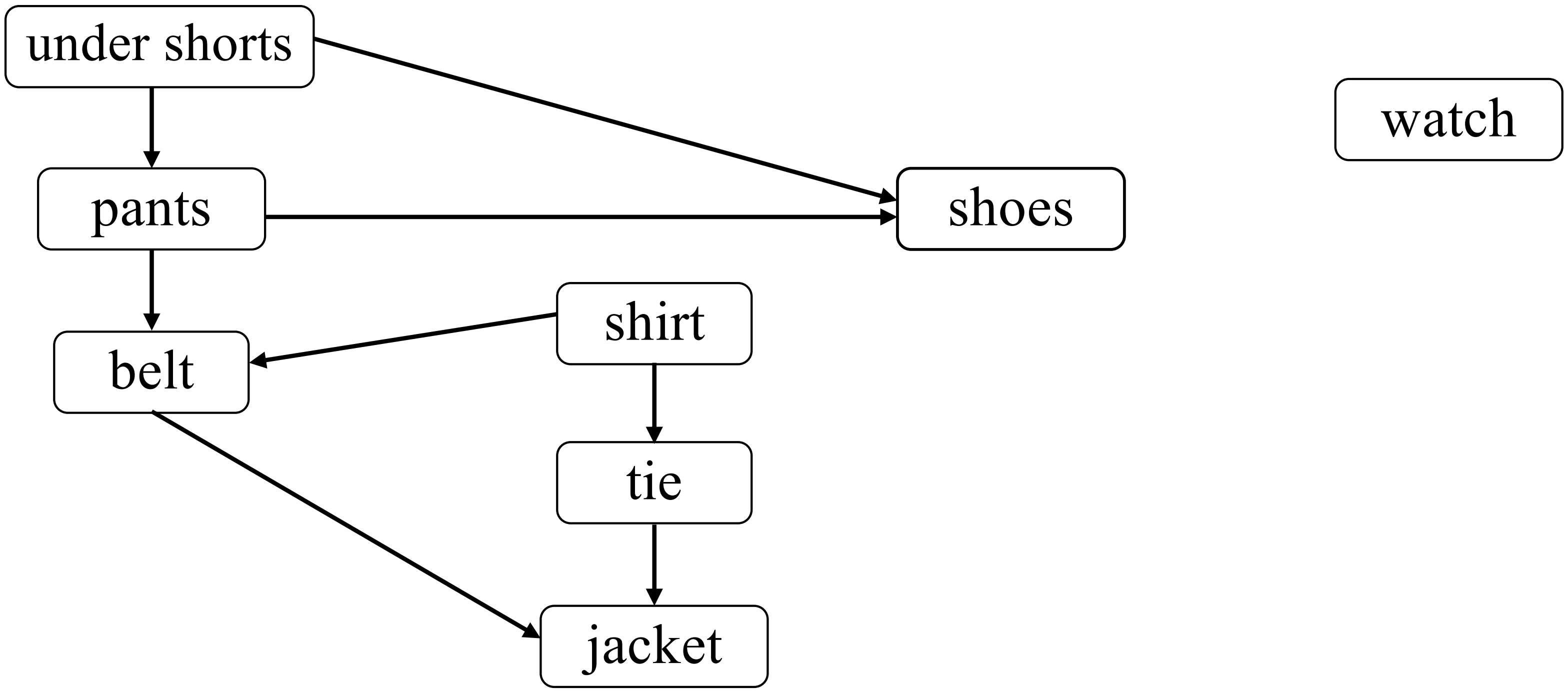
socks



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	3 → 2	0	0	2	1	2

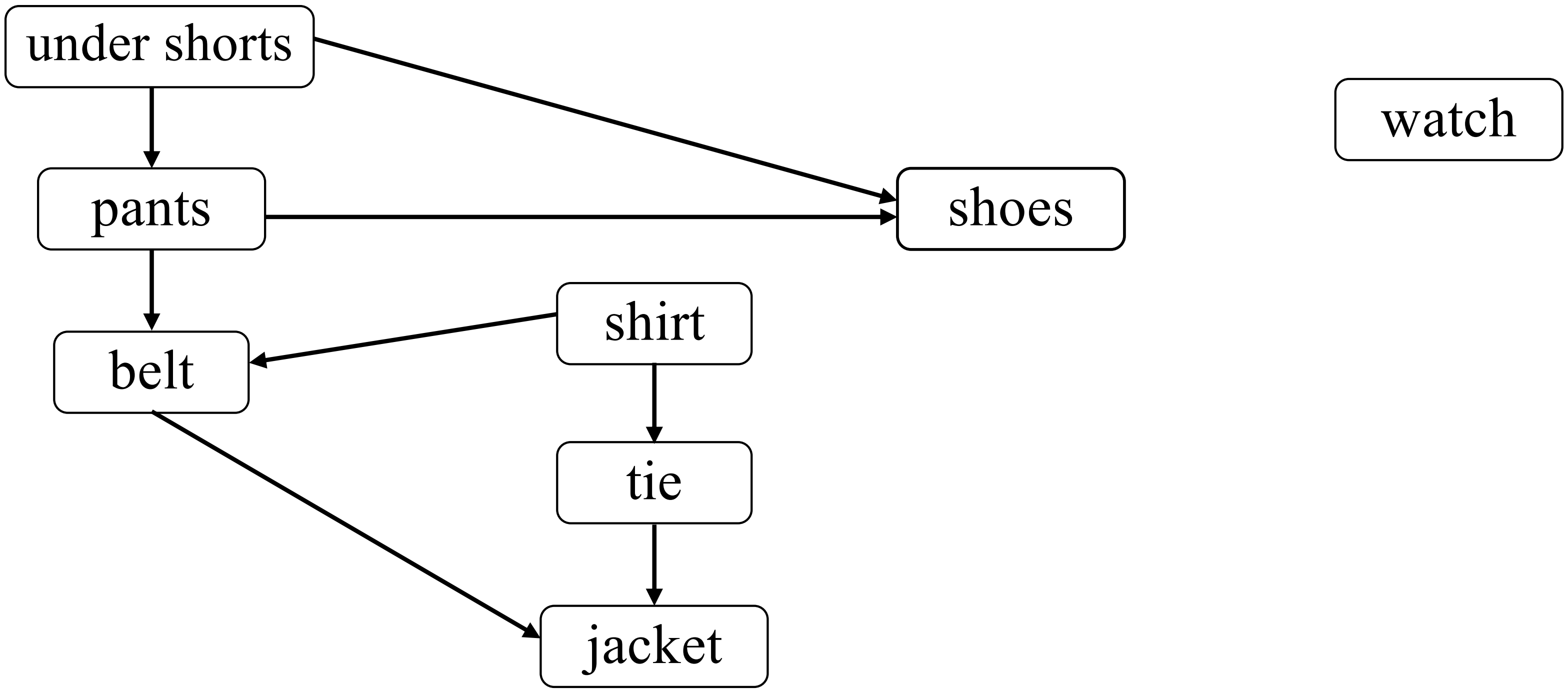
socks



Indegree Array

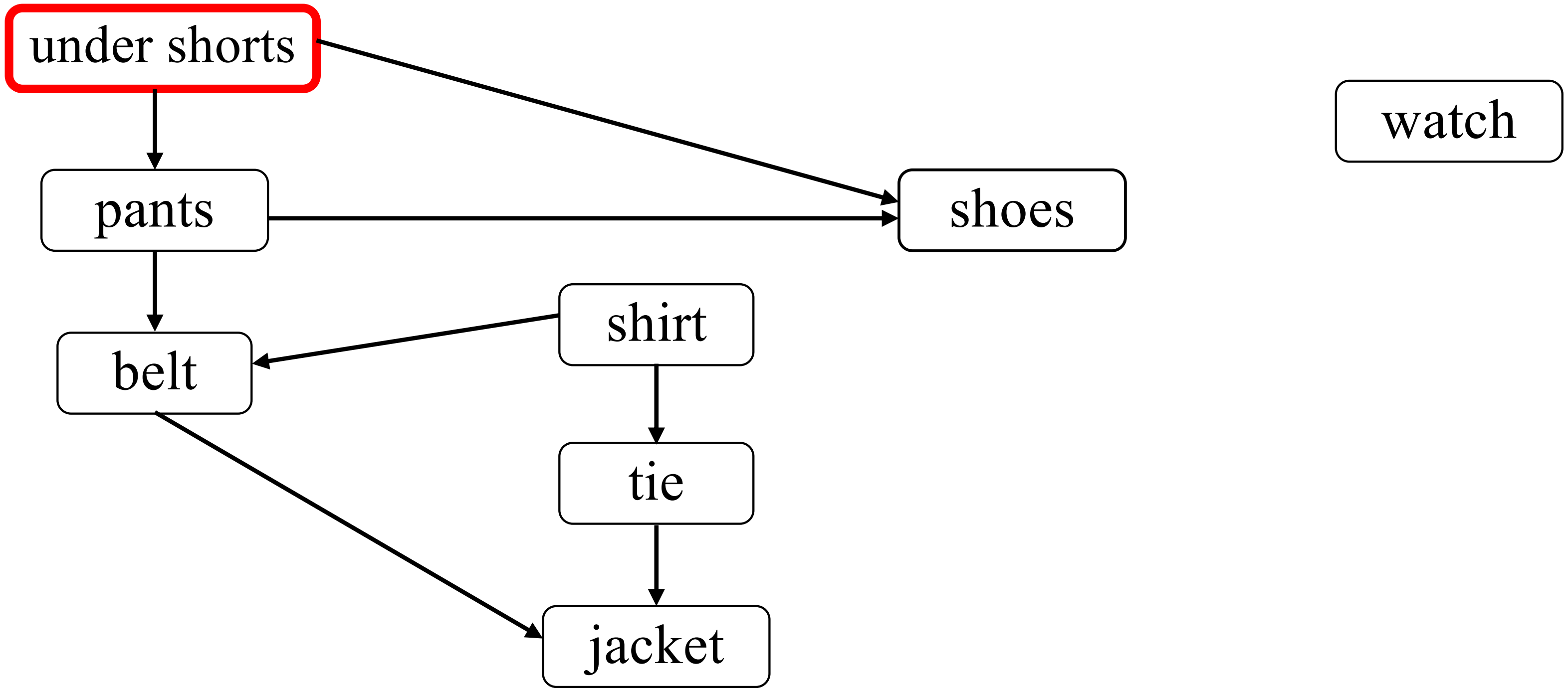
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	2	0	0	2	1	2

socks



Indegree Array

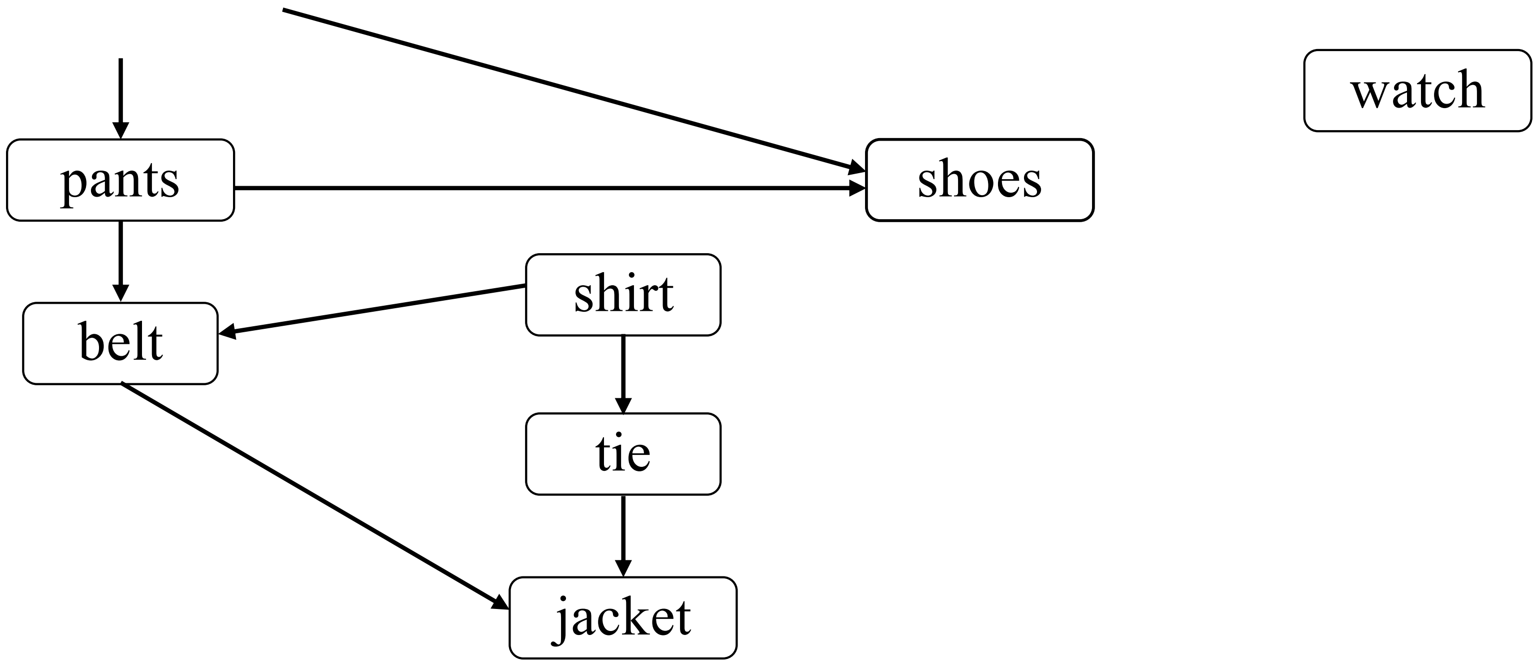
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	2	0	0	2	1	2



