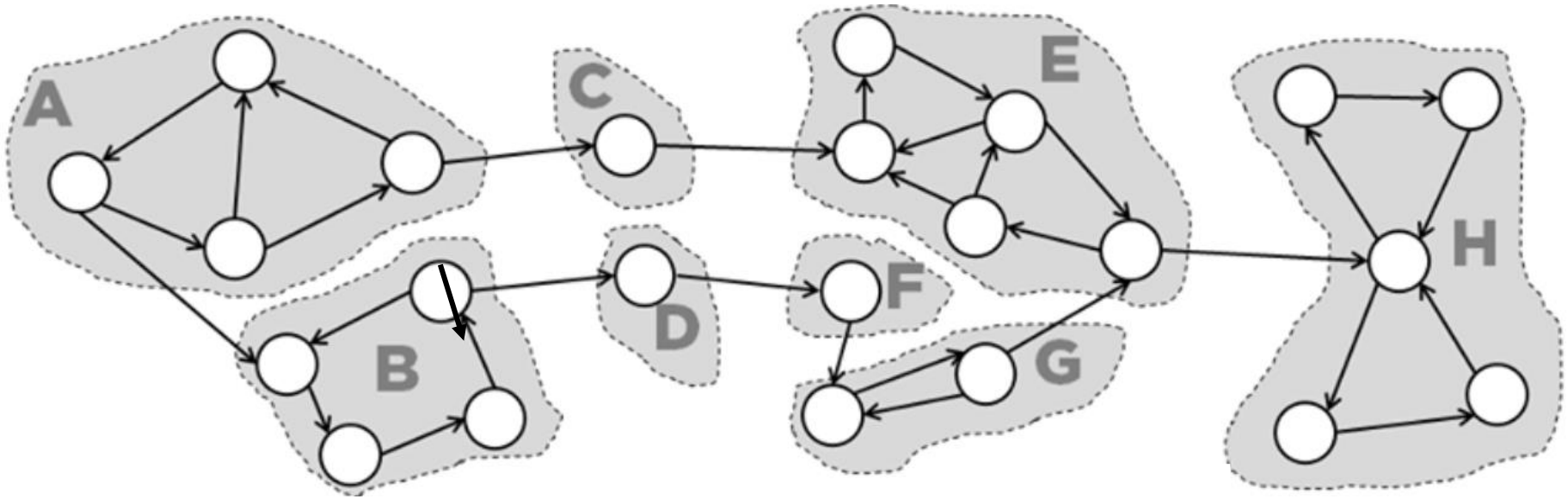
Strongly Connected Components with DFS

- Tarjan 1972 (wikipedia)
- Performs a single DFS with an additional **S**tack

Strongly connected components (SCC)

Def: Equivalence classes of the relation (u, v) iff there is a path from u to v and a path from v to u



Goal: Compute for each vertex the number of its component

Strongly connected components (SCC)

Obs: Let u and v be in a SC C. Let P be a path from u to v then every vertex on P is also in C

```
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 vertex v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-Visit(G,v)
u.color = BLACK
time = time + 1
u.f = time
```

Thm 2: $[v.d, v.f] \subset [u.d, u.f]$ iff v is a descendant of u in the DFS-forest, $[v.d, v.f] \cap [u.d, u.f] = \emptyset$ iff u and v are unrelated in the DFS-forest

```
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 vertex v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-Visit(G,v)
u.color = BLACK
time = time + 1
u.f = time
```

Thm 3: v is a descendant of u in the DFS-forest iff when we invoke DFS-Visit(u) there is a white path from u to v

Edge classification

- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest
- Forward edges: nontree edges (u,v) such that v is a descendant of u
- Cross edges: (u,v) such that $v.f < u.d$

Observe, among non-tree edges:

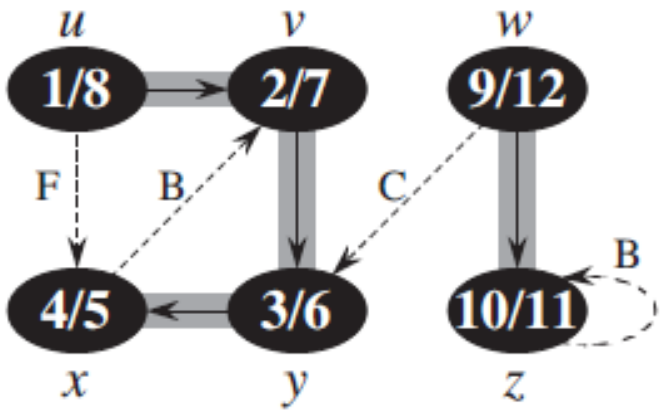
1. (Only) backward edges go to a vertex with a later finish time
2. (Only) forward edges go to a vertex with later discovery time

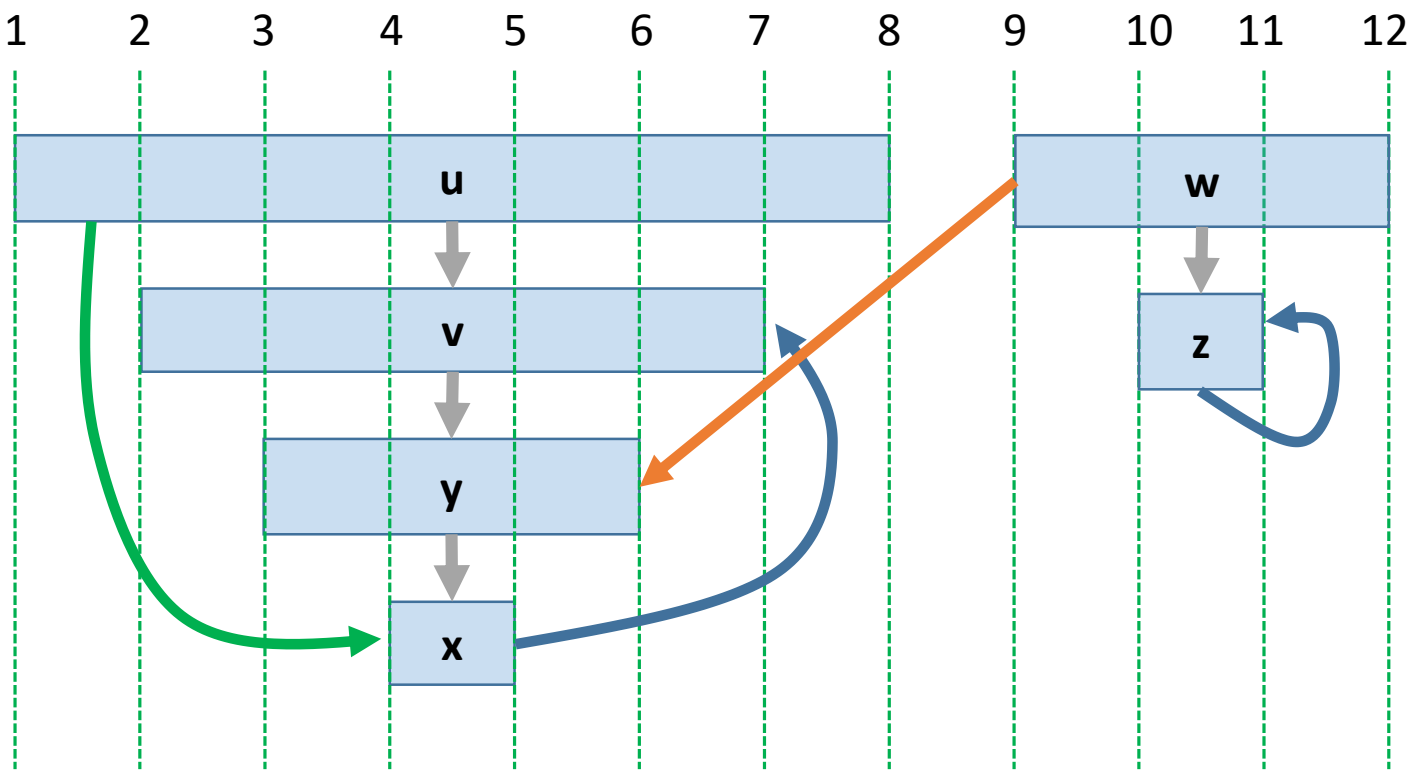
Edge classification

- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest
- Forward edges: nontree edges (u,v) such that v is a descendant of u
- Cross edges: (u,v) such that $v.f < u.d$

Observe, among non-tree edges:

1. (Only) backward edges go to a vertex with a later finish time
2. **(Only) forward edges go to a vertex with later discovery time**