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	0	1	2	0	0	2	1	2

socks

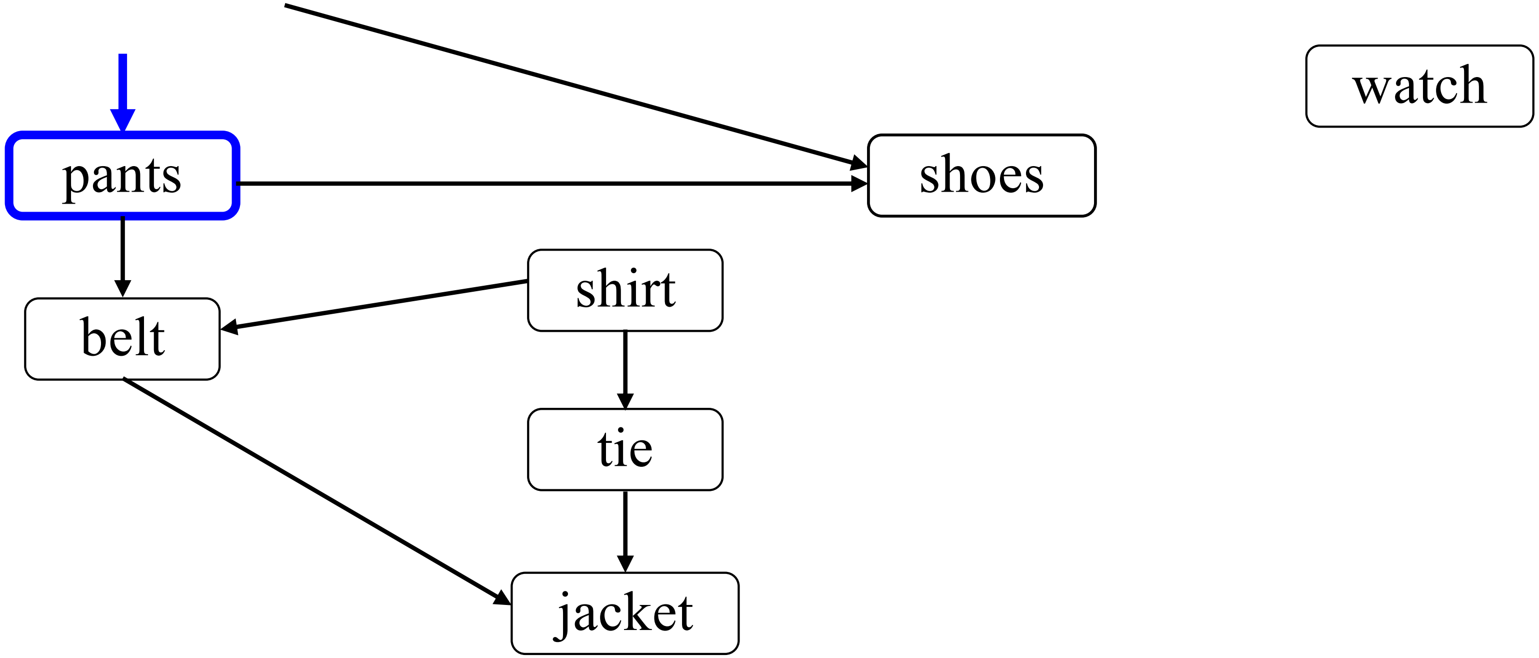


Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	1	2	0	0	2	1	2

socks
under shorts

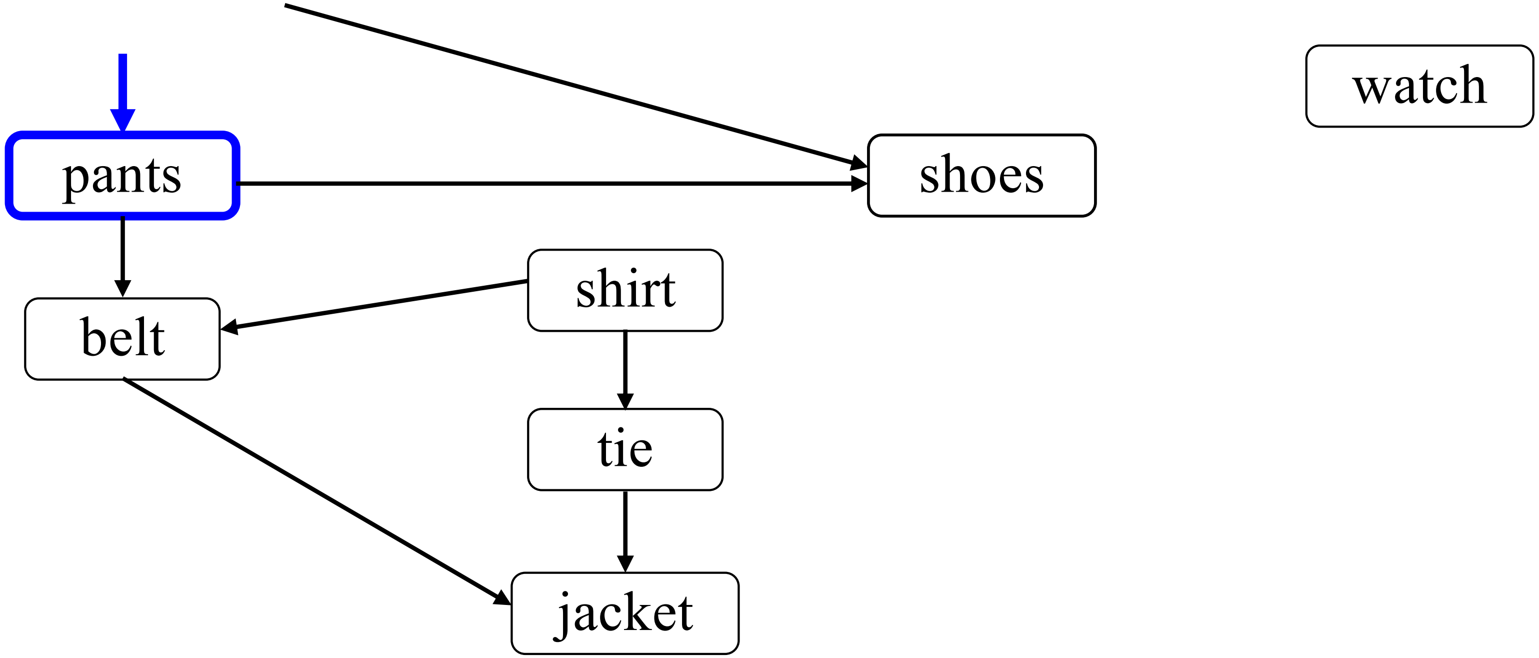
Topological sort



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	1	2	0	0	2	1	2

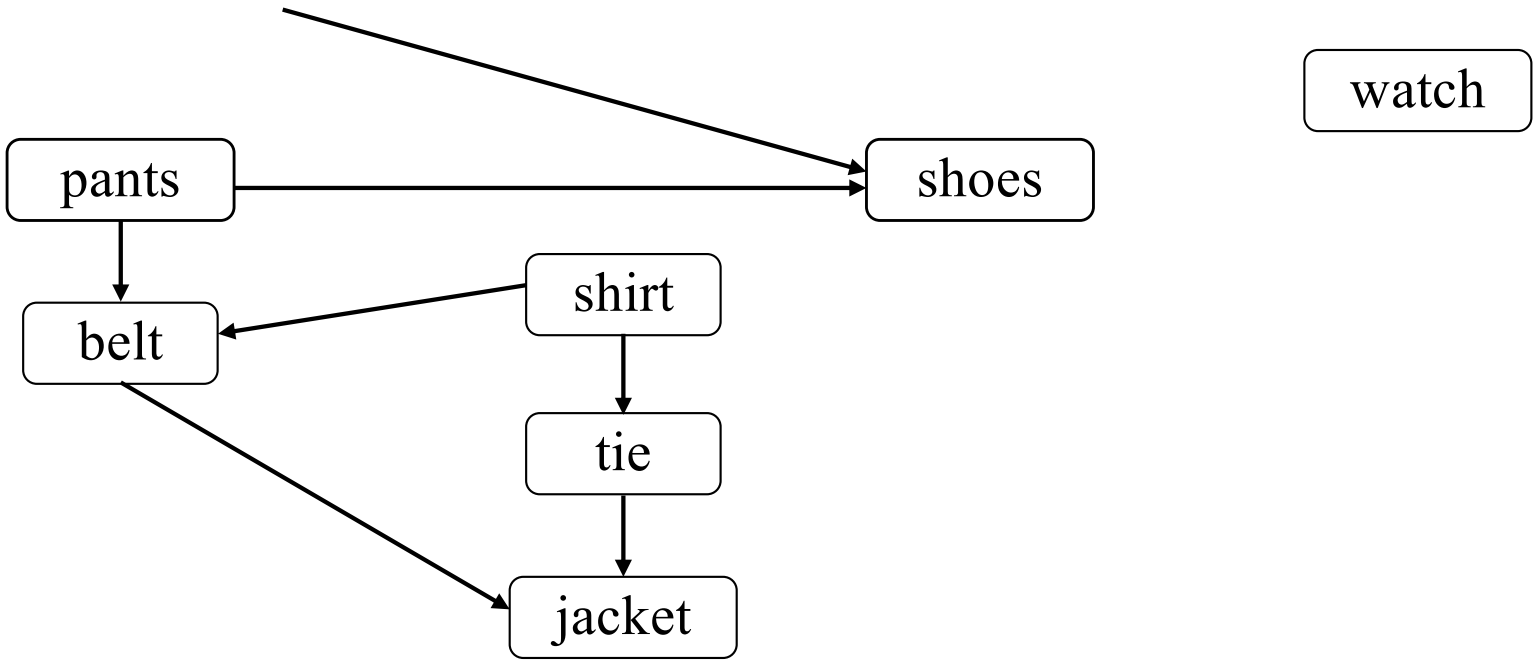
socks | under shorts



Indegree Array

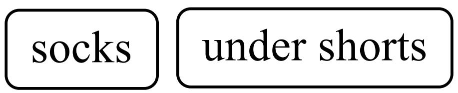
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	1 → 0	2	0	0	2	1	2

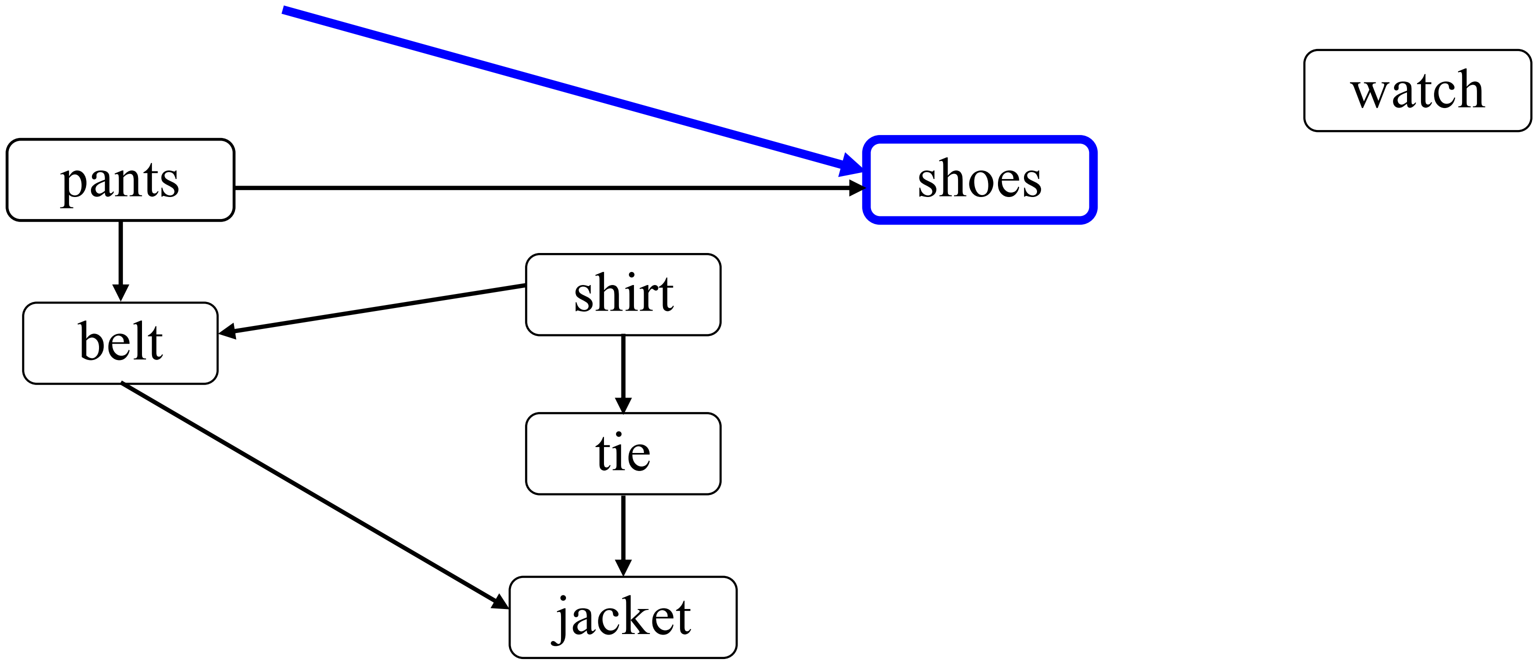
socks under shorts



Indegree Array

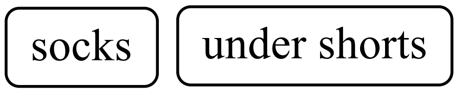
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	2	0	0	2	1	2

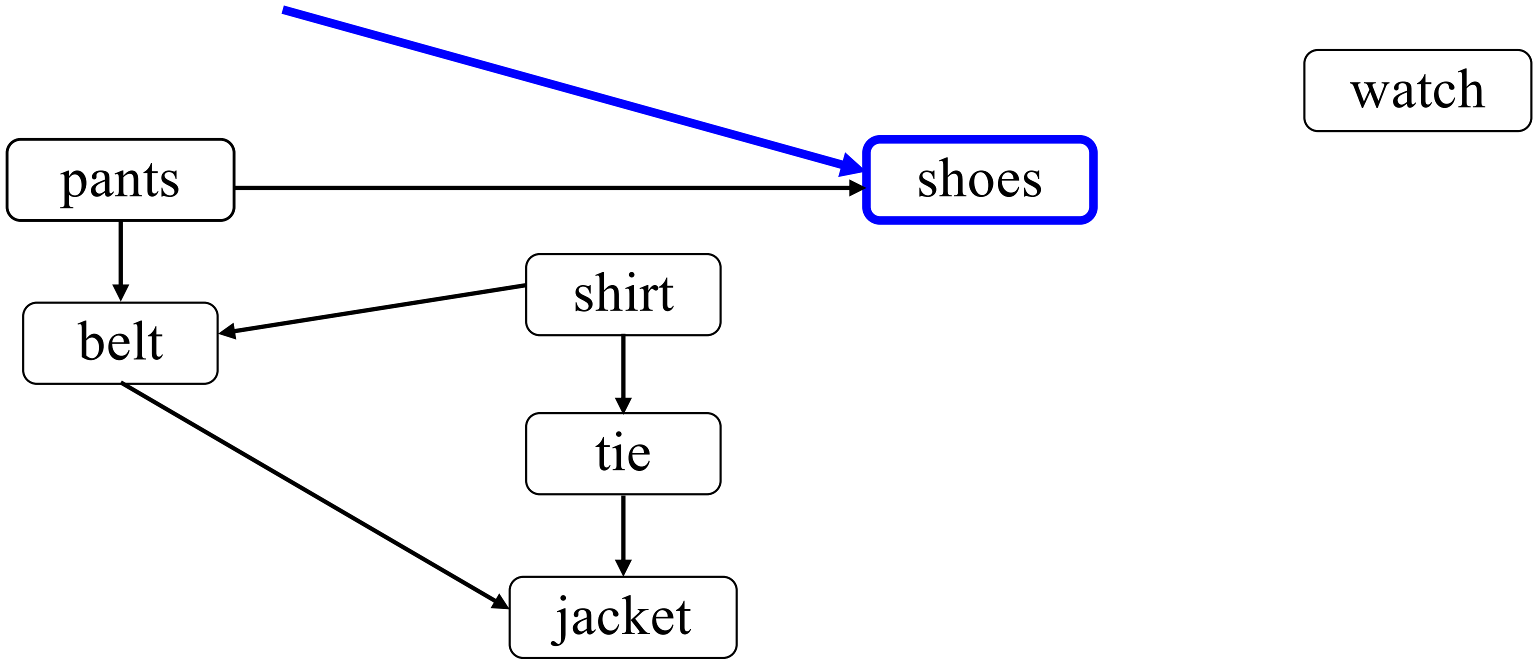




Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	2	0	0	2	1	2



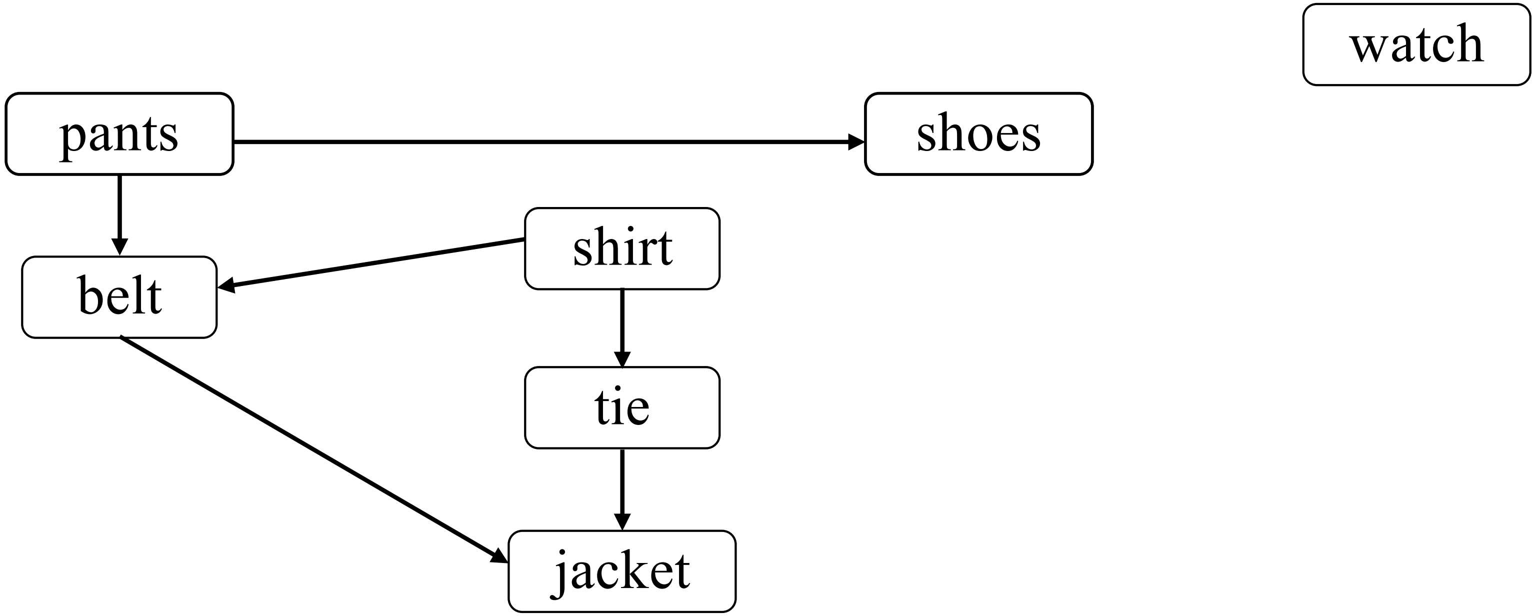


Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	2 → 1	0	0	2	1	2

socks under shorts

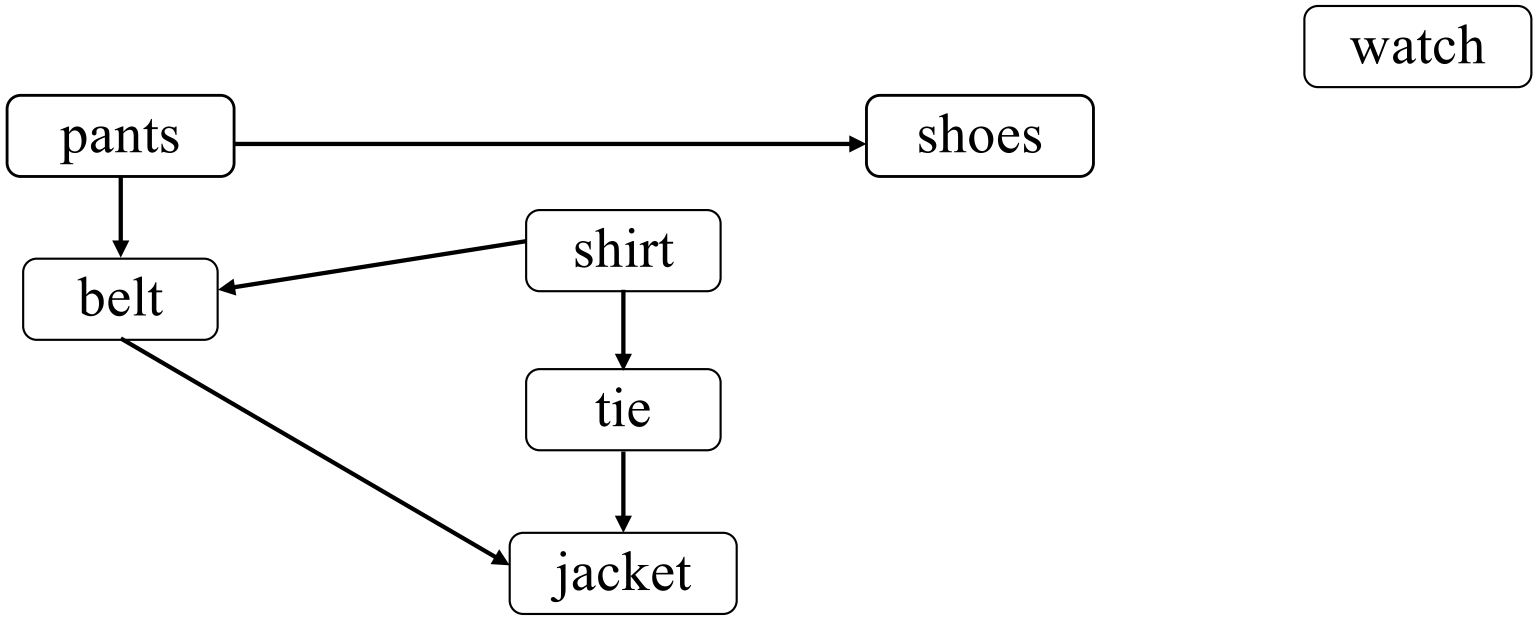
Topological sort



Indegree Array

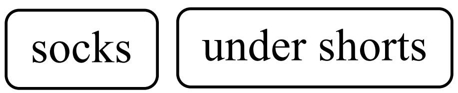
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	1	0	0	2	1	2

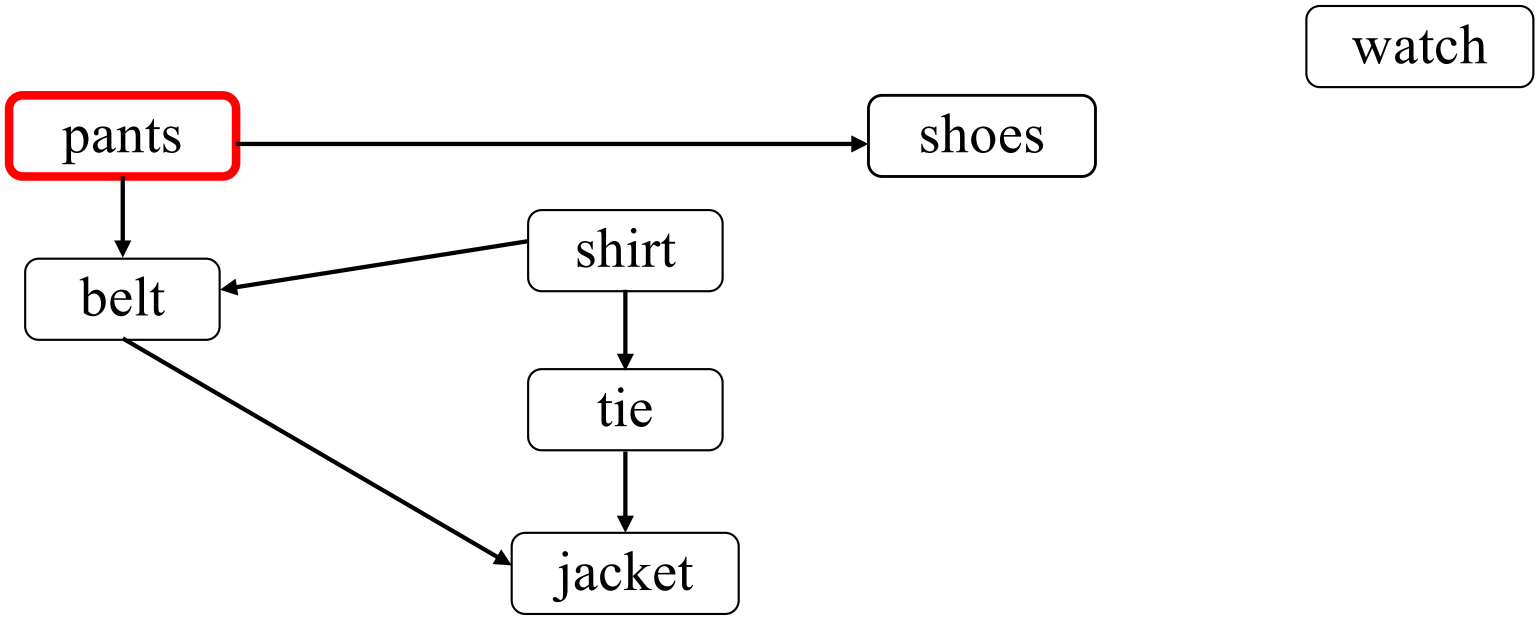
socks | under shorts



Indegree Array

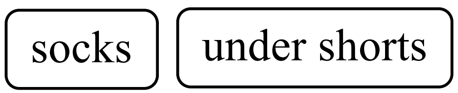
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	1	0	0	2	1	2