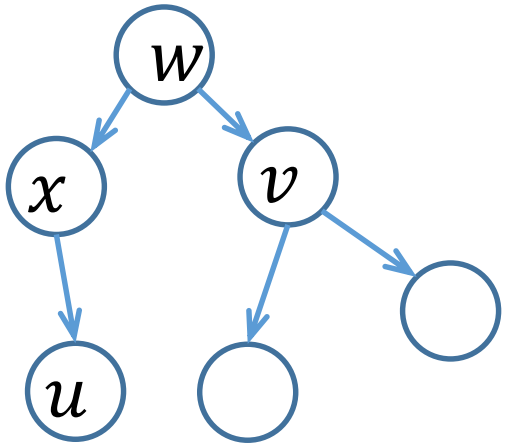

u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z


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

Tarjan's algorithm

- We will find SCCs using only one DFS

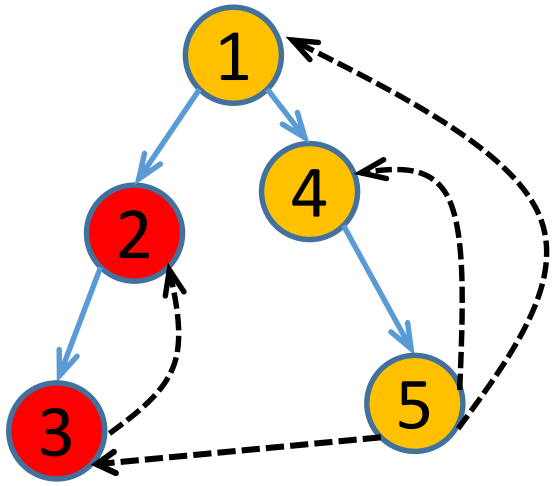
Thm 9: If u and v are in the same SC then they are in same tree of the DFS-forest and $LCA(u, v)$ is also in this component

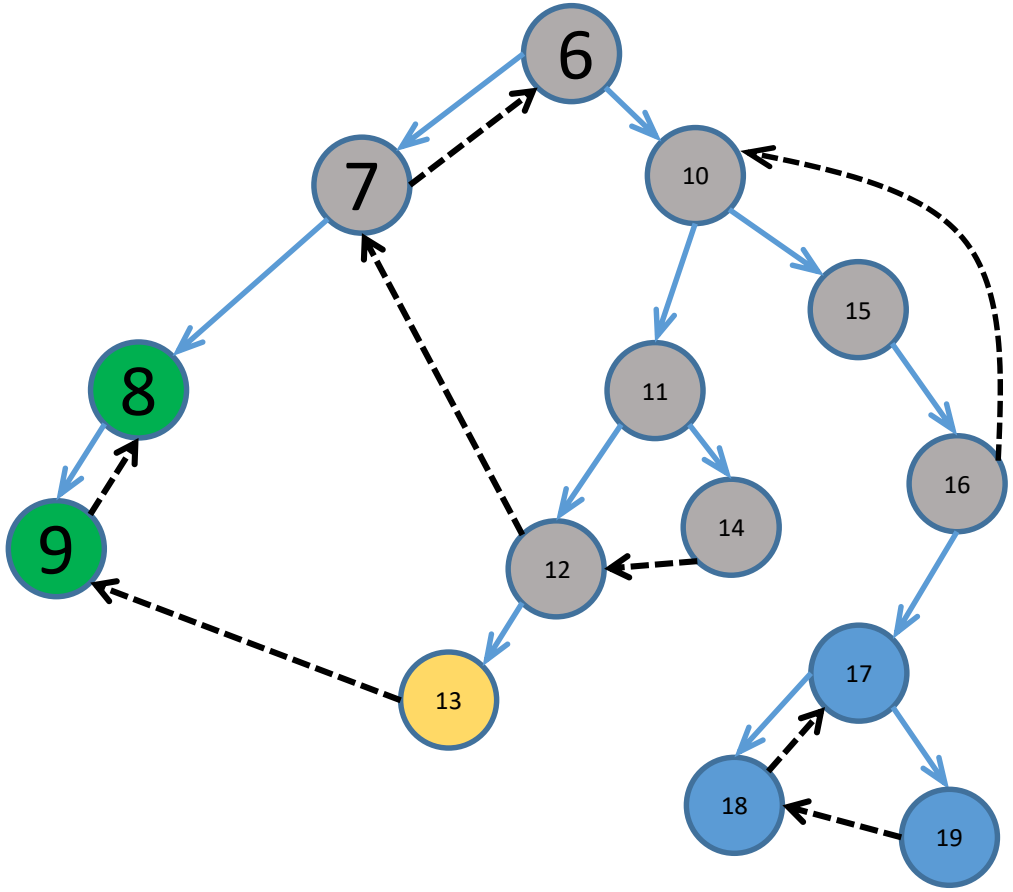


Proof: …

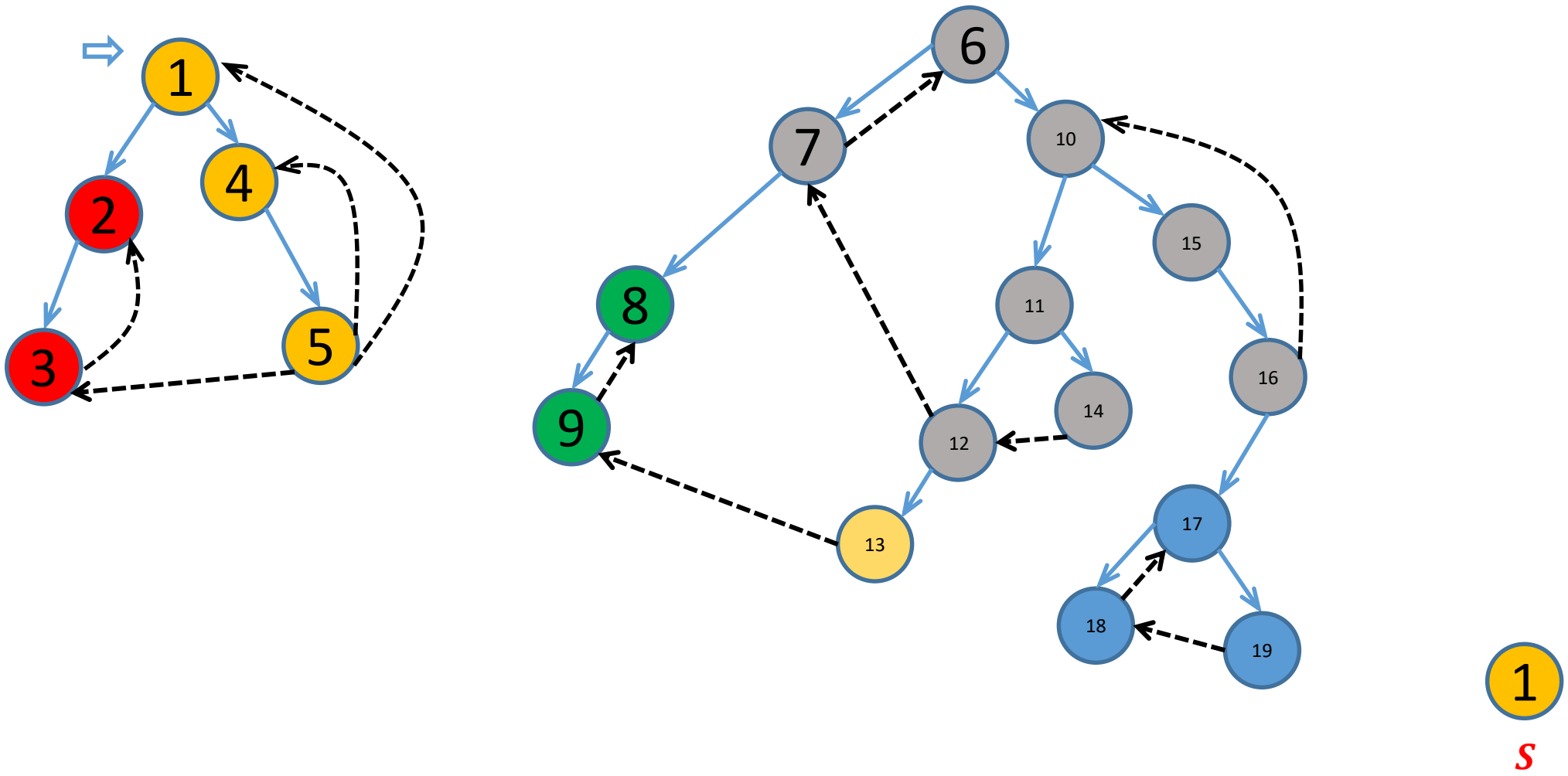
Corr: Each SC form a **connected** subtree of the DFS-forest