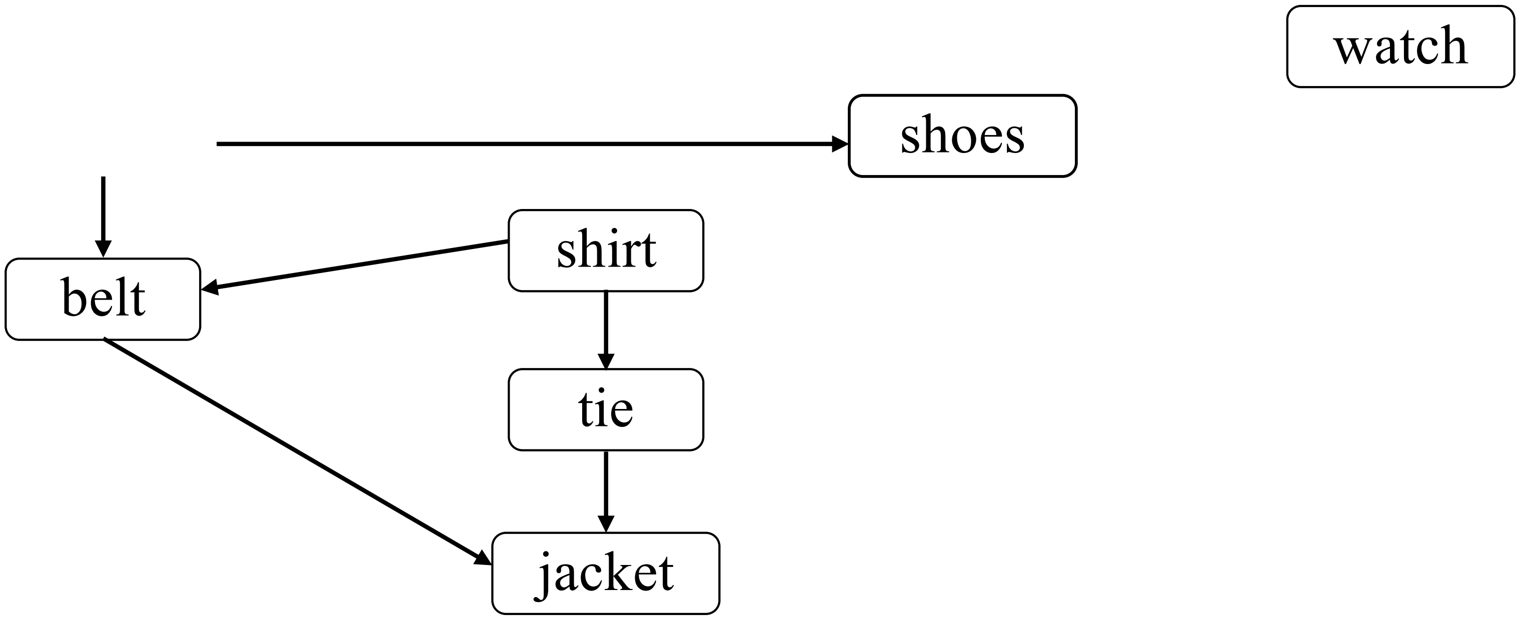




Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	0	1	0	0	2	1	2

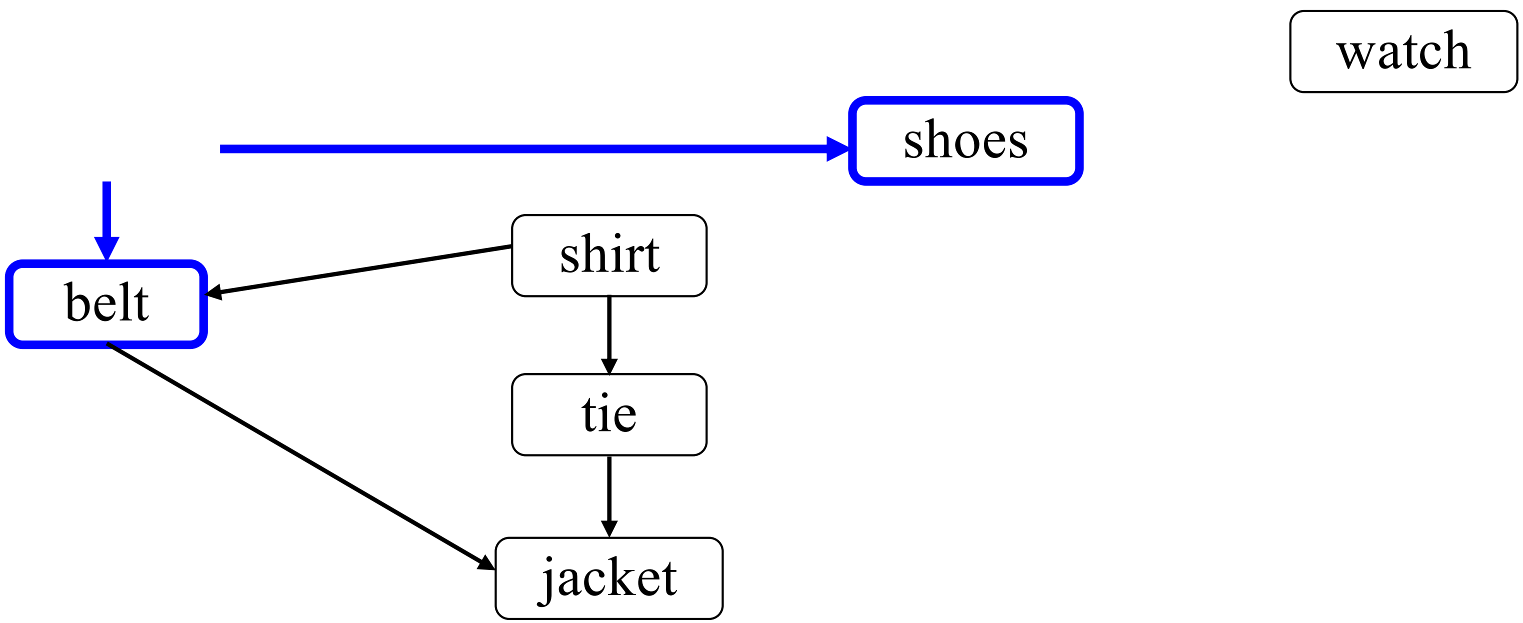




Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	1	0	0	2	1	2

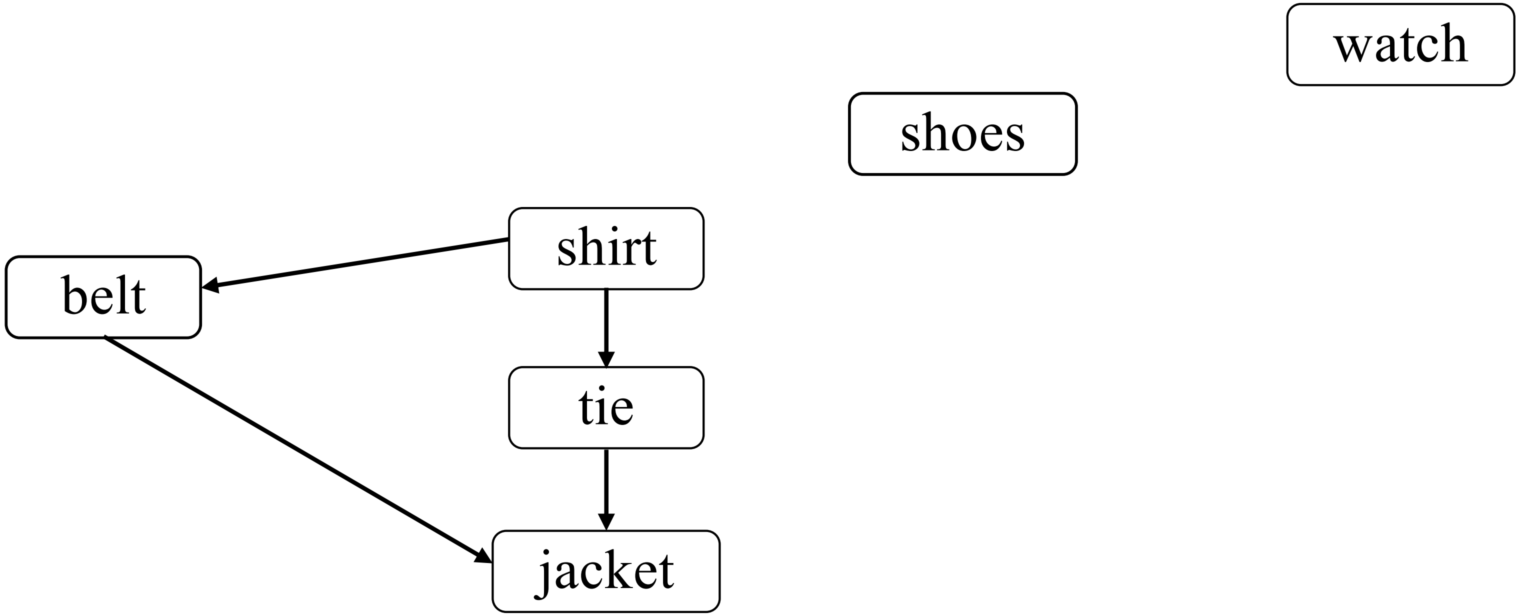
socks　under shorts　pants



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	1 → 0	0	0	2 → 1	1	2

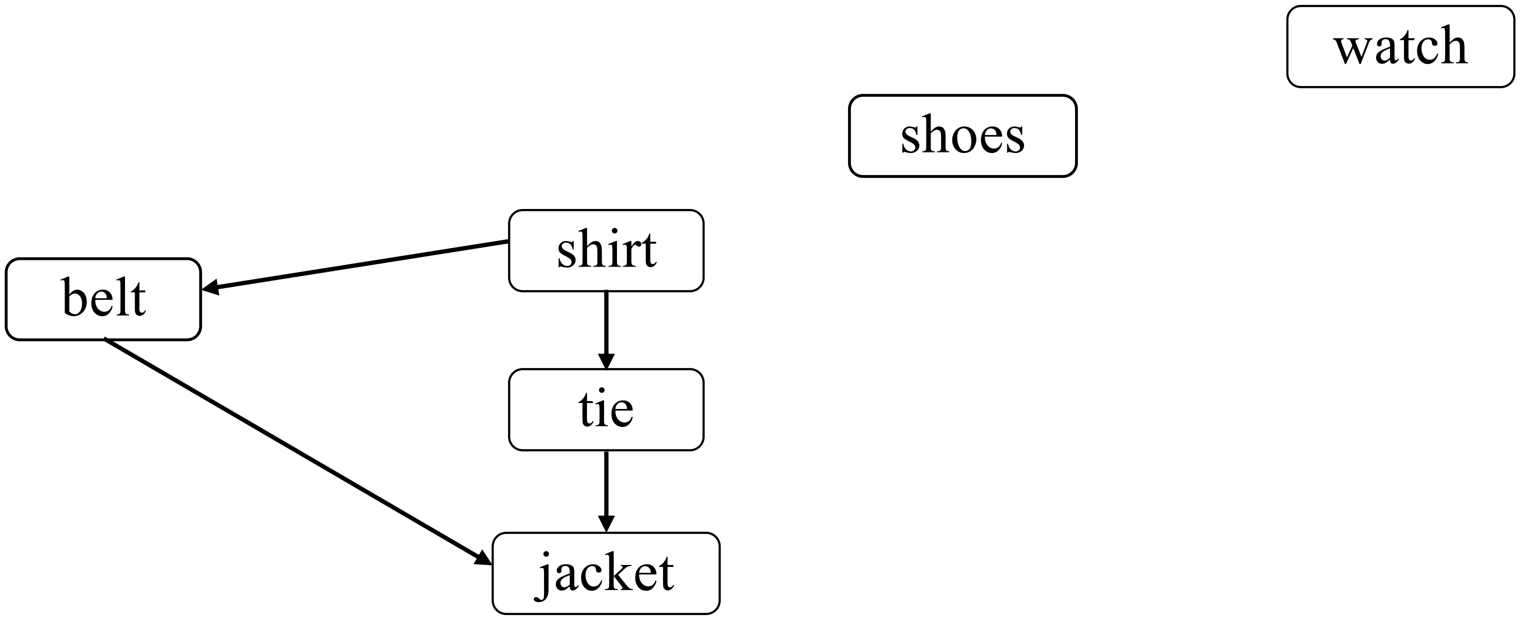
socks | under shorts | pants



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	0	0	0	1	1	2

socks | under shorts | pants

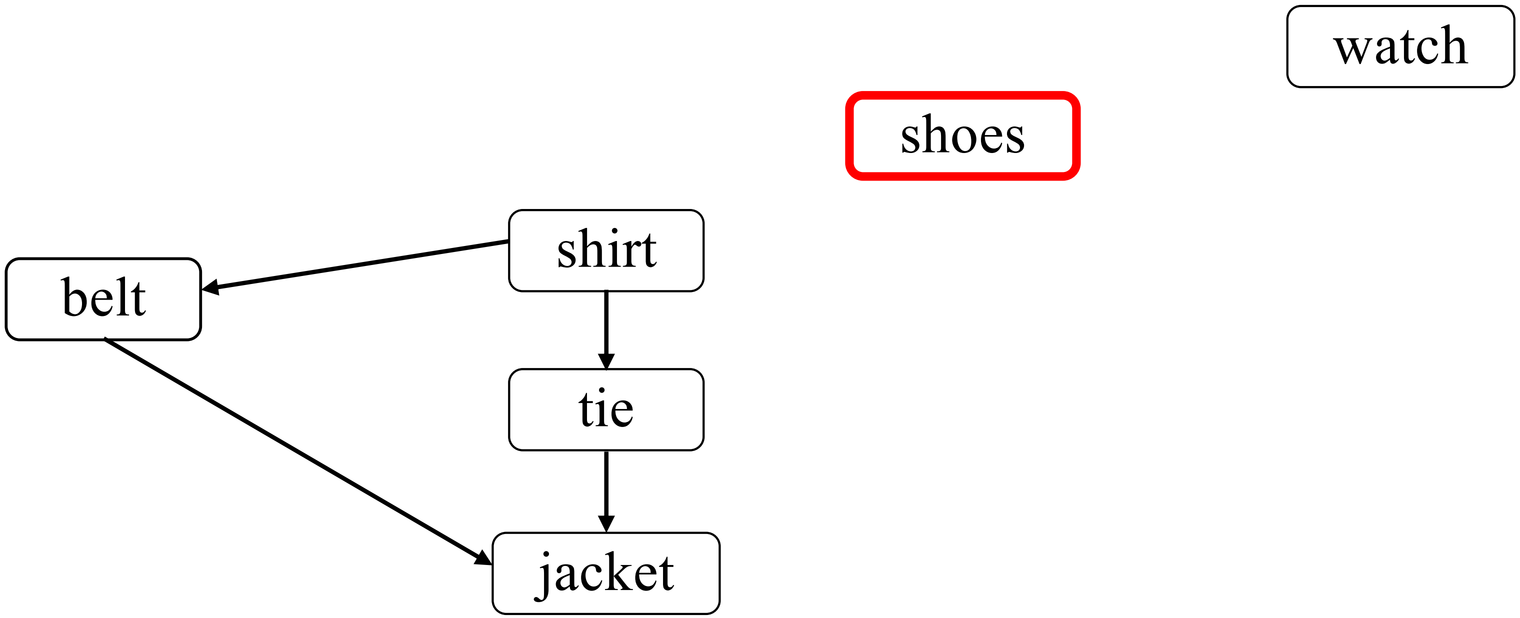


Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	0	0	0	1	1	2

socks under shorts pants

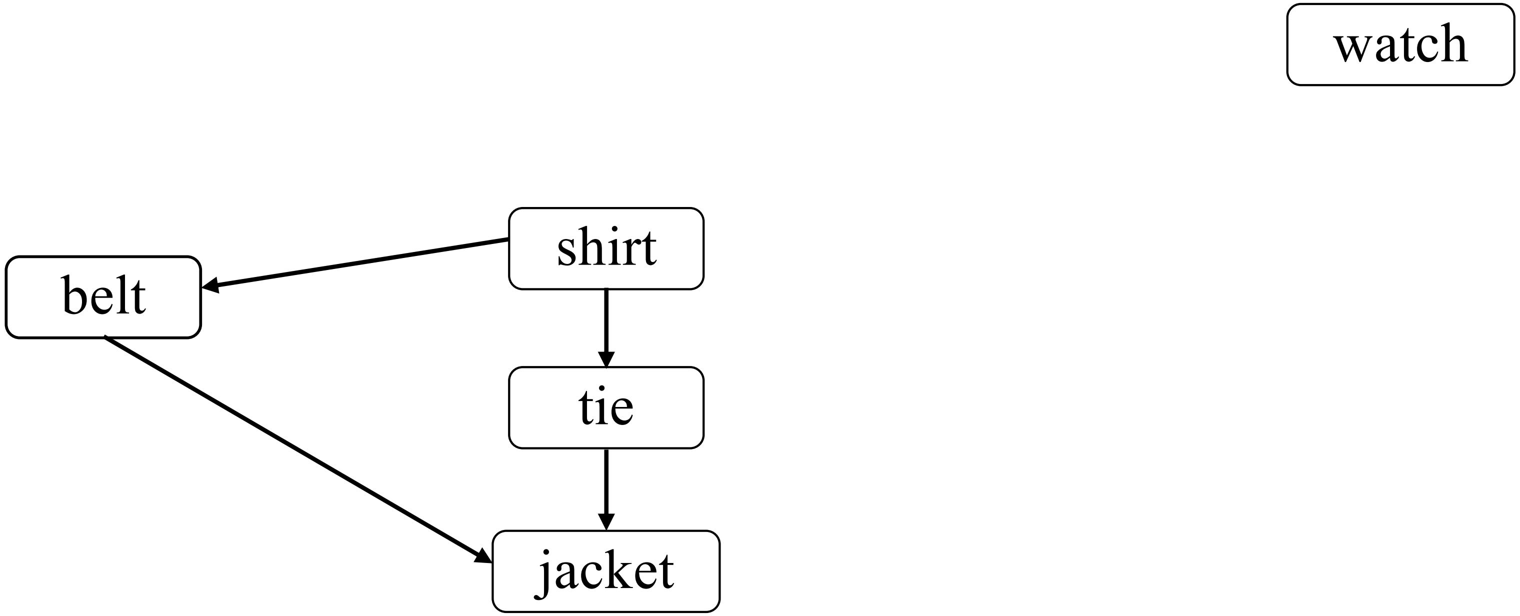
Topological sort



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	0	0	0	1	1	2

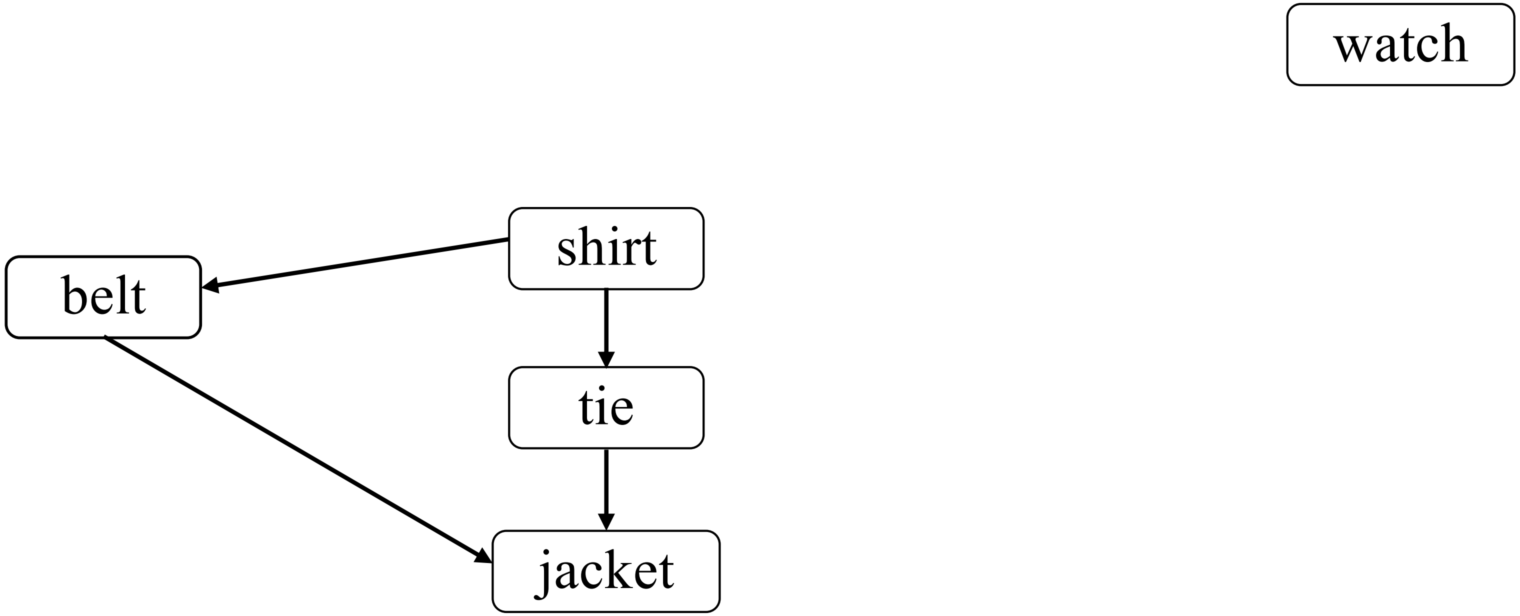
socks under shorts pants



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	0	0	1	1	2

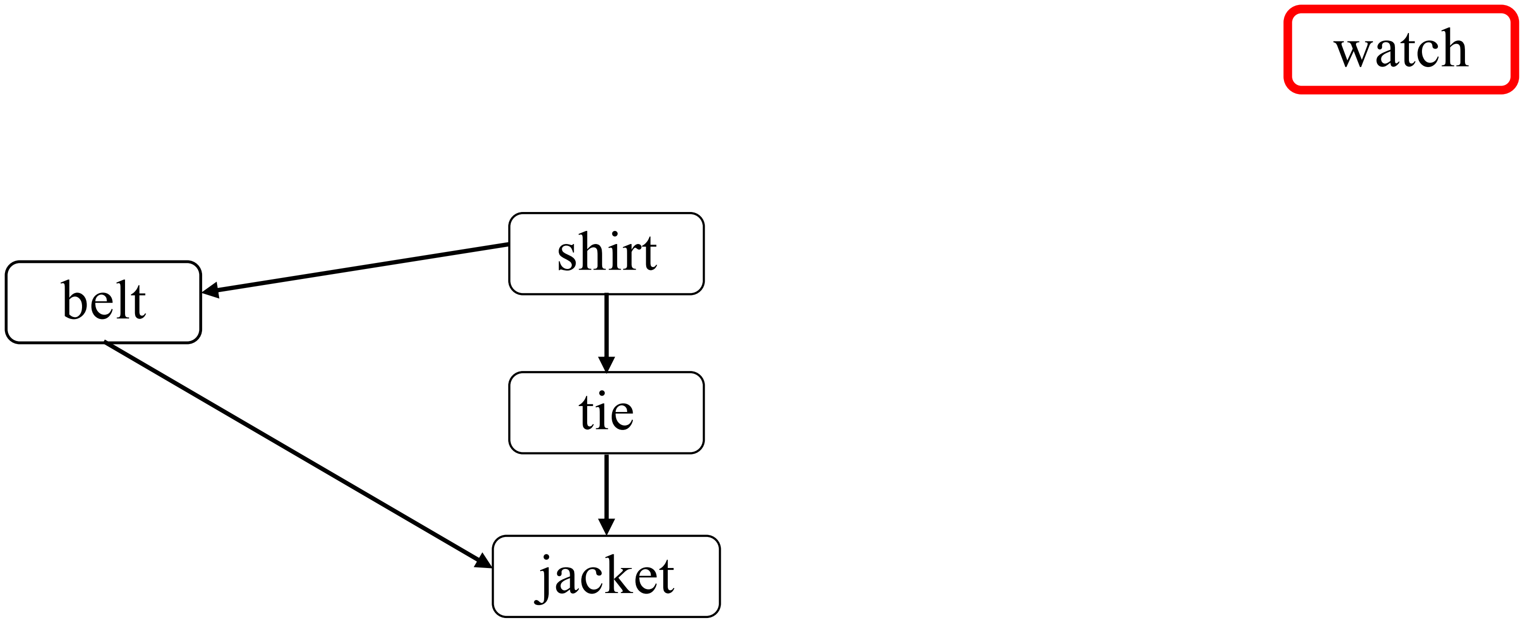
socks under shorts pants shoes



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	0	0	1	1	2

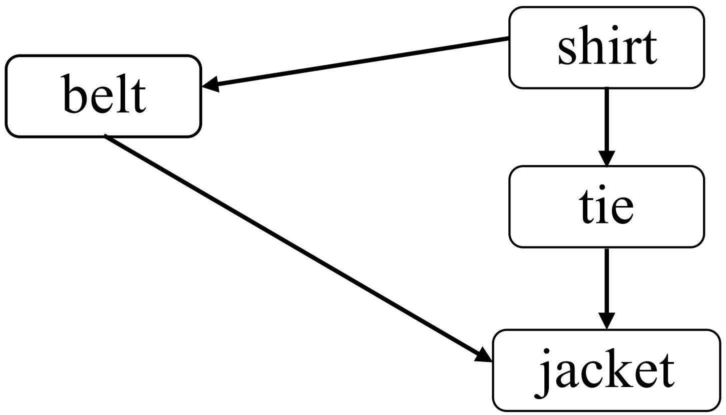
socks | under shorts | pants | shoes



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	0	0	1	1	2

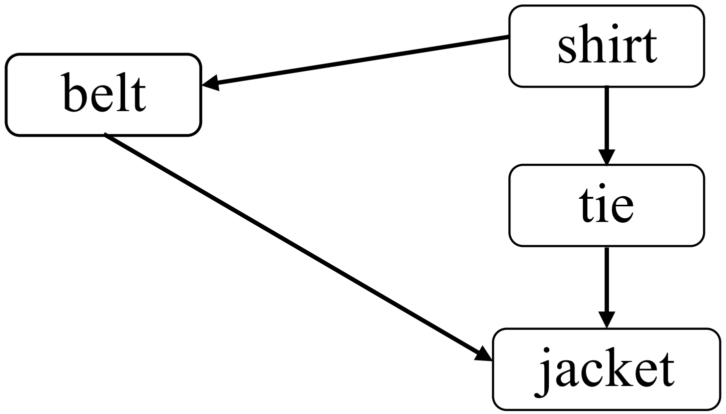
socks | under shorts | pants | shoes



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	0	1	1	2

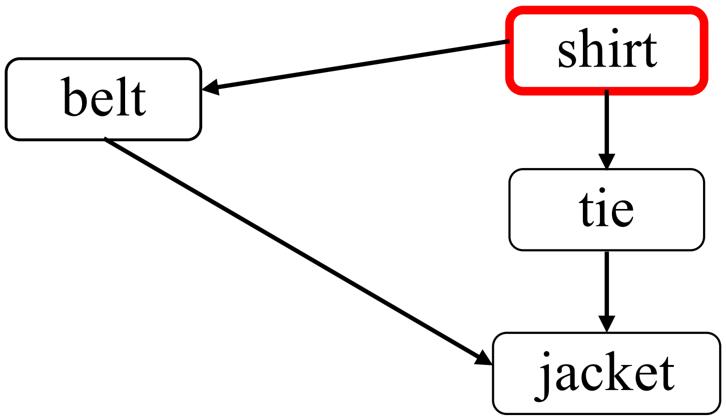
socks | under shorts | pants | shoes | watch



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	0	1	1	2

socks | under shorts | pants | shoes | watch

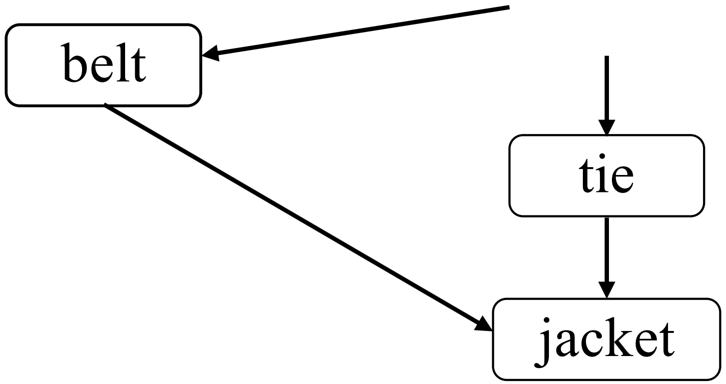


Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	0	1	1	2

socks under shorts pants shoes watch

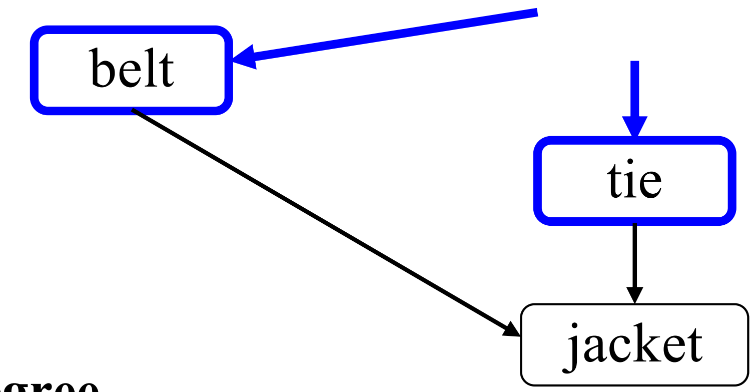
Topological sort



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	1	1	2

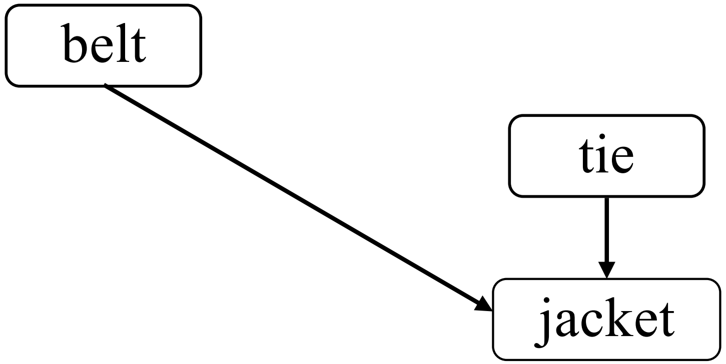
socks | under shorts | pants | shoes | watch | shirt



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	1 → 0	1 → 0	2

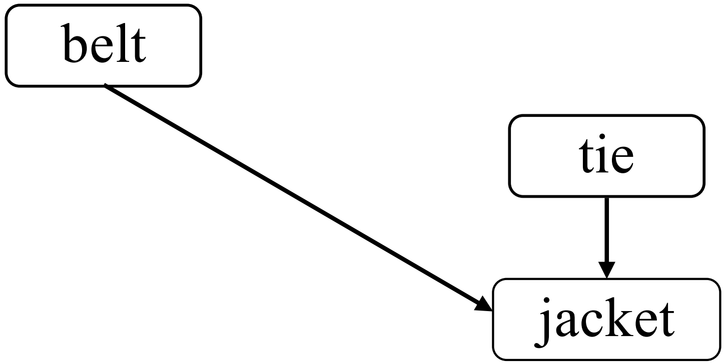
socks | under shorts | pants | shoes | watch | shirt



Indegree Array

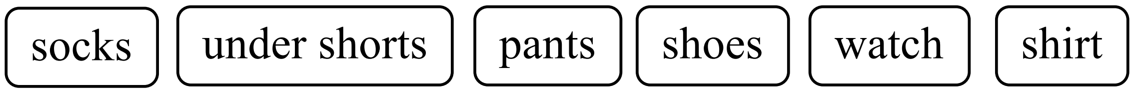
socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	0	0	2

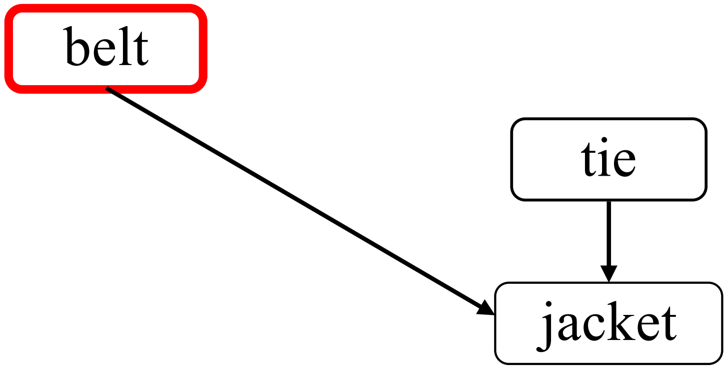
socks under shorts pants shoes watch shirt



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	0	0	2

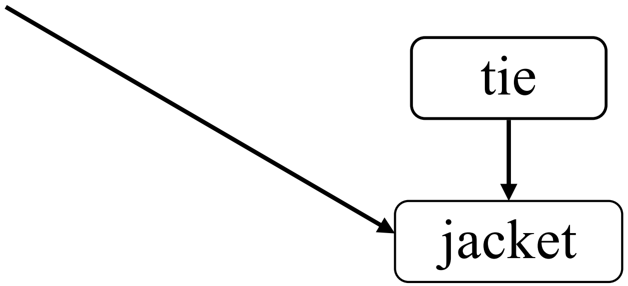




Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	0	0	2

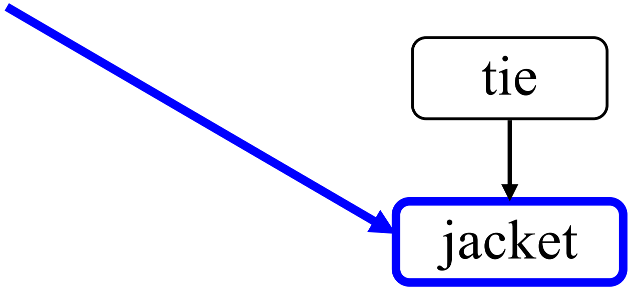
socks | under shorts | pants | shoes | watch | shirt



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	0	2

socks under shorts pants shoes watch shirt belt

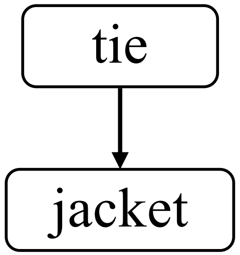


Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	0	2 → 1

socks | under shorts | pants | shoes | watch | shirt | belt

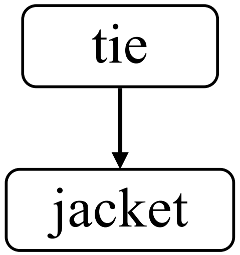
Topological sort



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	0	1

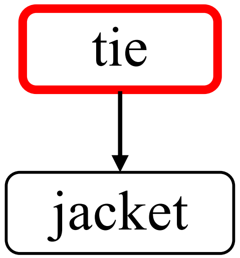
socks | under shorts | pants | shoes | watch | shirt | belt