Identify the roots of these subtrees

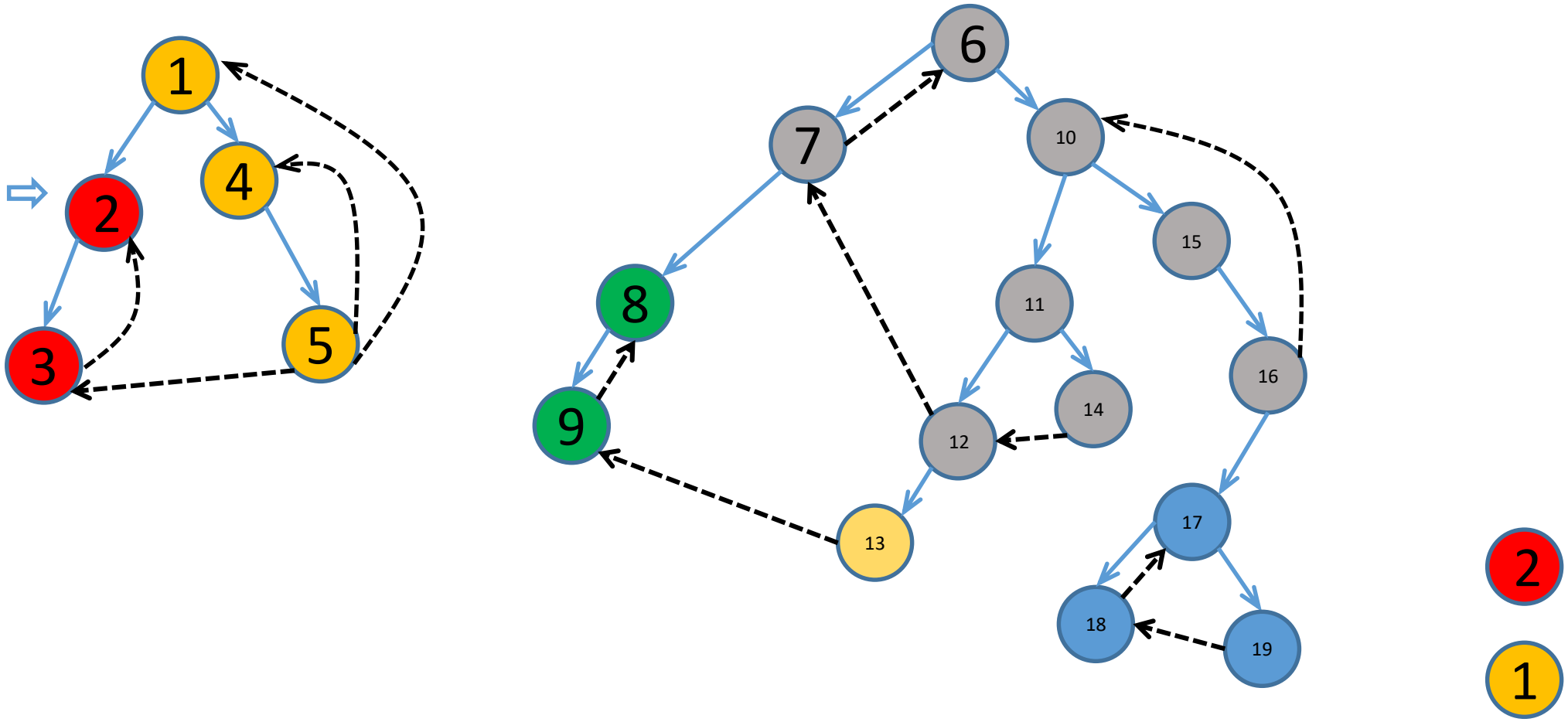




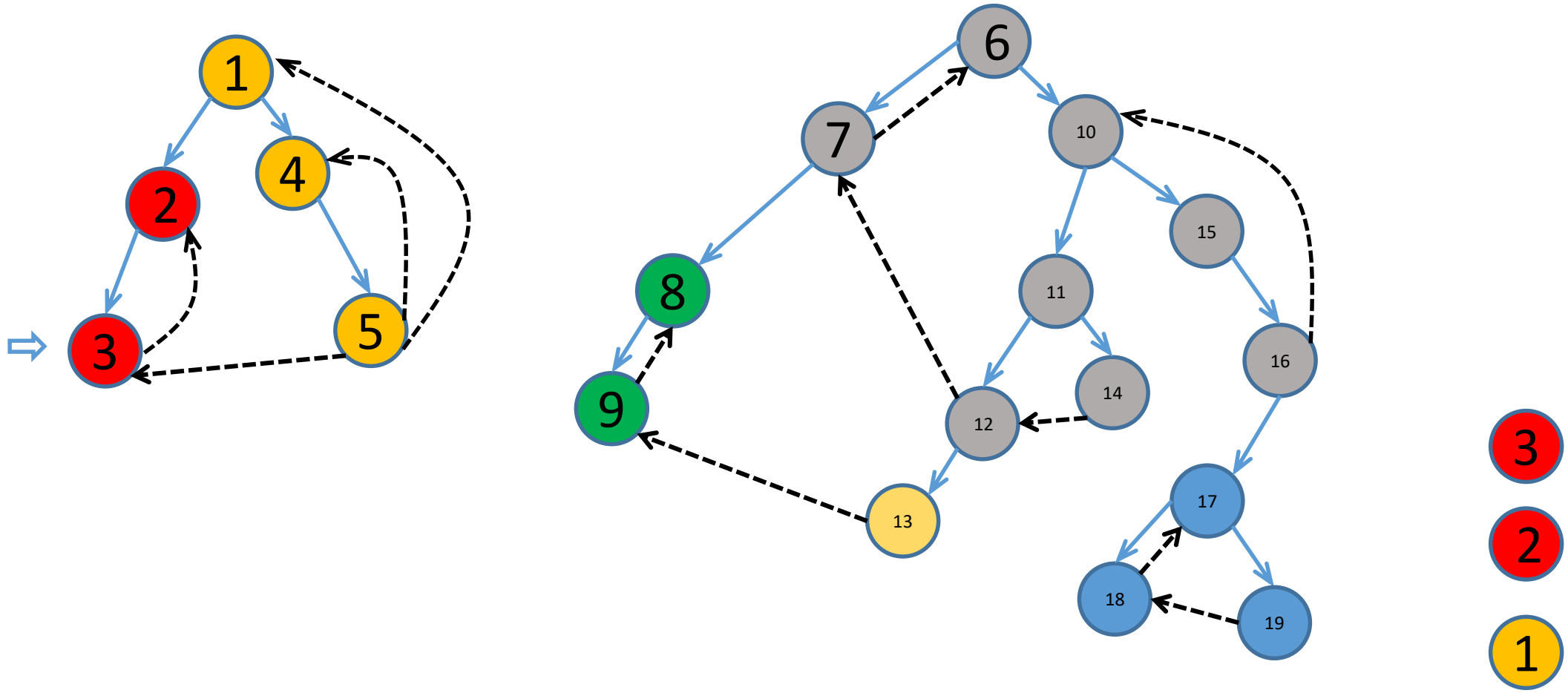
Use an additional stack (S)