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	0	1

socks under shorts pants shoes watch shirt belt



Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	0	1

socks | under shorts | pants | shoes | watch | shirt | belt

jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	1

socks under shorts pants shoes watch shirt belt tie

Topological sort

jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	1 → 0

socks | under shorts | pants | shoes | watch | shirt | belt | tie

jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	0

socks under shorts pants shoes watch shirt belt tie

jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	0

socks under shorts pants shoes watch shirt belt tie

jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	0

socks under shorts pants shoes watch shirt belt tie

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	-1

socks | under shorts | pants | shoes | watch | shirt | belt | tie | jacket

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	-1

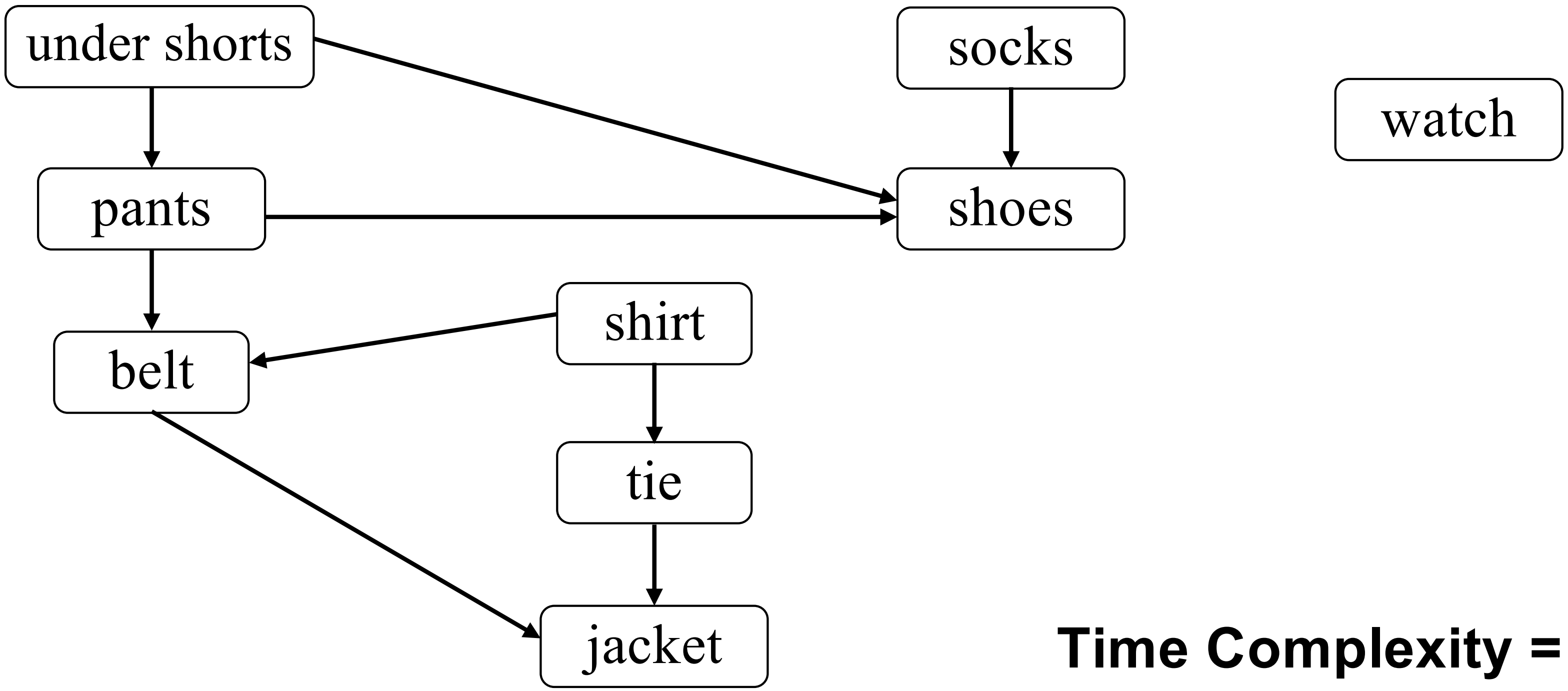
socks under shorts pants shoes watch shirt belt tie jacket

Time Complexity = $O(V^2)$

Indegree Array

socks	under shorts	pants	shoes	watch	shirt	belt	tie	jacket
-1	-1	-1	-1	-1	-1	-1	-1	-1

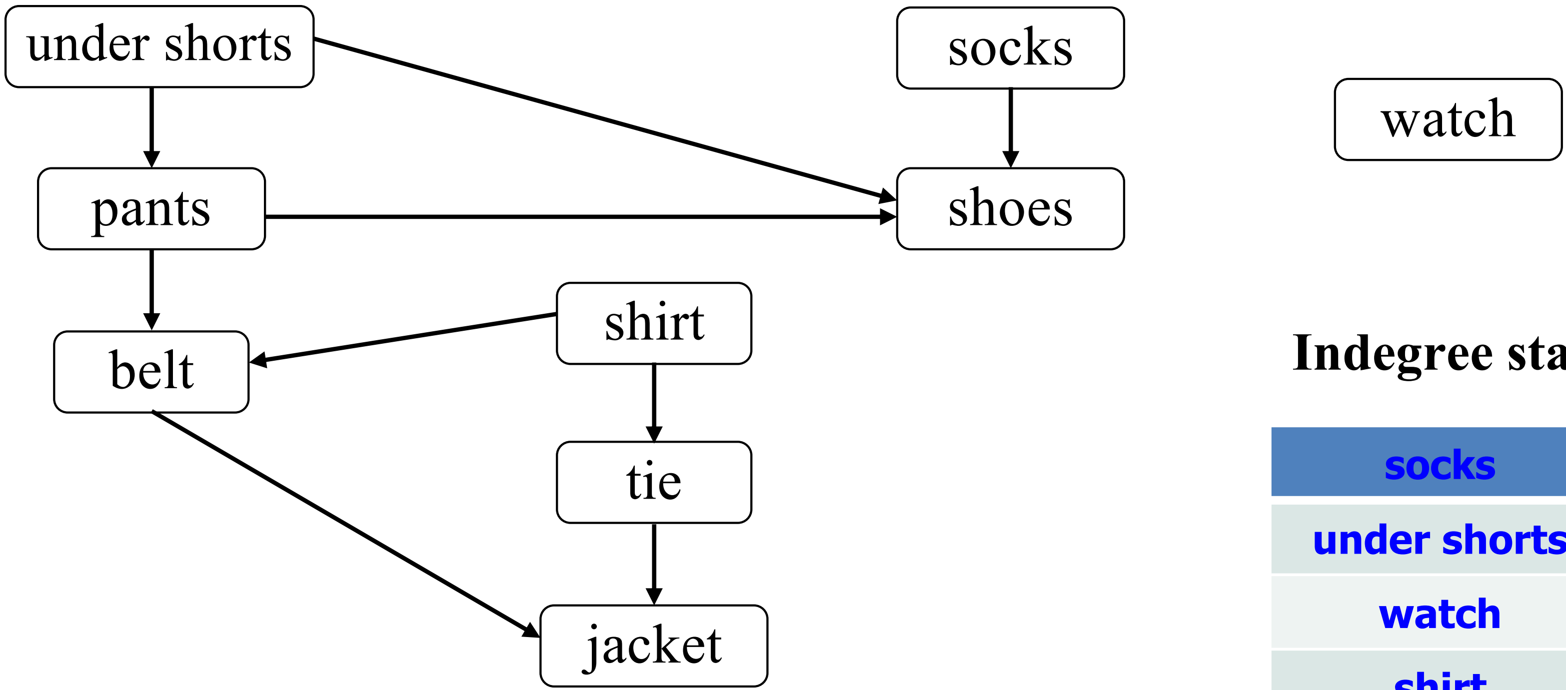
socks | under shorts | pants | shoes | watch | shirt | belt | tie | jacket



Time Complexity = O(V+E)