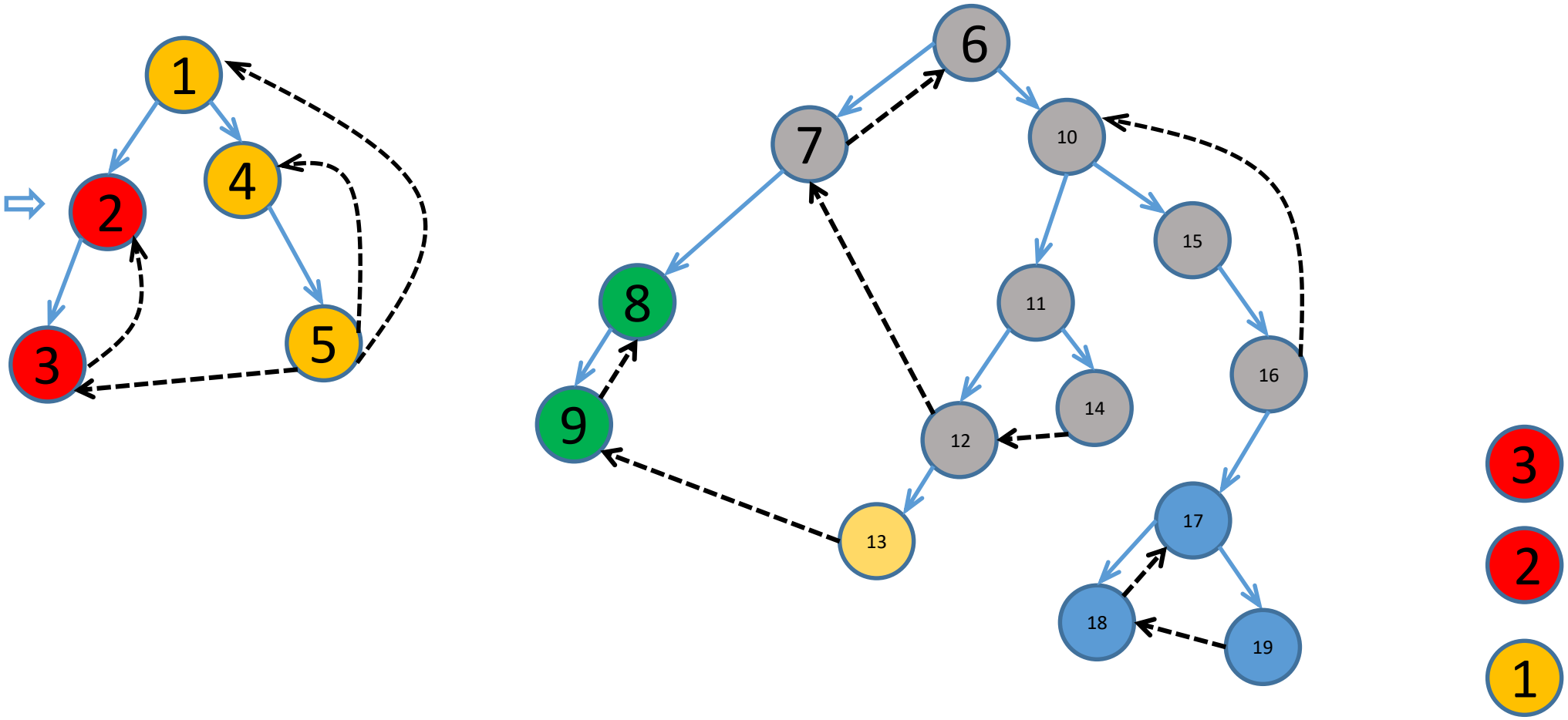
Use an additional stack



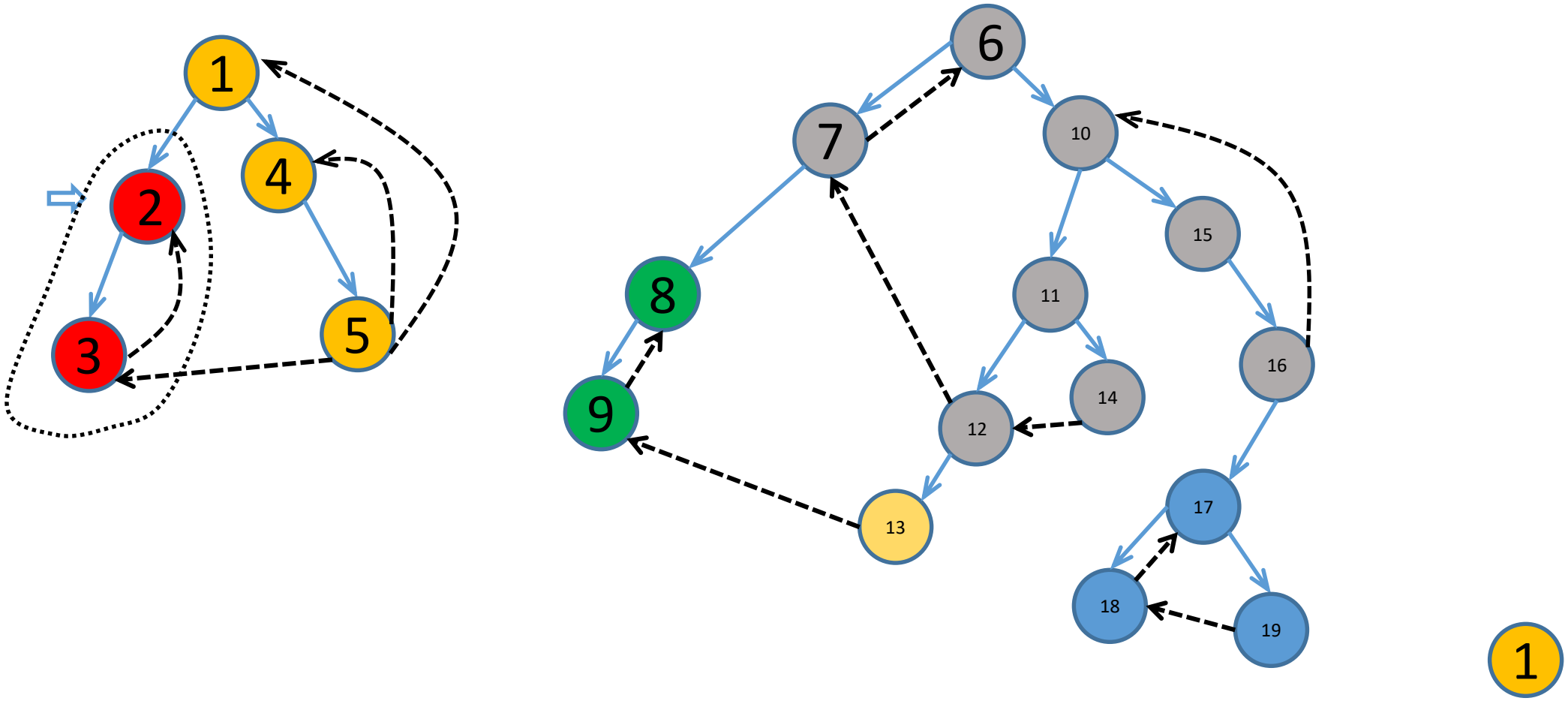
Use an additional stack



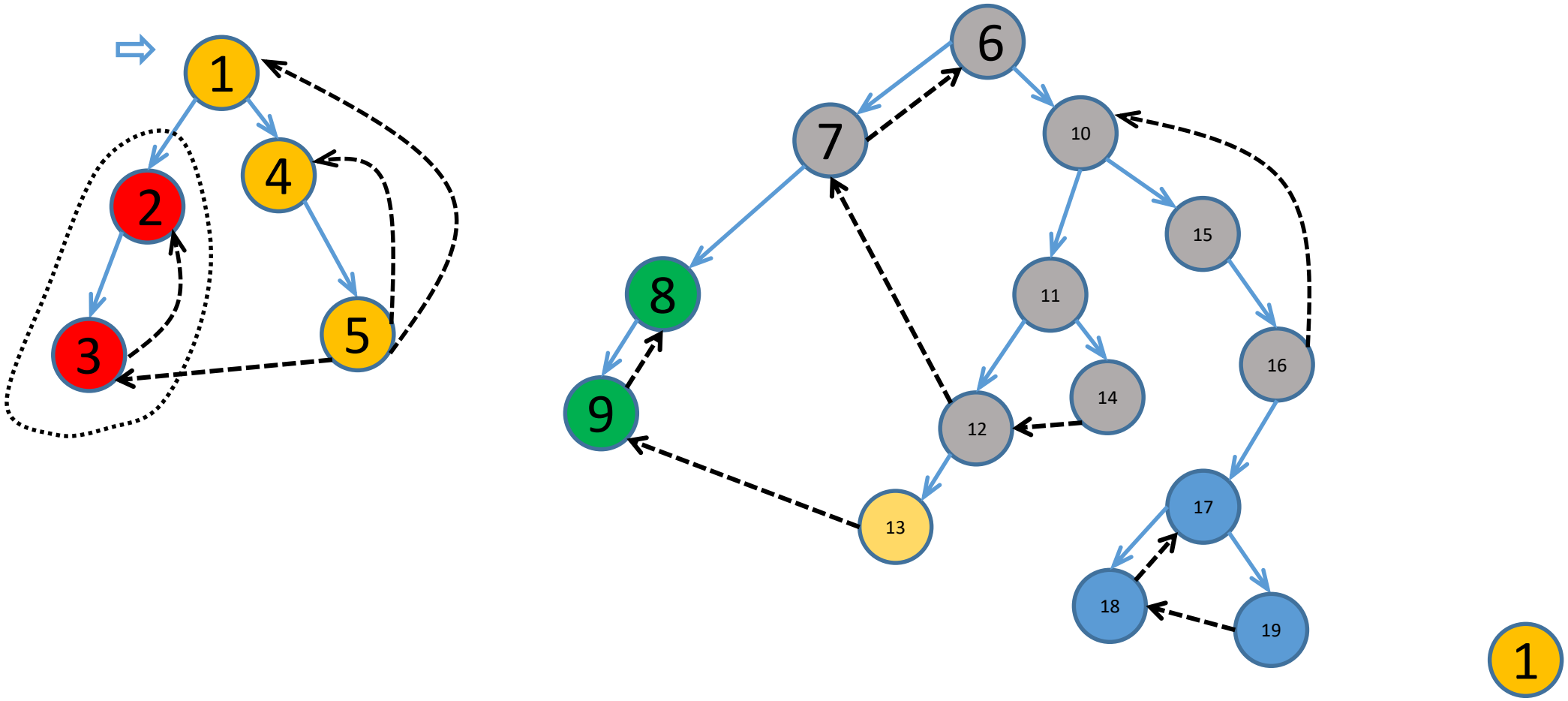