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	3	0	0	2	1	2

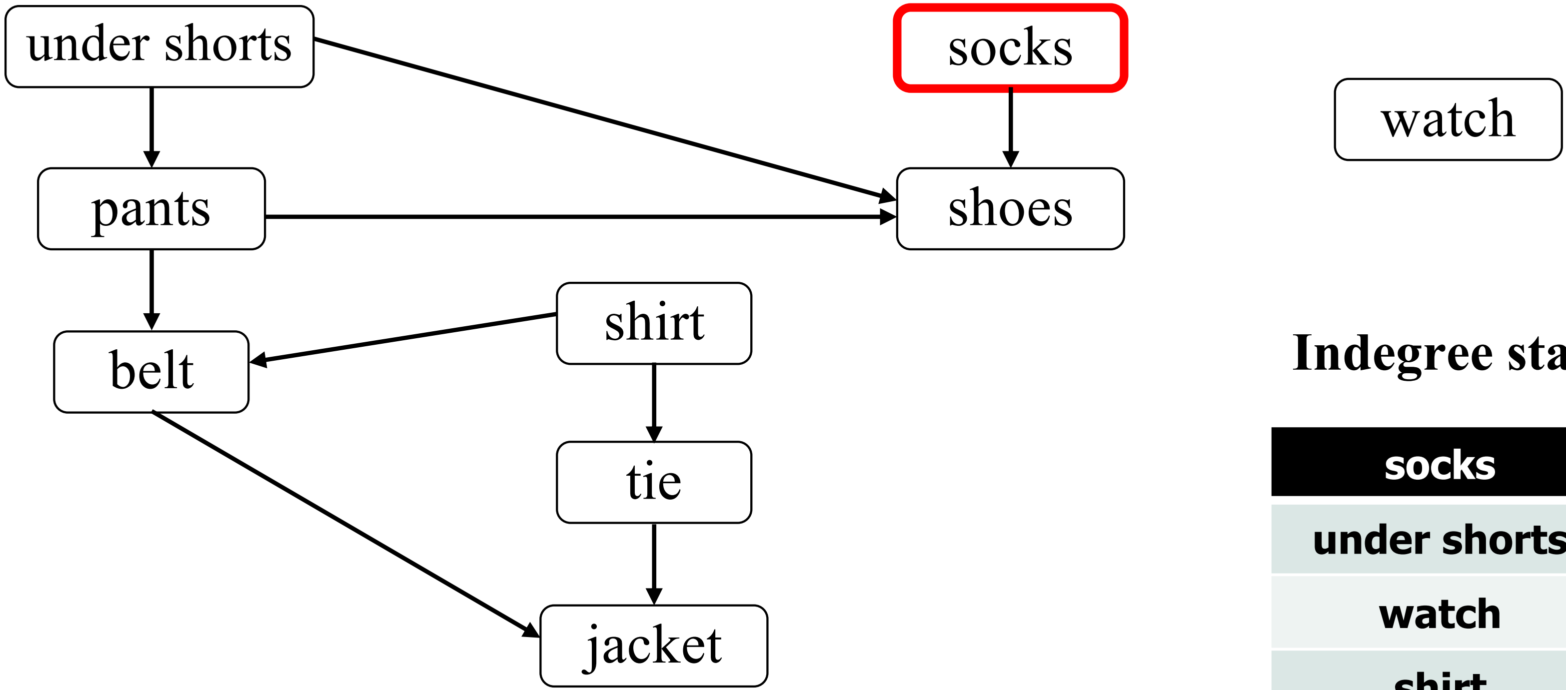


Indegree stack

socks	0
under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	3	0	0	2	1	2

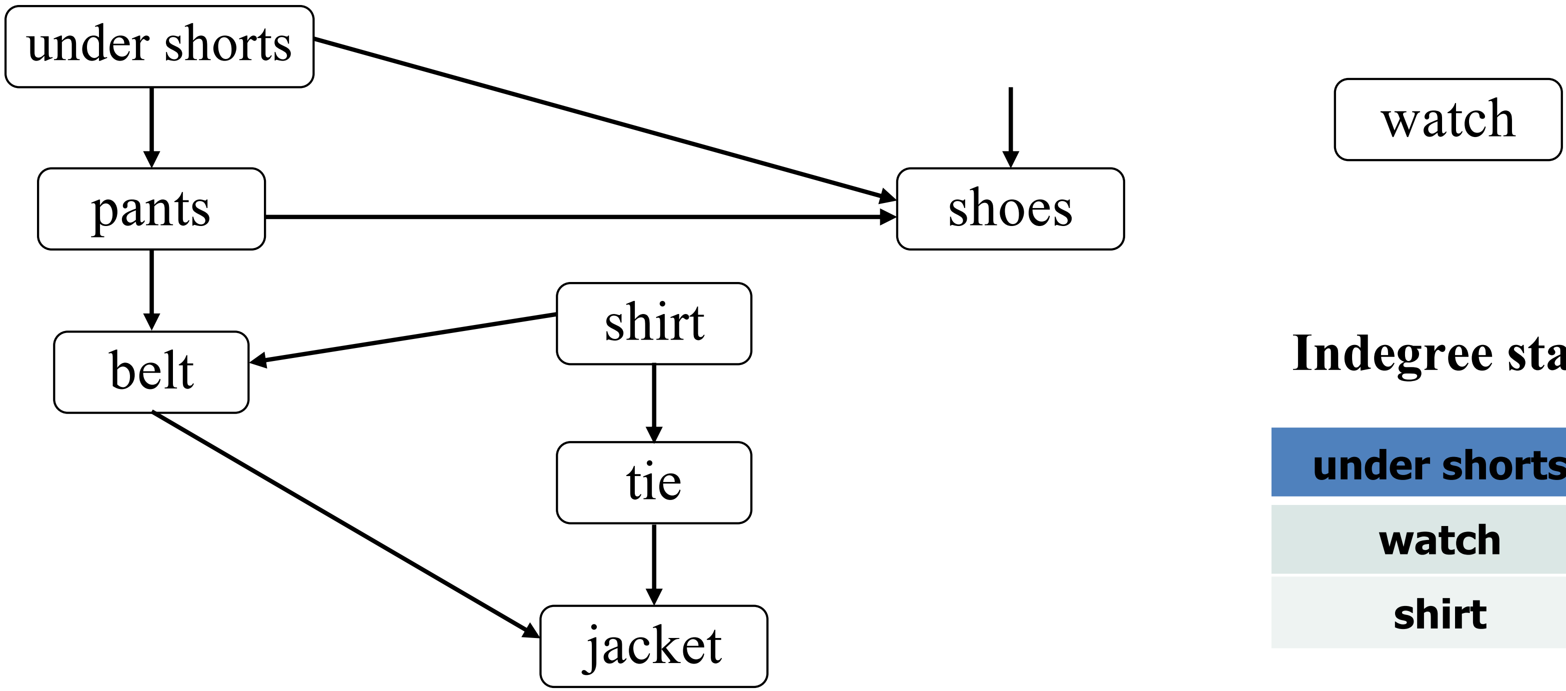


Indegree stack

socks	0
under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	3	0	0	2	1	2

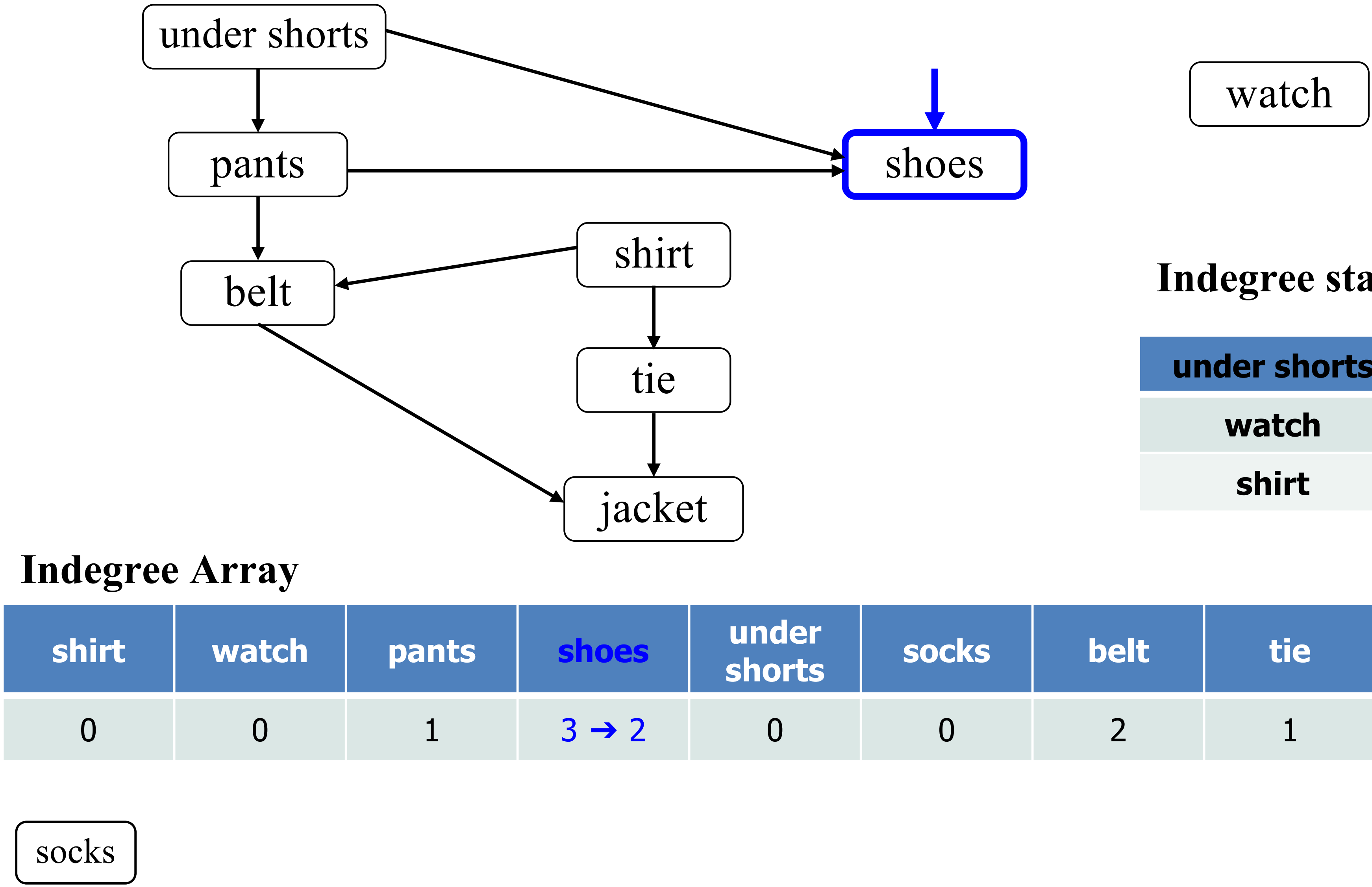


Indegree stack

under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	3	0	0	2	1	2

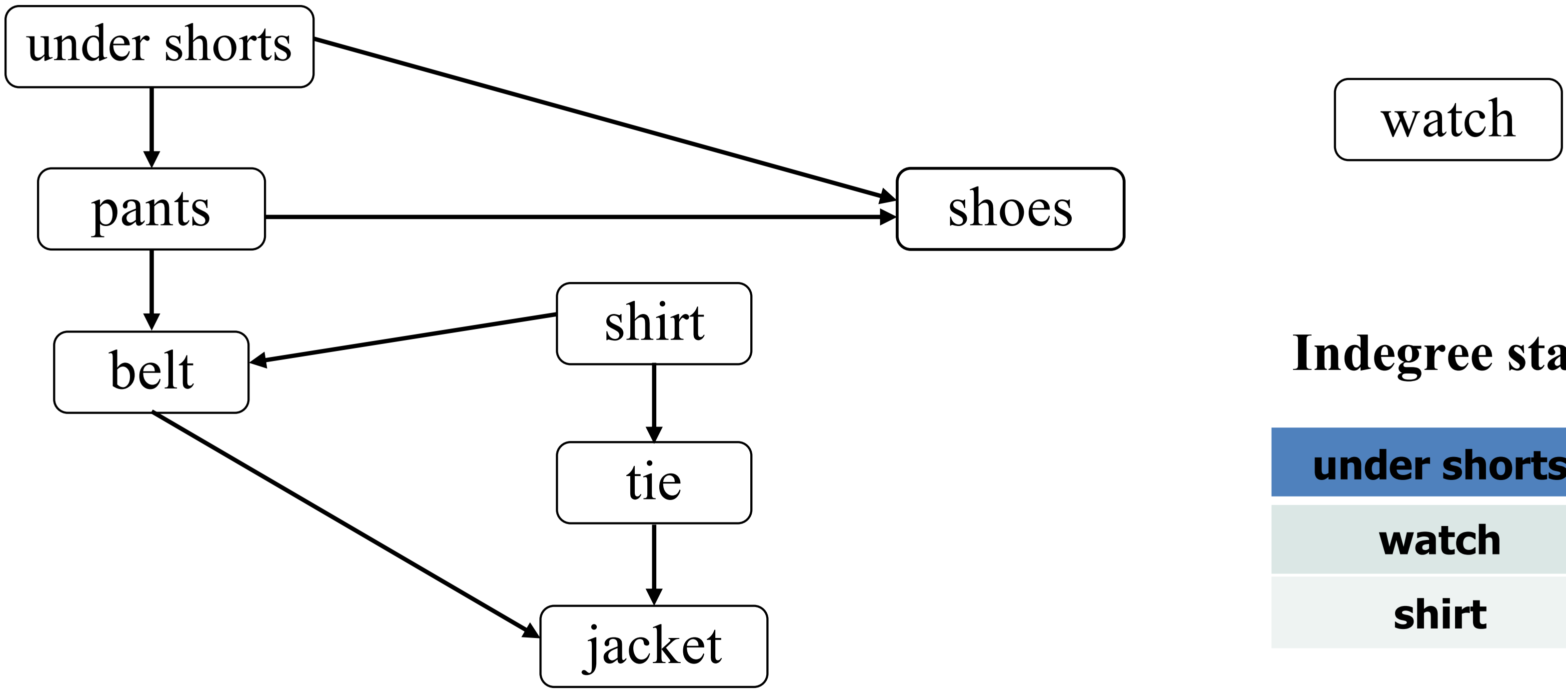


Indegree stack

under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	3 → 2	0	0	2	1	2

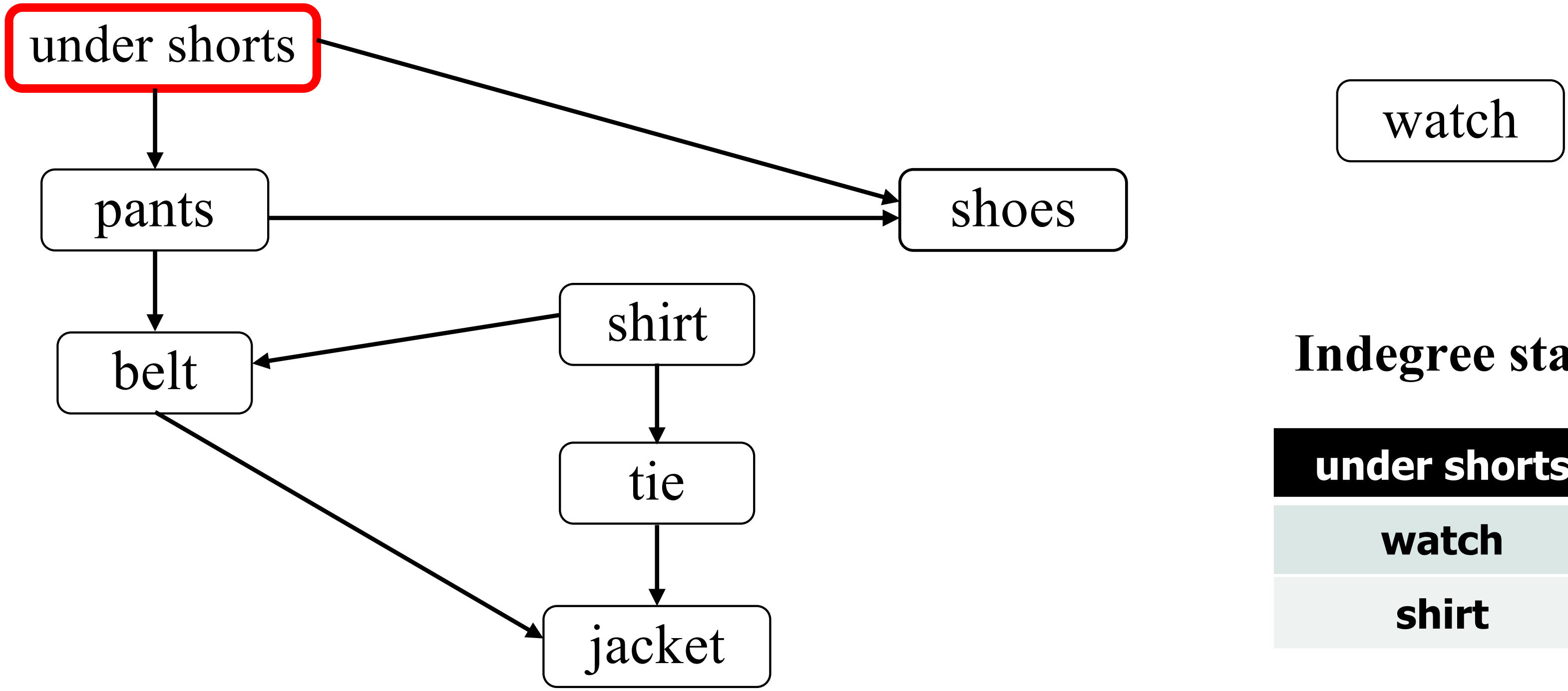


Indegree stack

under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	2	0	0	2	1	2

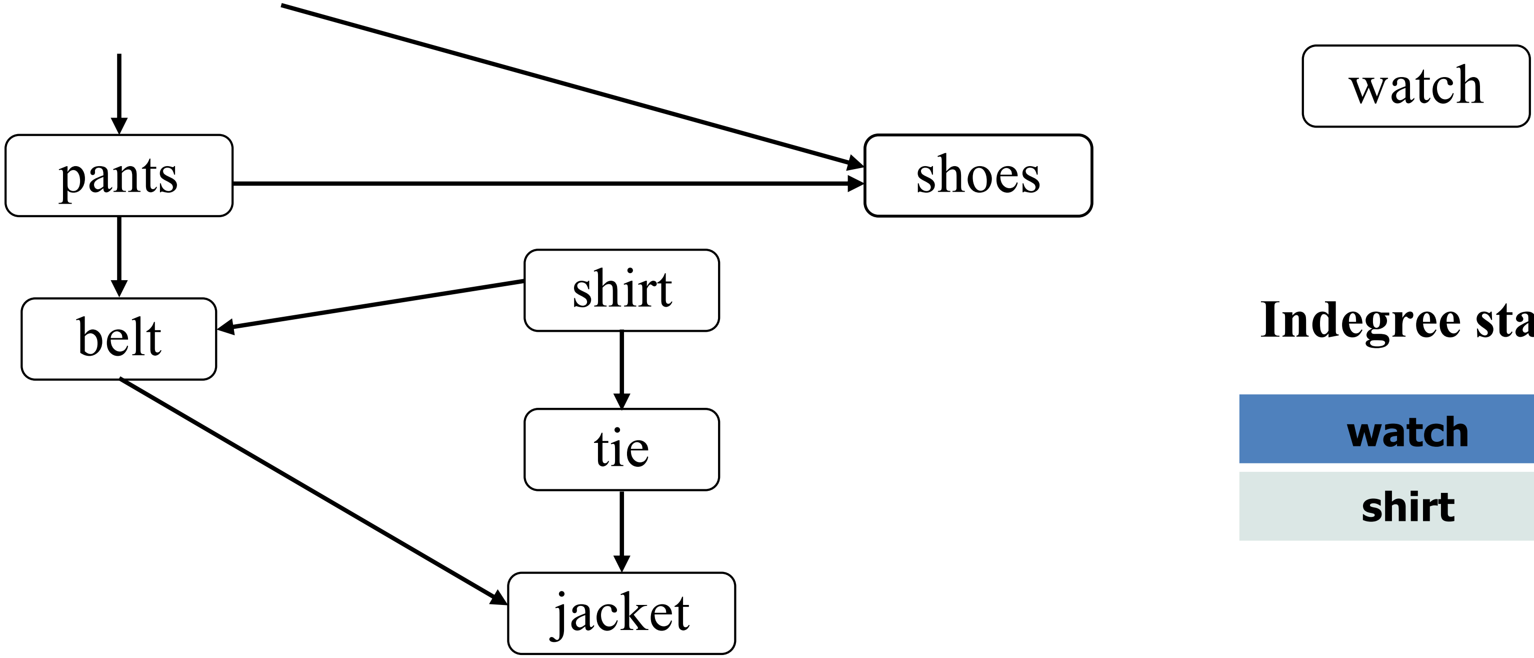


Indegree stack

under shorts	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	2	0	0	2	1	2

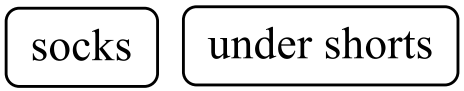


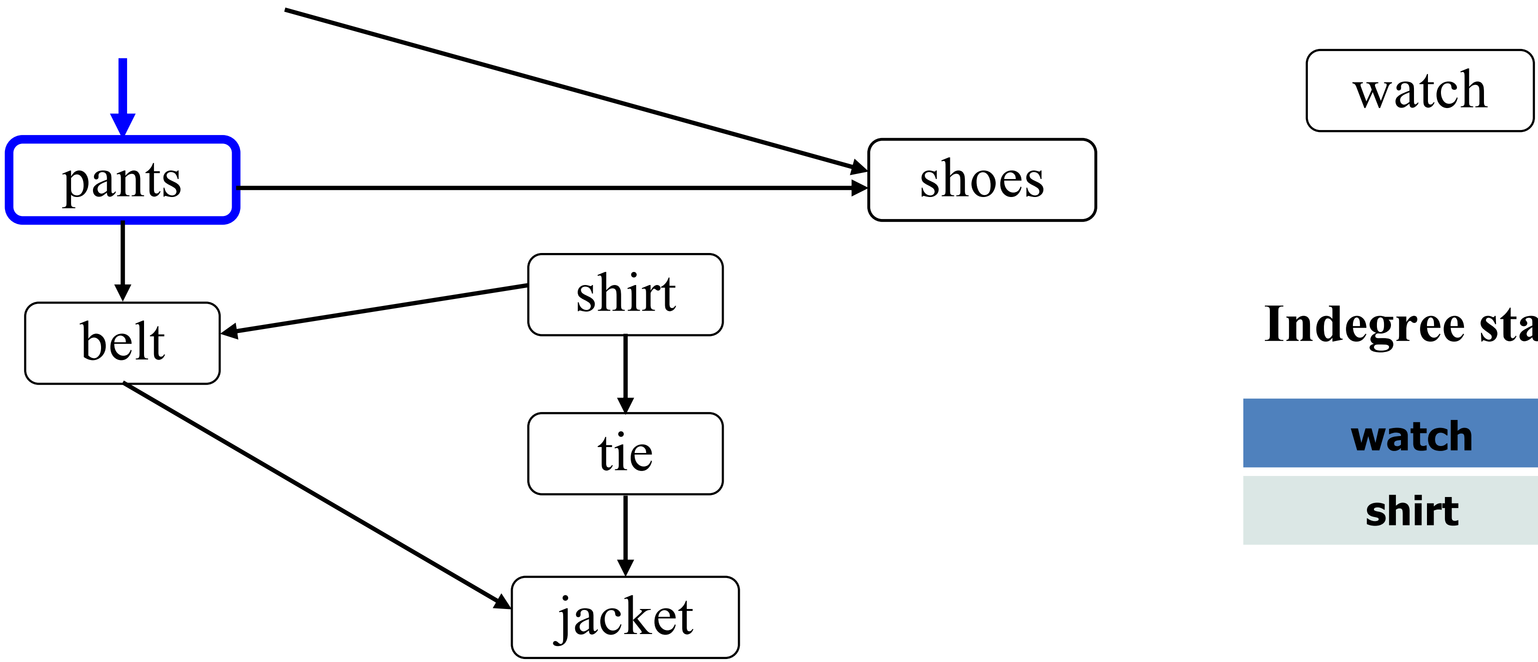
Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1	2	0	0	2	1	2



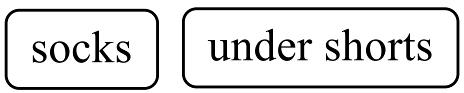


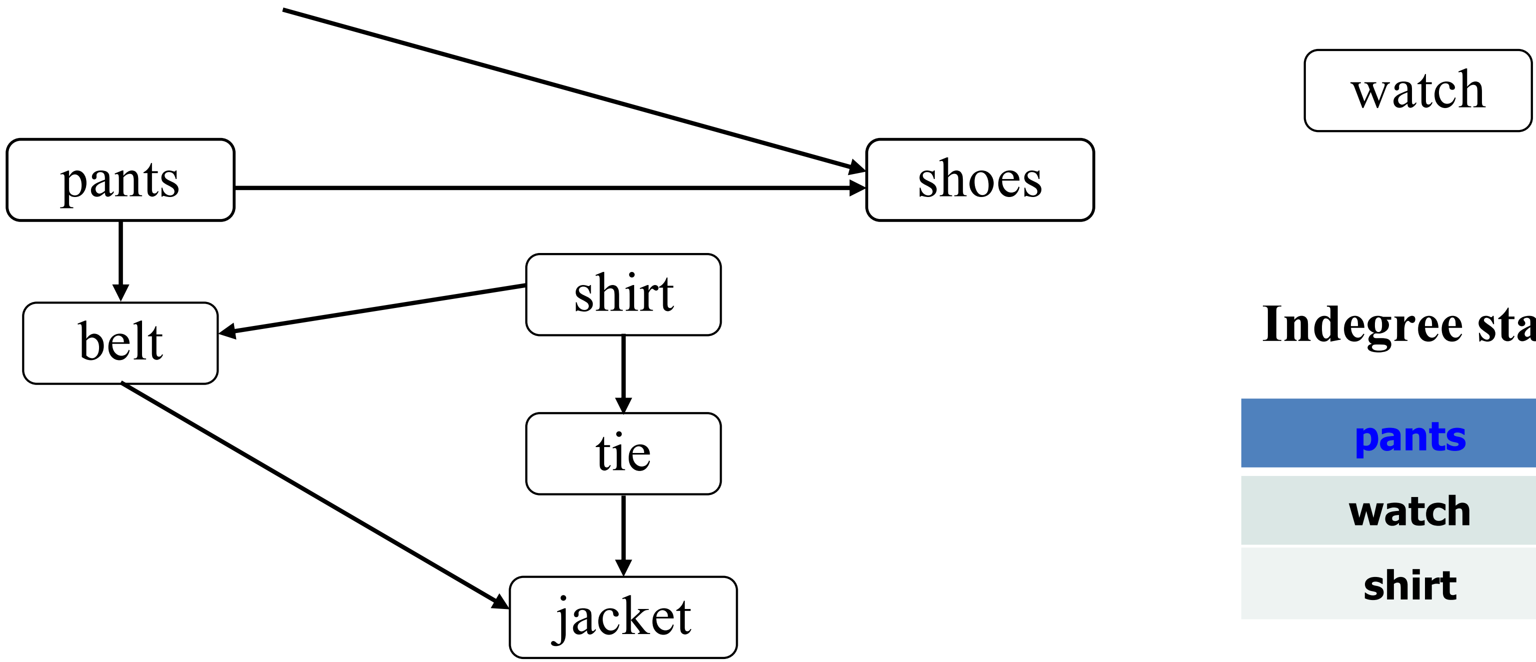
Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	1 → 0	2	0	0	2	1	2