Use an additional stack



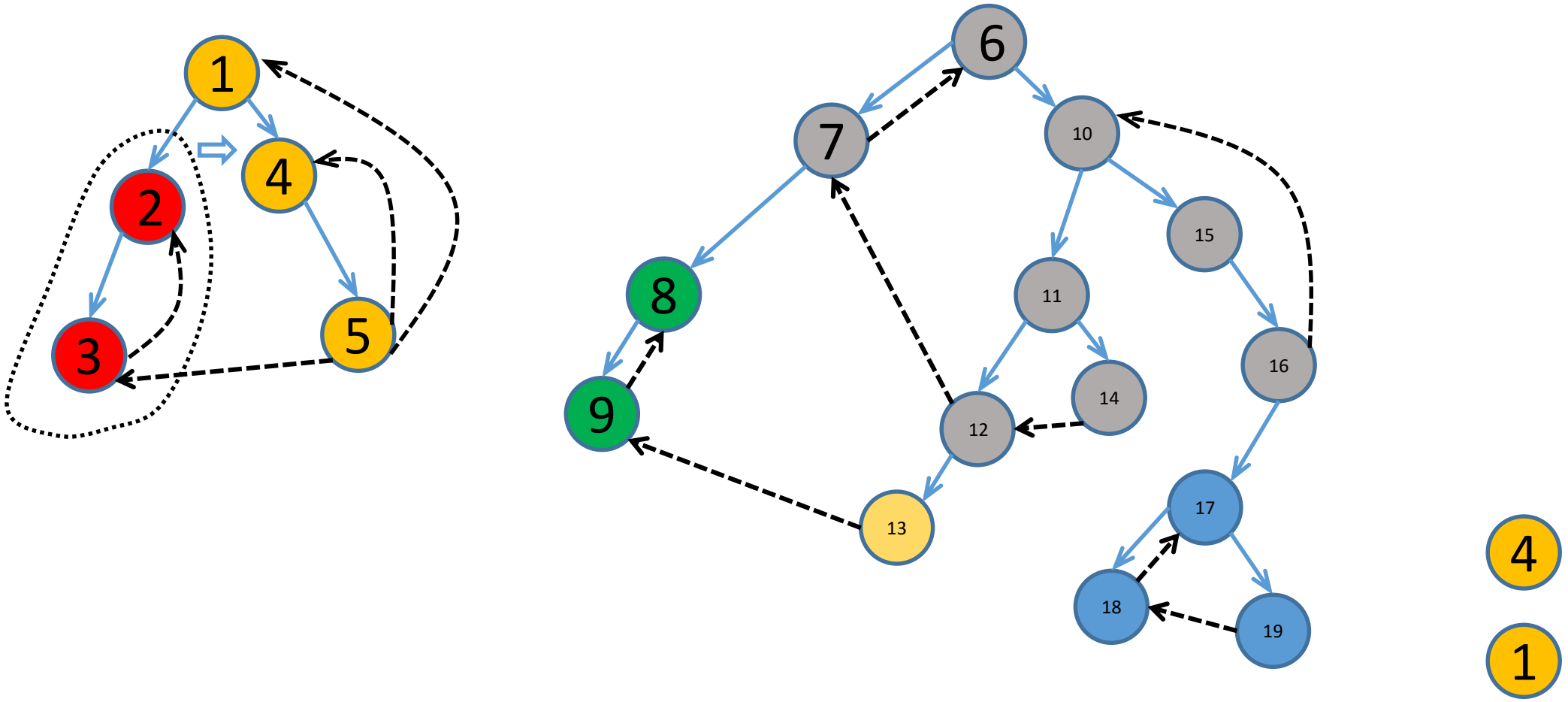
Use an additional stack



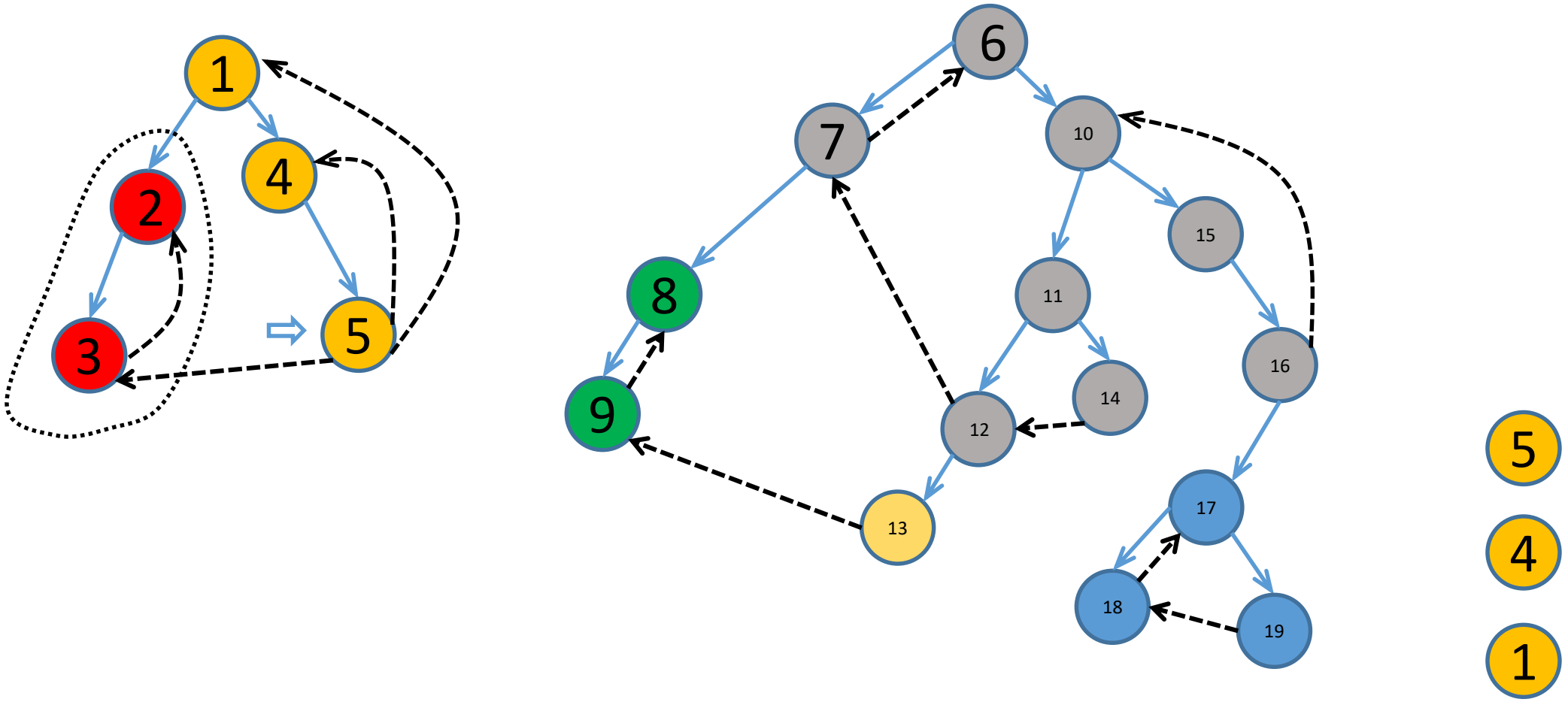
Use an additional stack



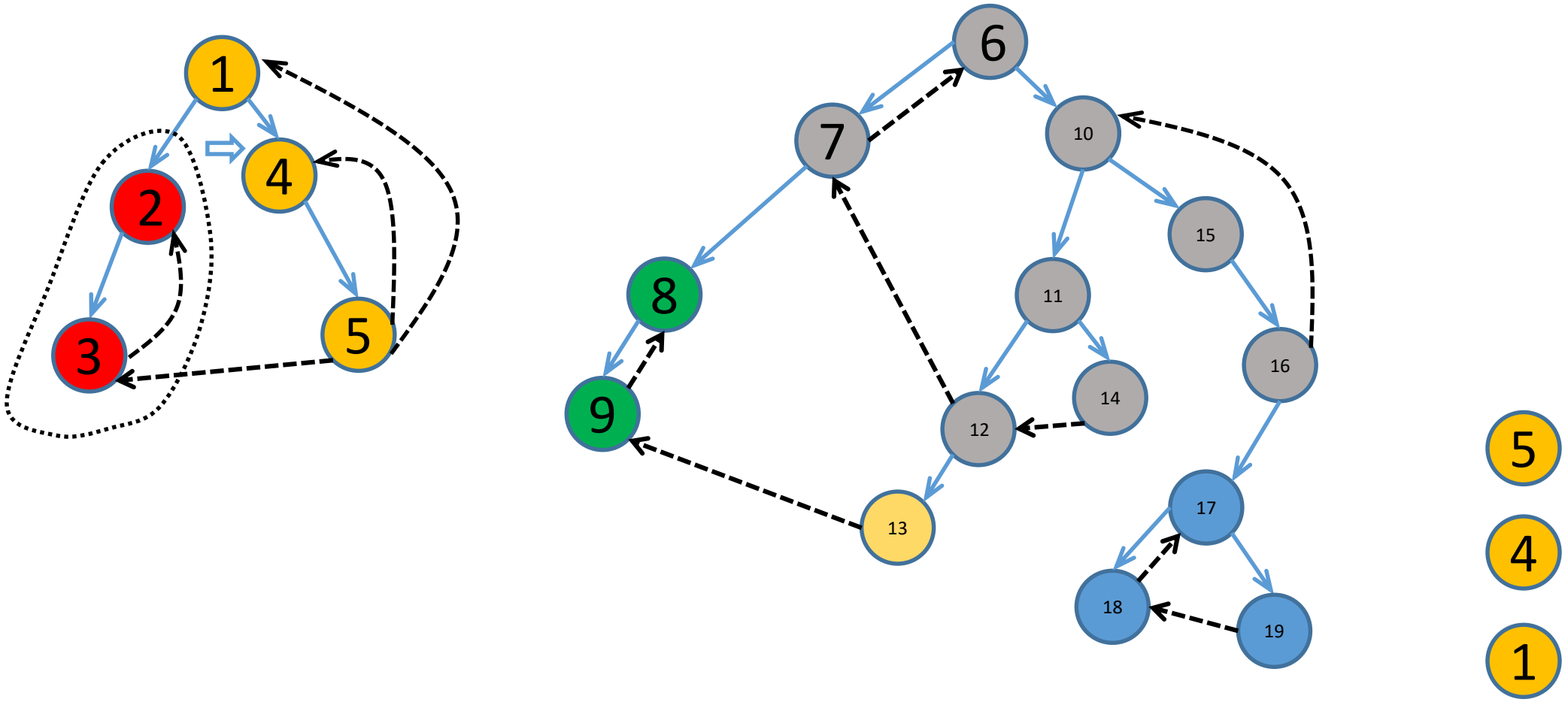
Use an additional stack



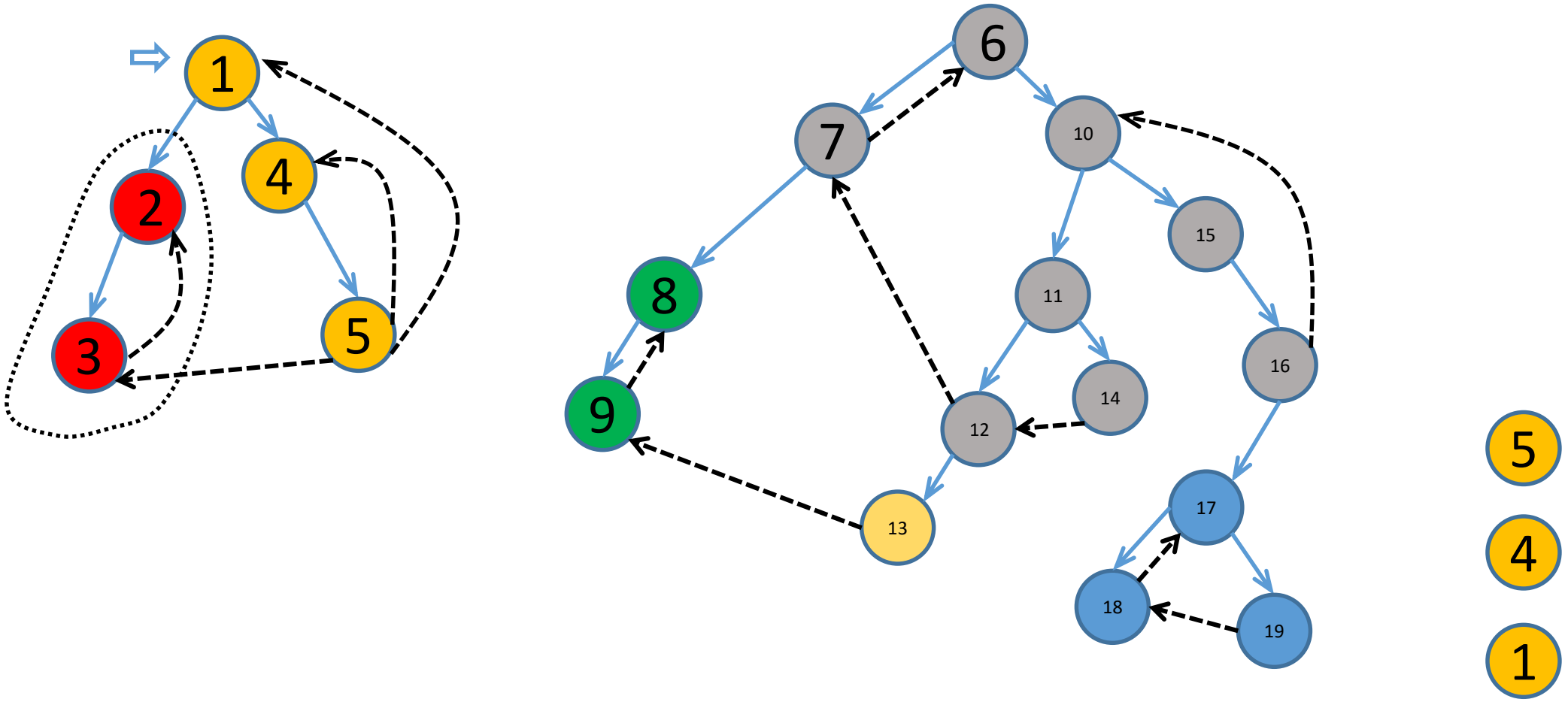
Use an additional stack



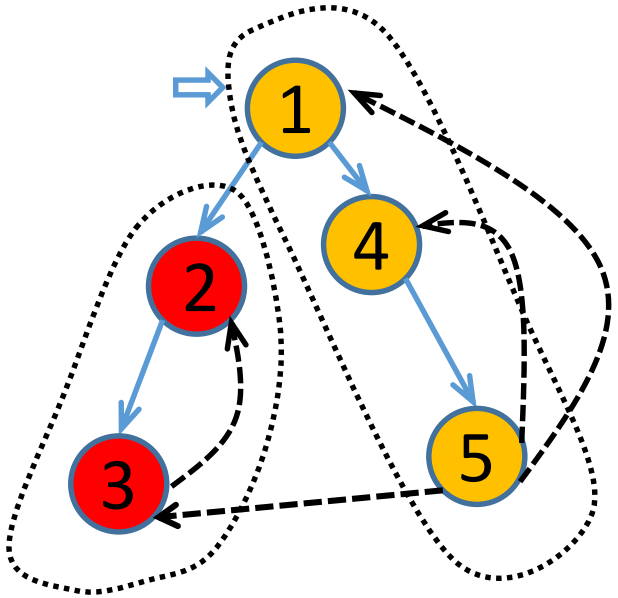
Use an additional stack