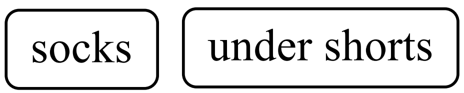


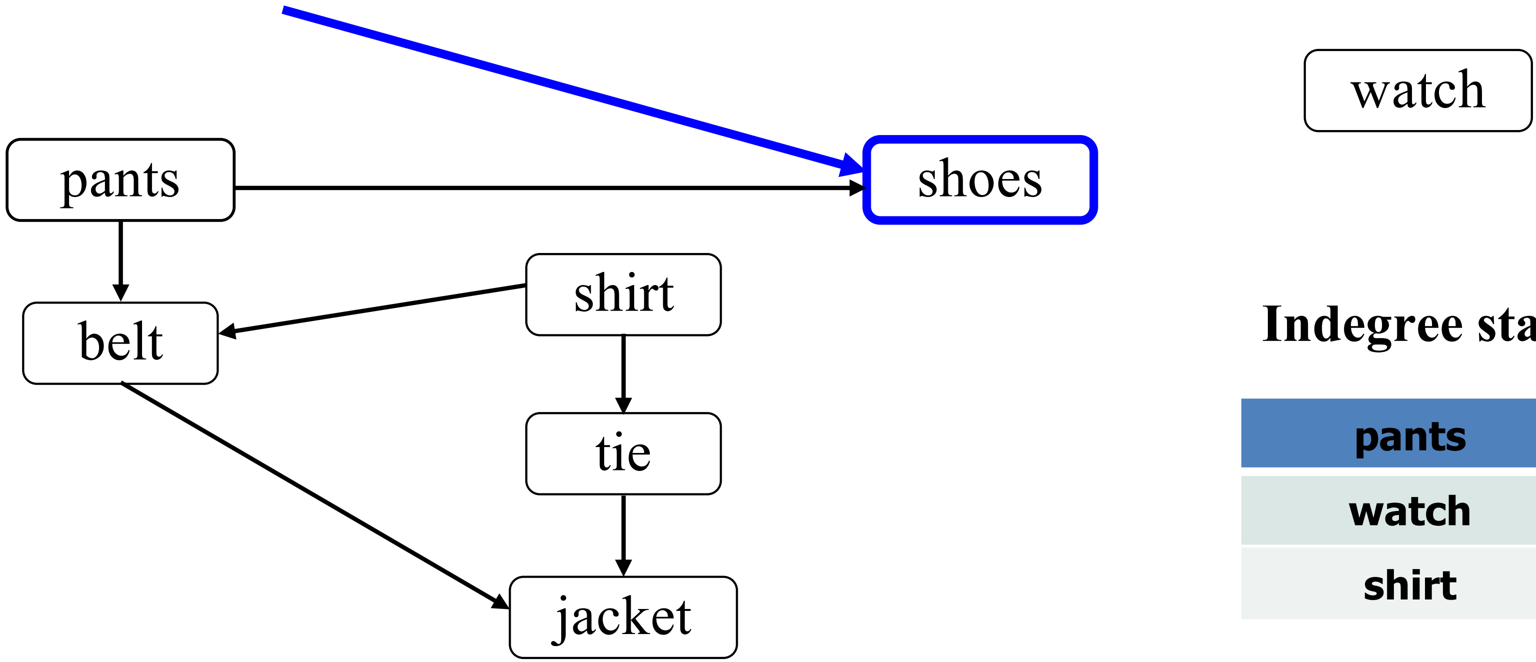
Indegree stack

pants	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	2	0	0	2	1	2



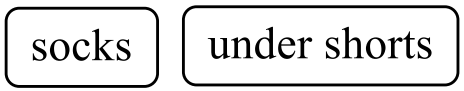


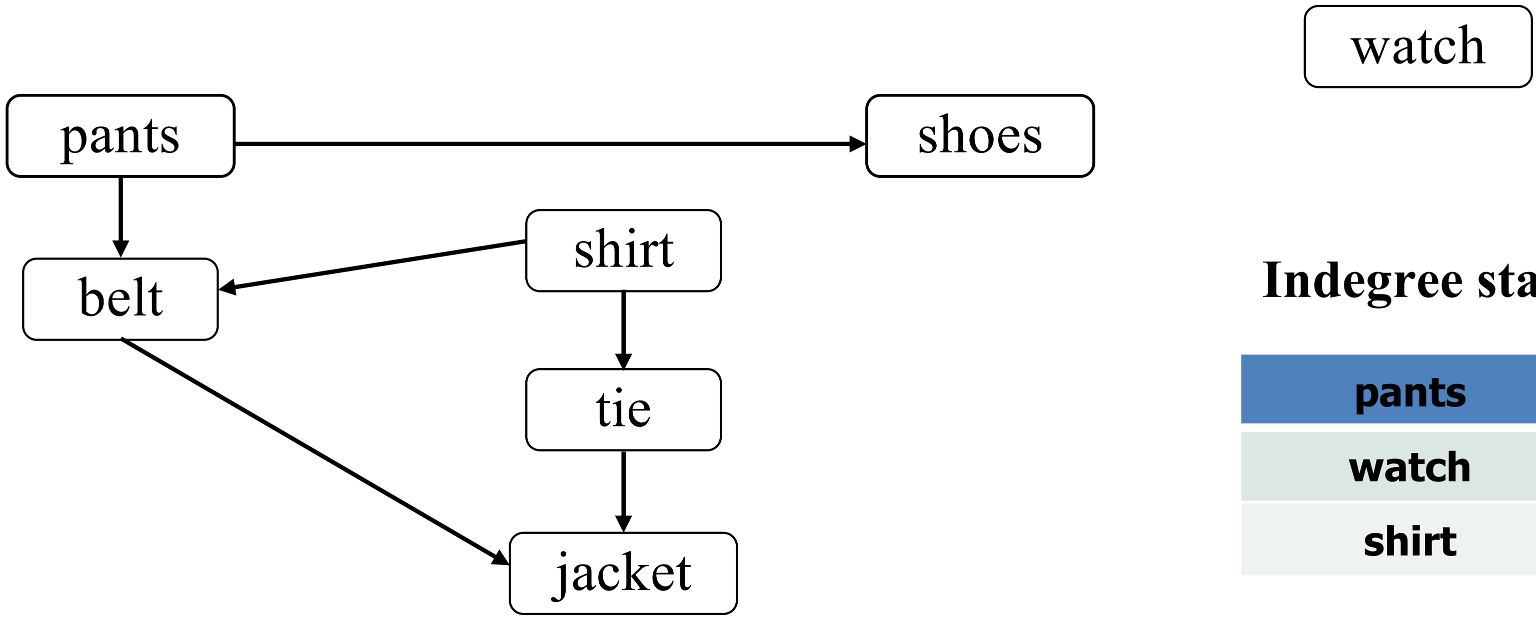
Indegree stack

pants	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	2 → 1	0	0	2	1	2



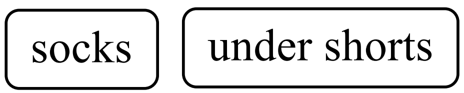


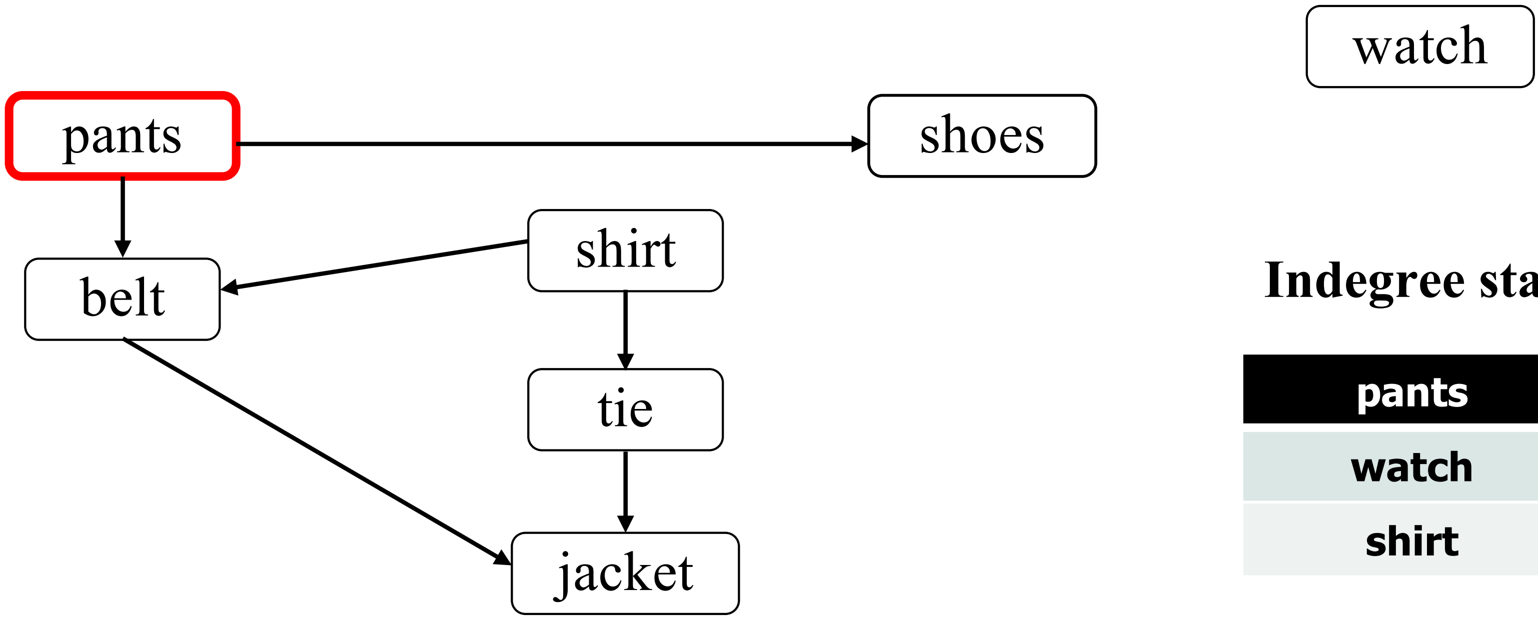
Indegree stack

pants	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	1	0	0	2	1	2



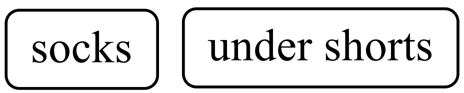


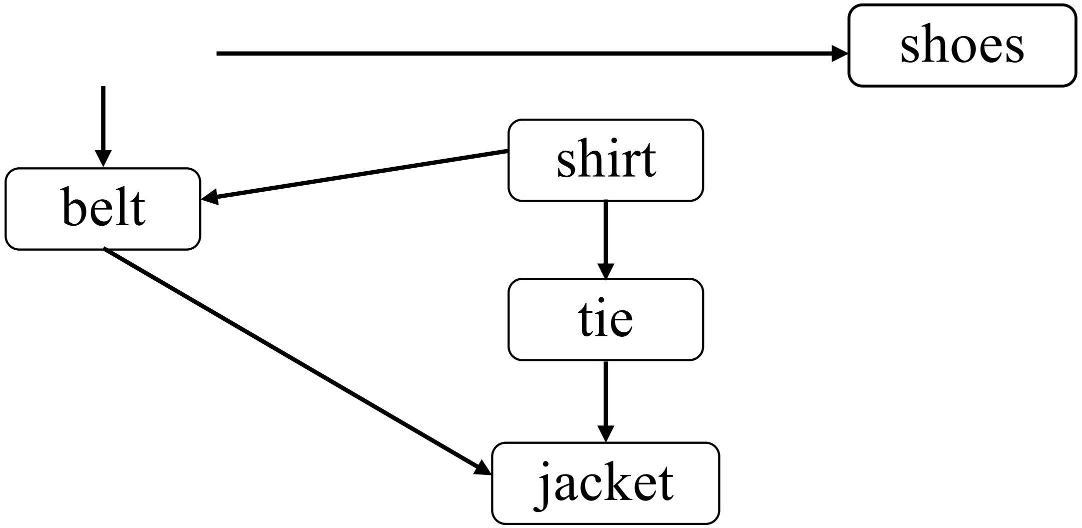
Indegree stack

pants	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	1	0	0	2	1	2



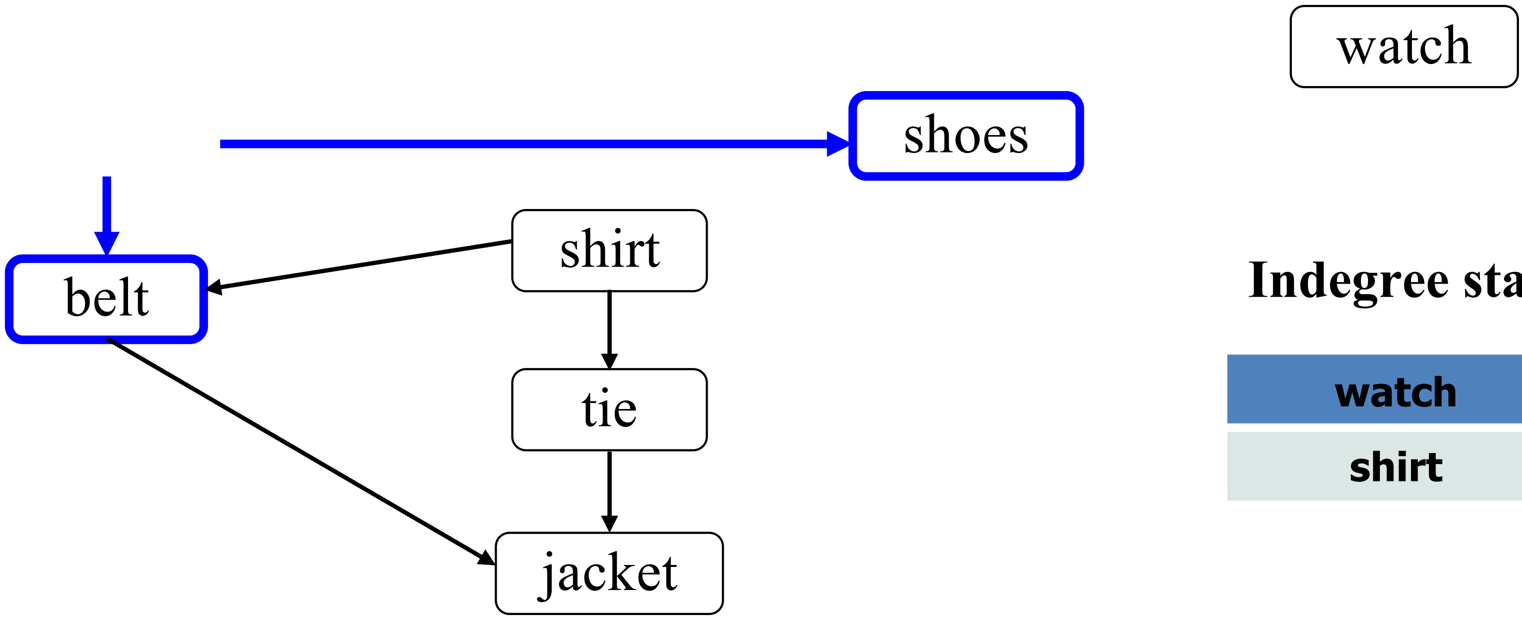


Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	1	0	0	2	1	2

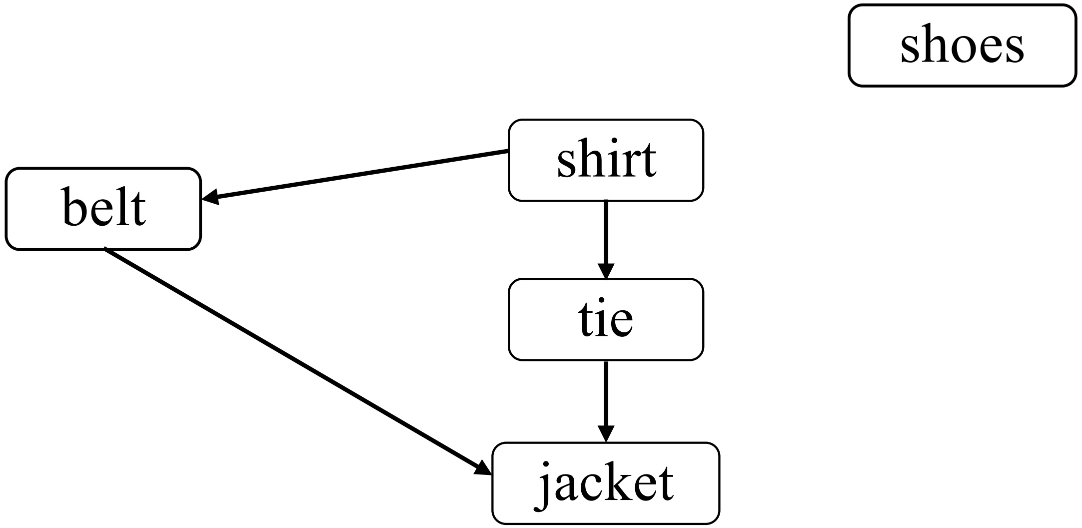


Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	1 ➔ 0	0	0	2 ➔ 1	1	2

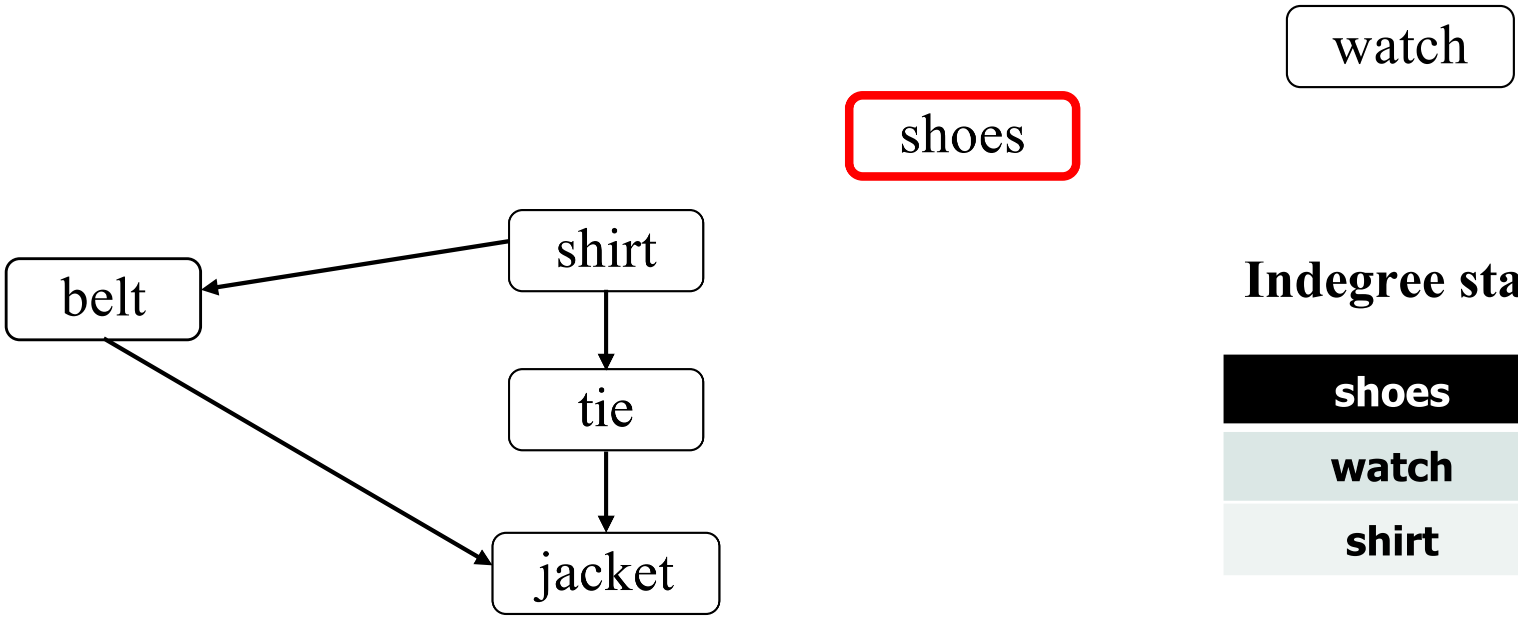


Indegree stack

shoes	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2



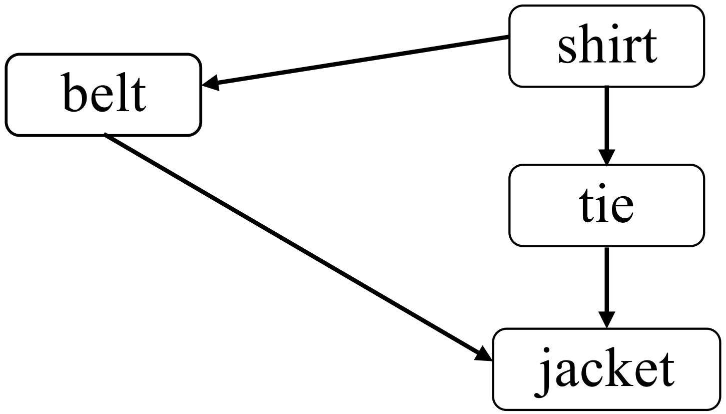
Indegree stack

shoes	0
watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2

socks under shorts pants



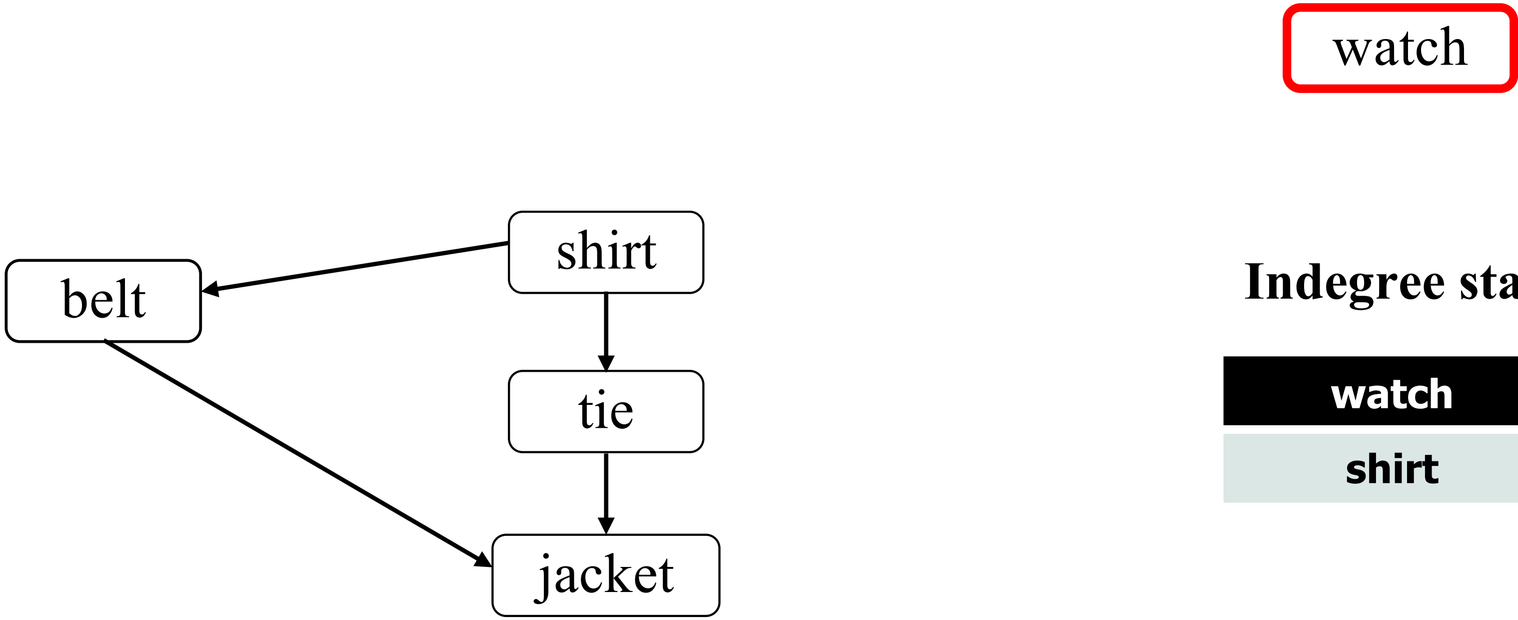
Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2

socks | under shorts | pants | shoes

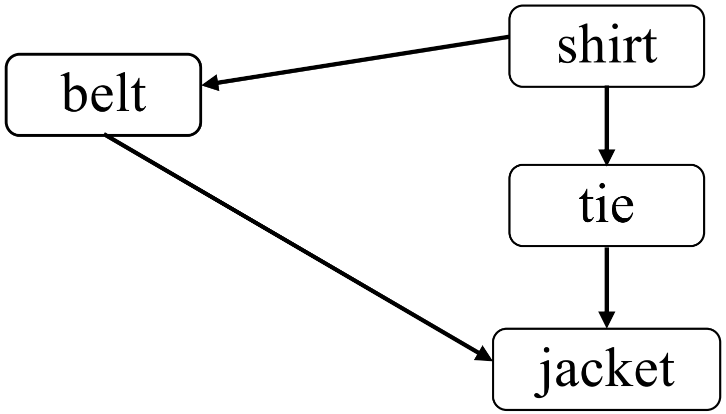


Indegree stack

watch	0
shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2



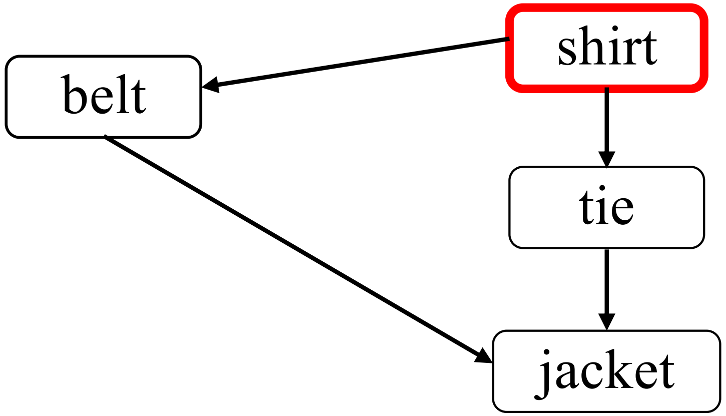
Indegree stack

shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2

socks | under shorts | pants | shoes | watch



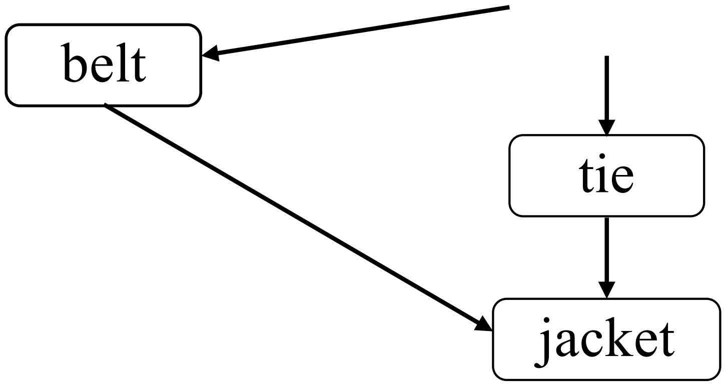
Indegree stack

shirt	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2

socks | under shorts | pants | shoes | watch

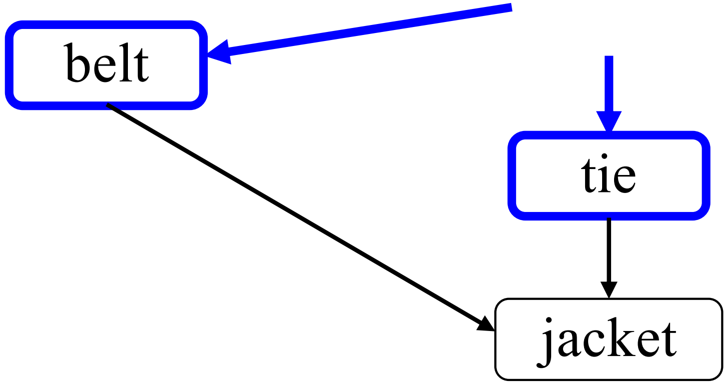


Indegree stack

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1	1	2

socks | under shorts | pants | shoes | watch | shirt

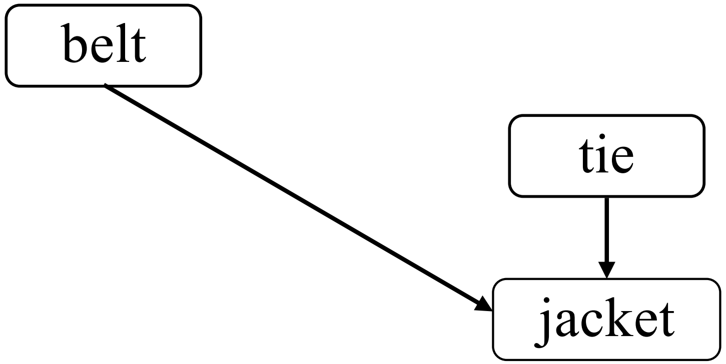


Indegree stack

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	1 → 0	1 → 0	2

socks | under shorts | pants | shoes | watch | shirt



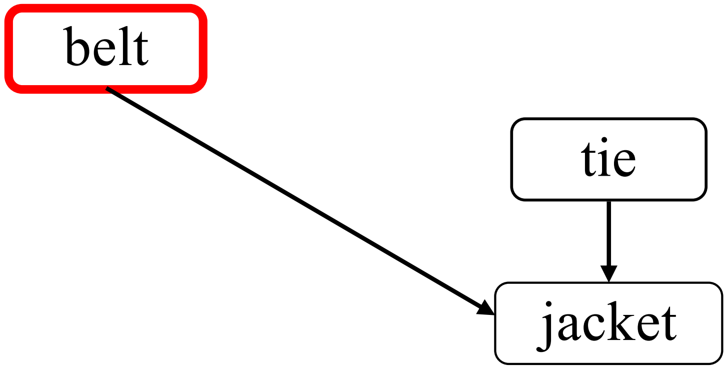
Indegree stack

belt	0
tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	2

socks | under shorts | pants | shoes | watch | shirt



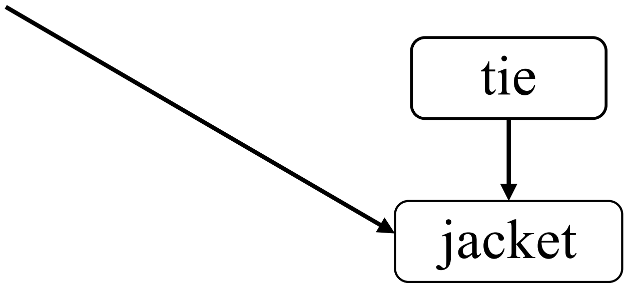
Indegree stack

belt	0
tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	2

socks under shorts pants shoes watch shirt



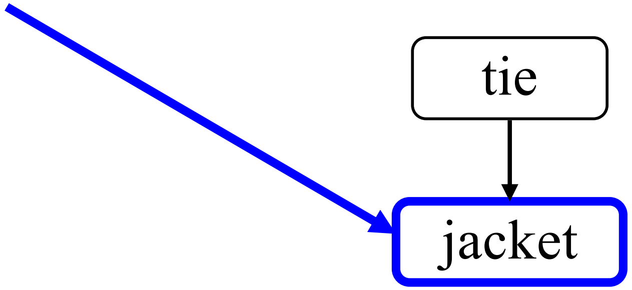
Indegree stack

tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	2

socks | under shorts | pants | shoes | watch | shirt | belt



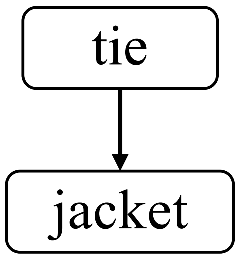
Indegree stack

tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	2 → 1

socks | under shorts | pants | shoes | watch | shirt | belt



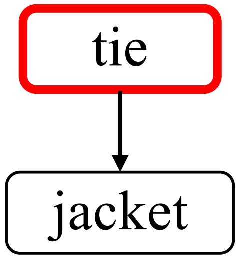
Indegree stack

tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	1

socks | under shorts | pants | shoes | watch | shirt | belt



Indegree stack

tie	0

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	1

socks | under shorts | pants | shoes | watch | shirt | belt

Topological sort

Indegree stack

jacket

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	1

socks | under shorts | pants | shoes | watch | shirt | belt | tie

Indegree stack

jacket

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	1 → 0

socks | under shorts | pants | shoes | watch | shirt | belt | tie

Indegree stack

jacket	0

jacket

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	0

socks under shorts pants shoes watch shirt belt tie

Indegree stack

jacket	0

jacket

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	0

socks | under shorts | pants | shoes | watch | shirt | belt | tie

Indegree stack

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	0

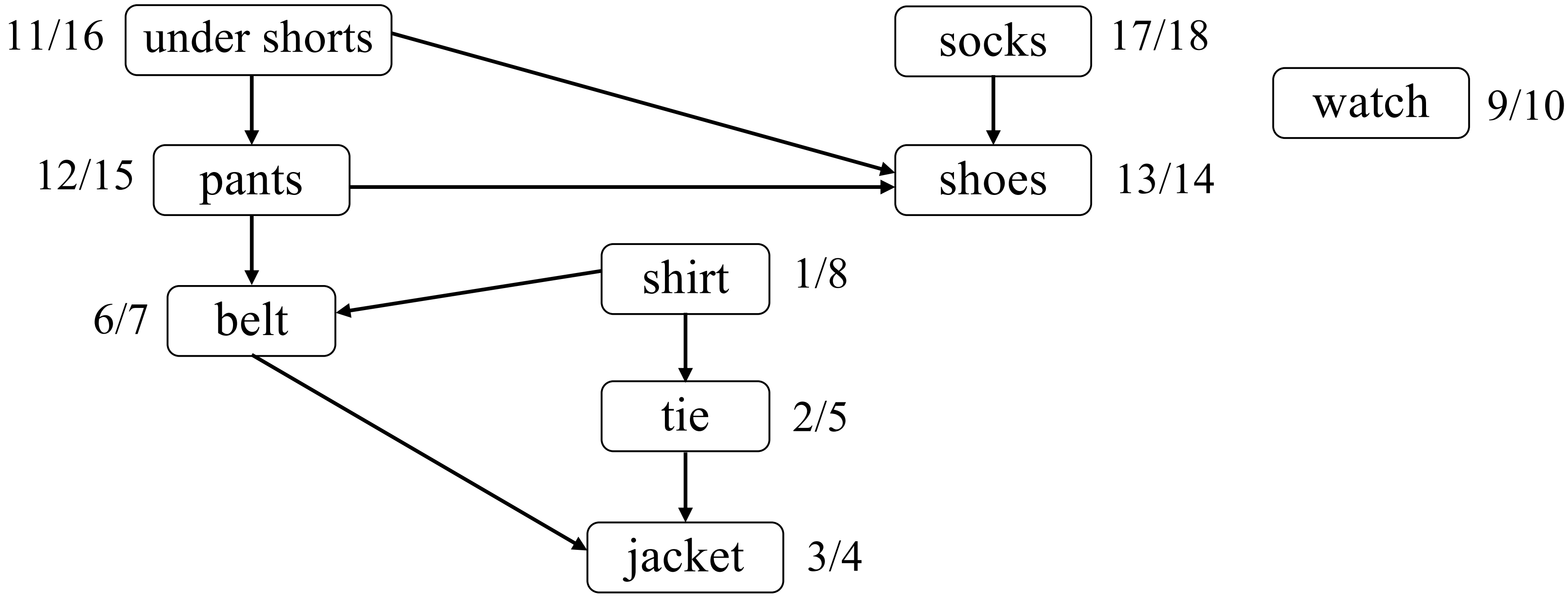
socks | under shorts | pants | shoes | watch | shirt | belt | tie | jacket

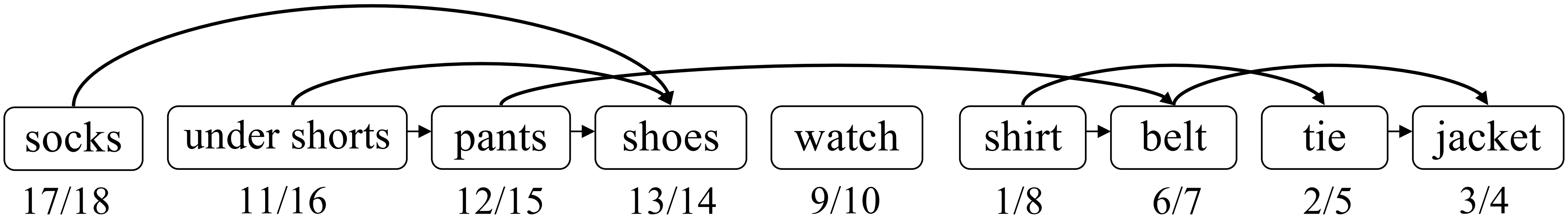
Time Complexity = O(V+E)

Indegree Array

shirt	watch	pants	shoes	under shorts	socks	belt	tie	jacket
0	0	0	0	0	0	0	0	0

socks | under shorts | pants | shoes | watch | shirt | belt | tie | jacket


11/16
under shorts
socks
17/18
watch
9/10
12/15
pants
shoes
13/14
shirt
1/8
6/7
belt
tie
2/5
jacket
3/4


socks
under shorts
pants
shoes
watch
shirt
belt
tie
jacket
17/18
11/16
12/15
13/14
9/10
1/8
6/7
2/5
3/4

- ***Correctness***
  - If there is an edge from u to v, then $v.f < u.f$.
  - A directed graph G is ***acyclic*** if and only if a depth-first search of G yields ***no back edges***.

- ***Main ideas***
  - Successively place a node from the ***left*** with **0 *in-degree***.
  - Successively place a node from the ***right*** with **0 *out-degree***.
  - Run DFS on G and place the nodes from the ***right*** in the ***increasing order of the finishing time***.
  - $\Theta(V+E)$ time

- **Exercise 22.4-2**
  - Computing the number of simple paths from s to t in linear time.

- **Exercise 22.4-3**
  - Cycle detection in an undirected graph.

- **Exercise 22.4-5**
  - Another topological sort algorithm.

- ***Depth-first search and its applications***
  - **Exercise 22.3-10 (22.3-9, 2nd ed.) (#1)**
    - Depth-first search with edge classification
  - **Exercise 22.3-12 (22.3-11, 2nd ed.) (#2)**
    - Connected component identification
  - **Topological sort (#3)**
    - The program should detect whether the input is a DAG or not.