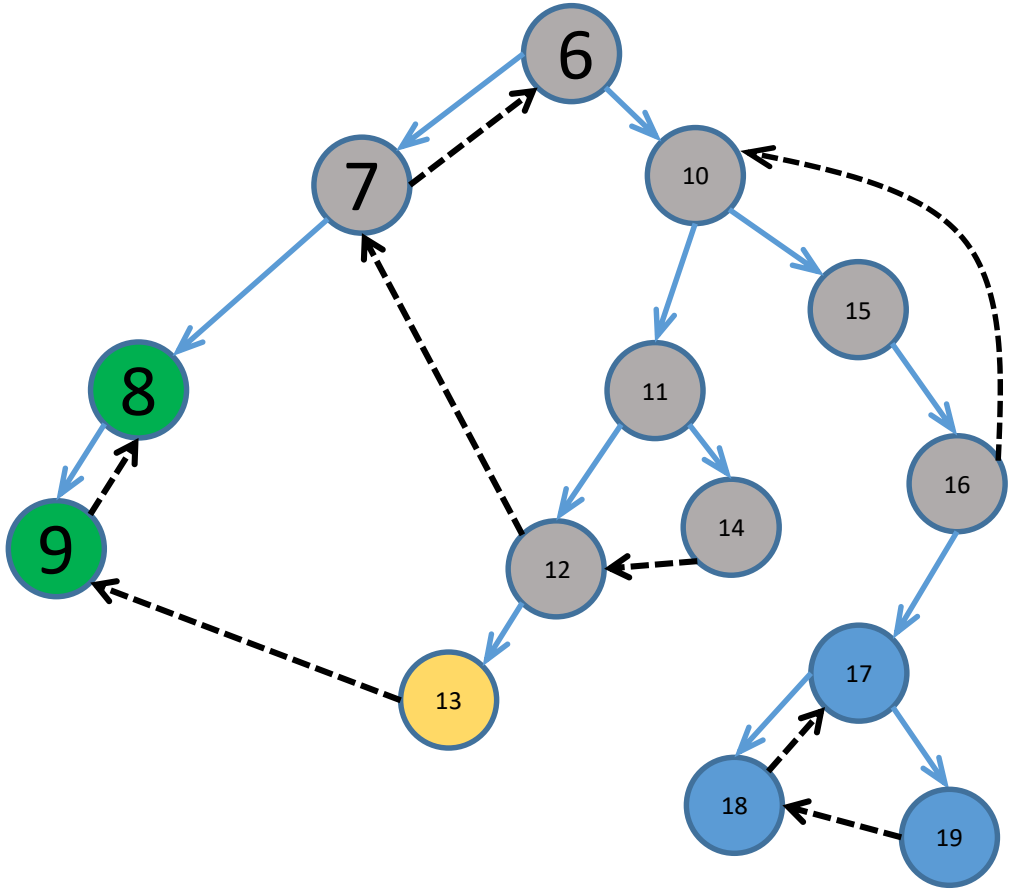


Use an additional stack

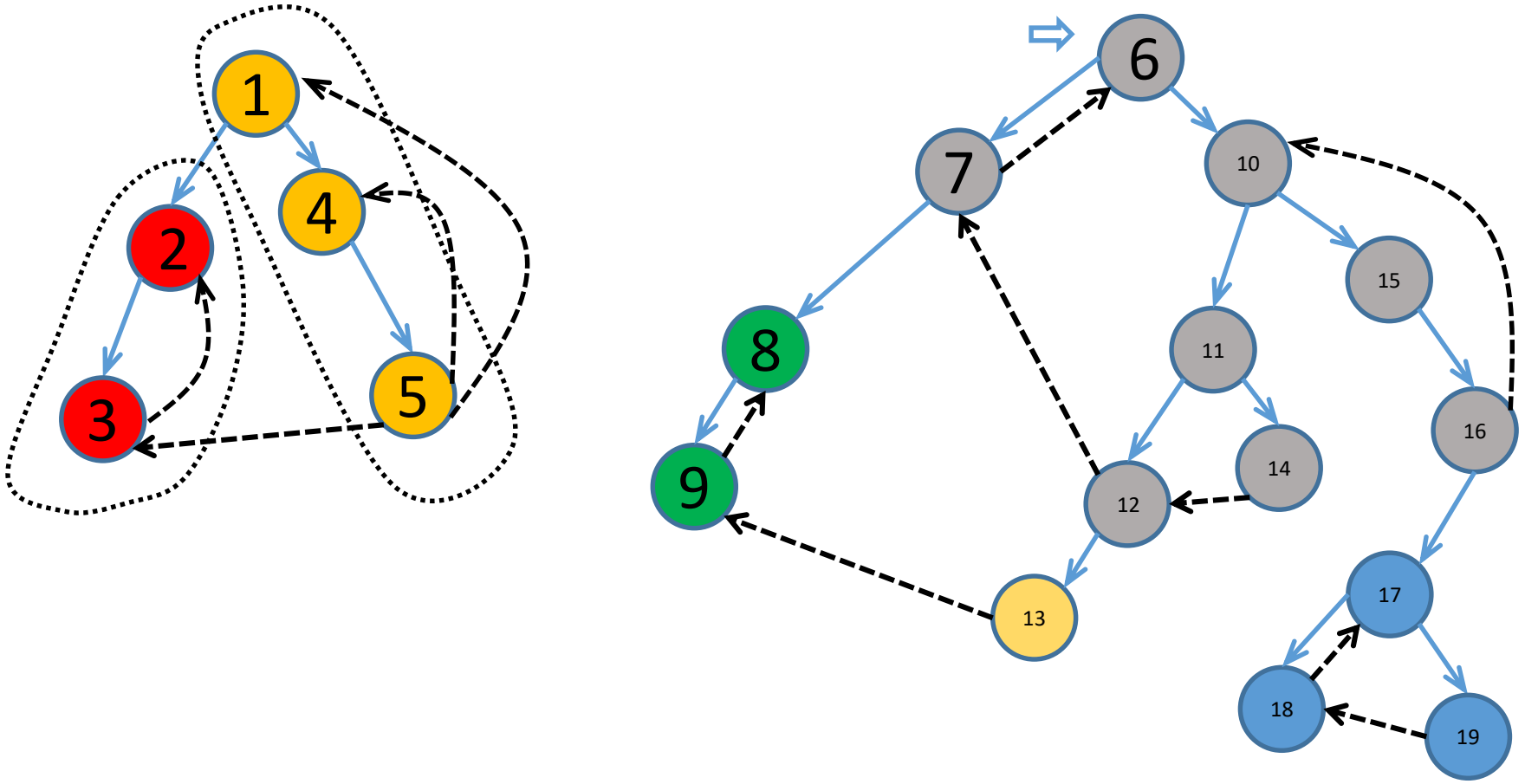


Use an additional stack

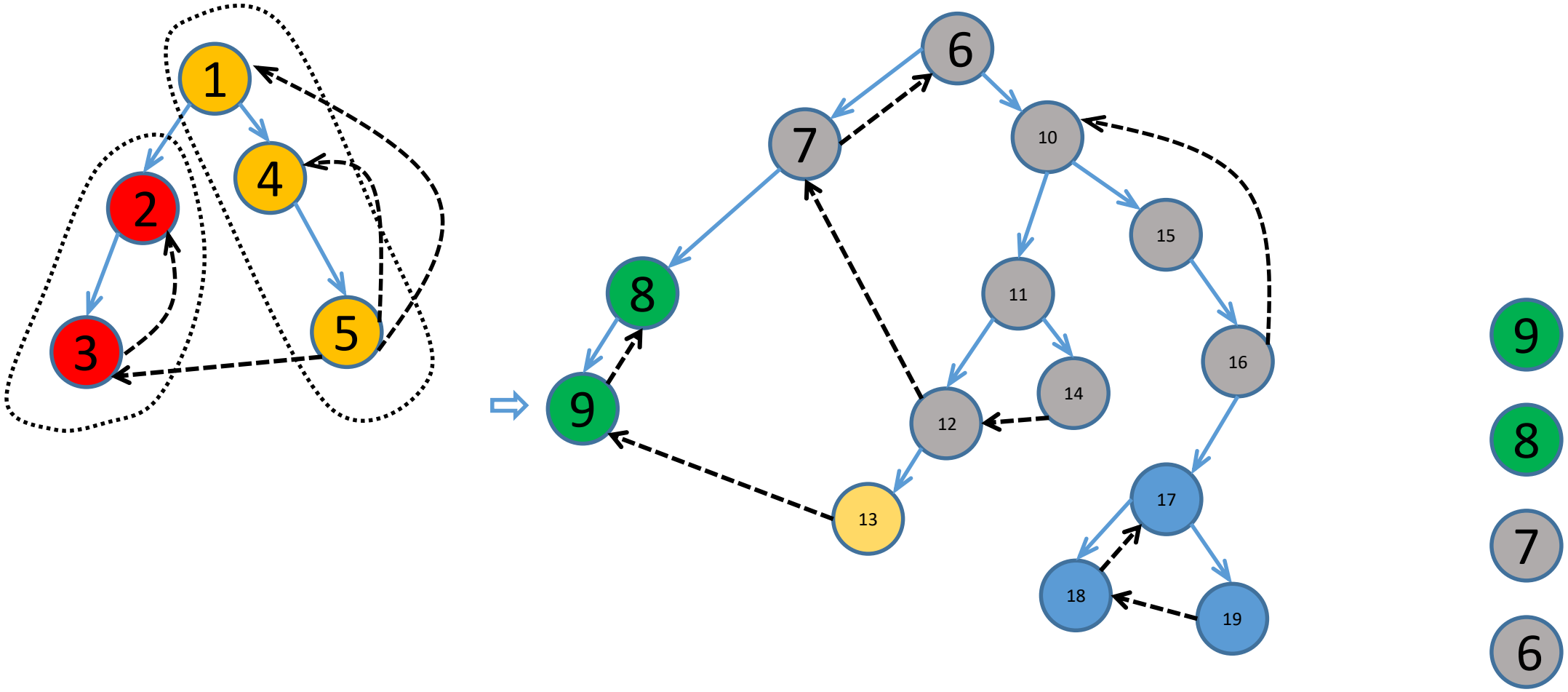




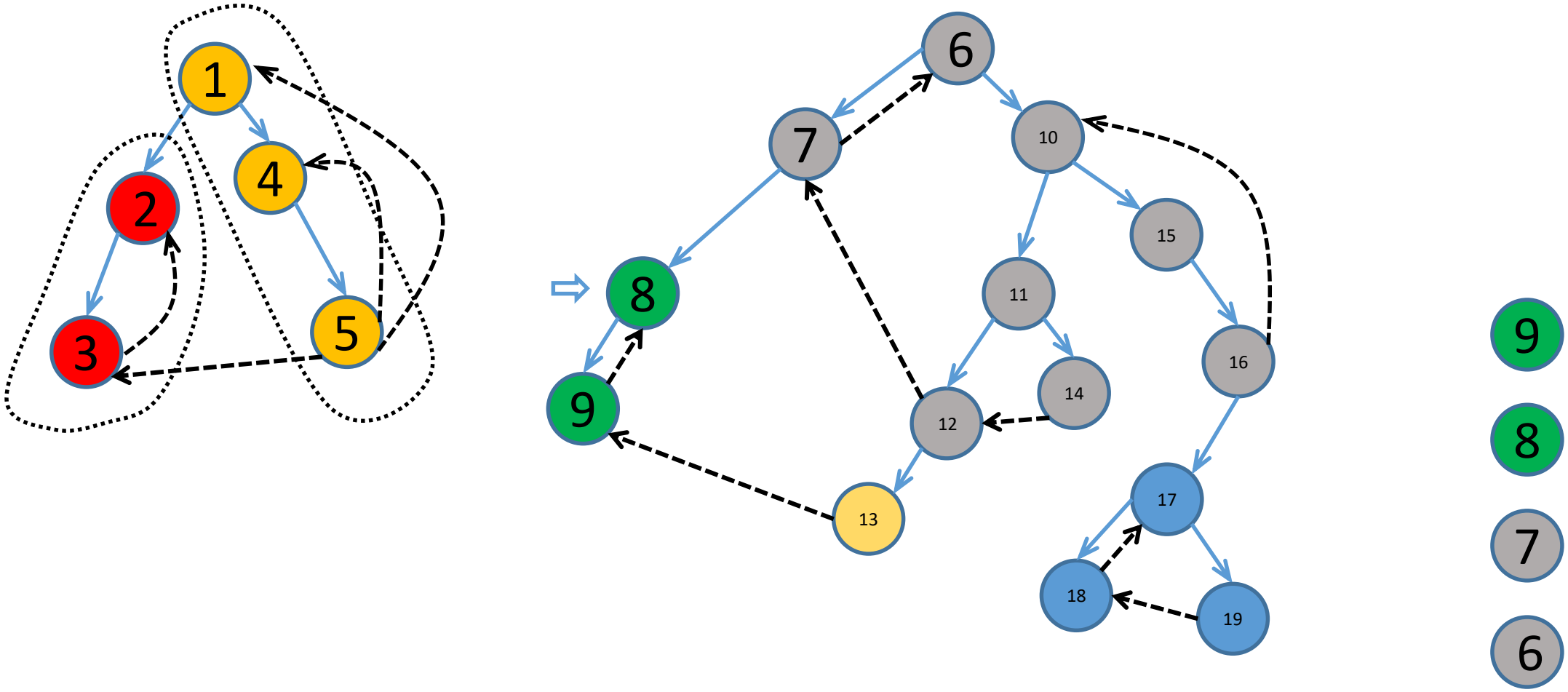
Use an additional stack



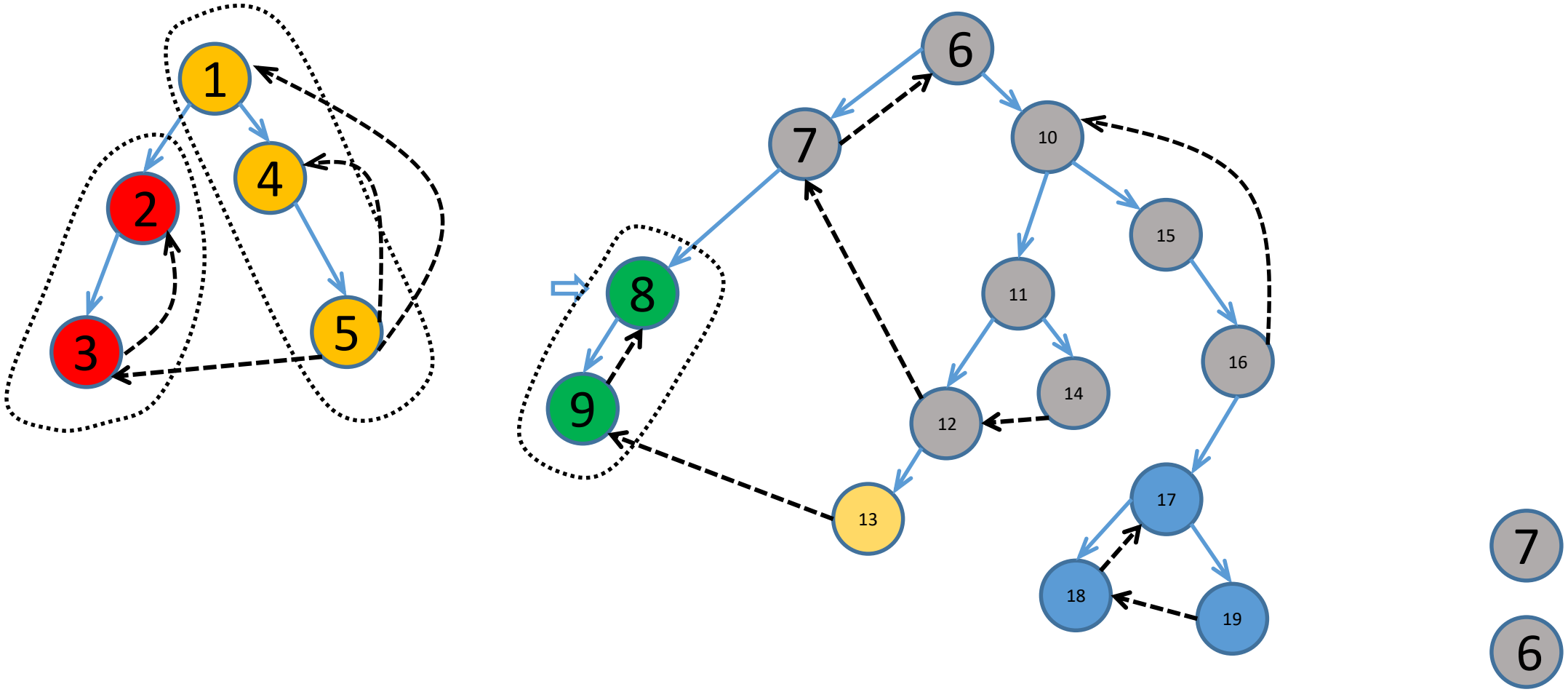
Use an additional stack



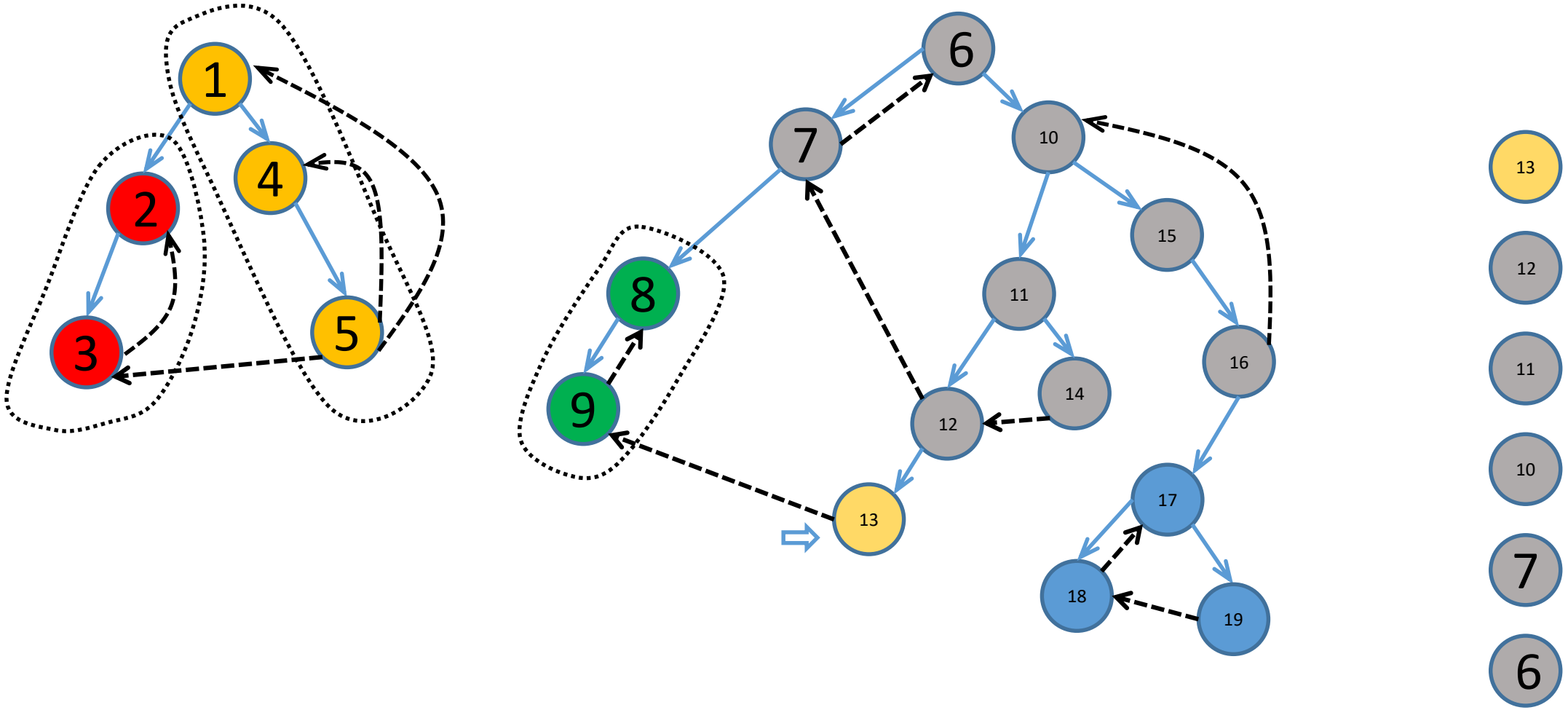
Use an additional stack



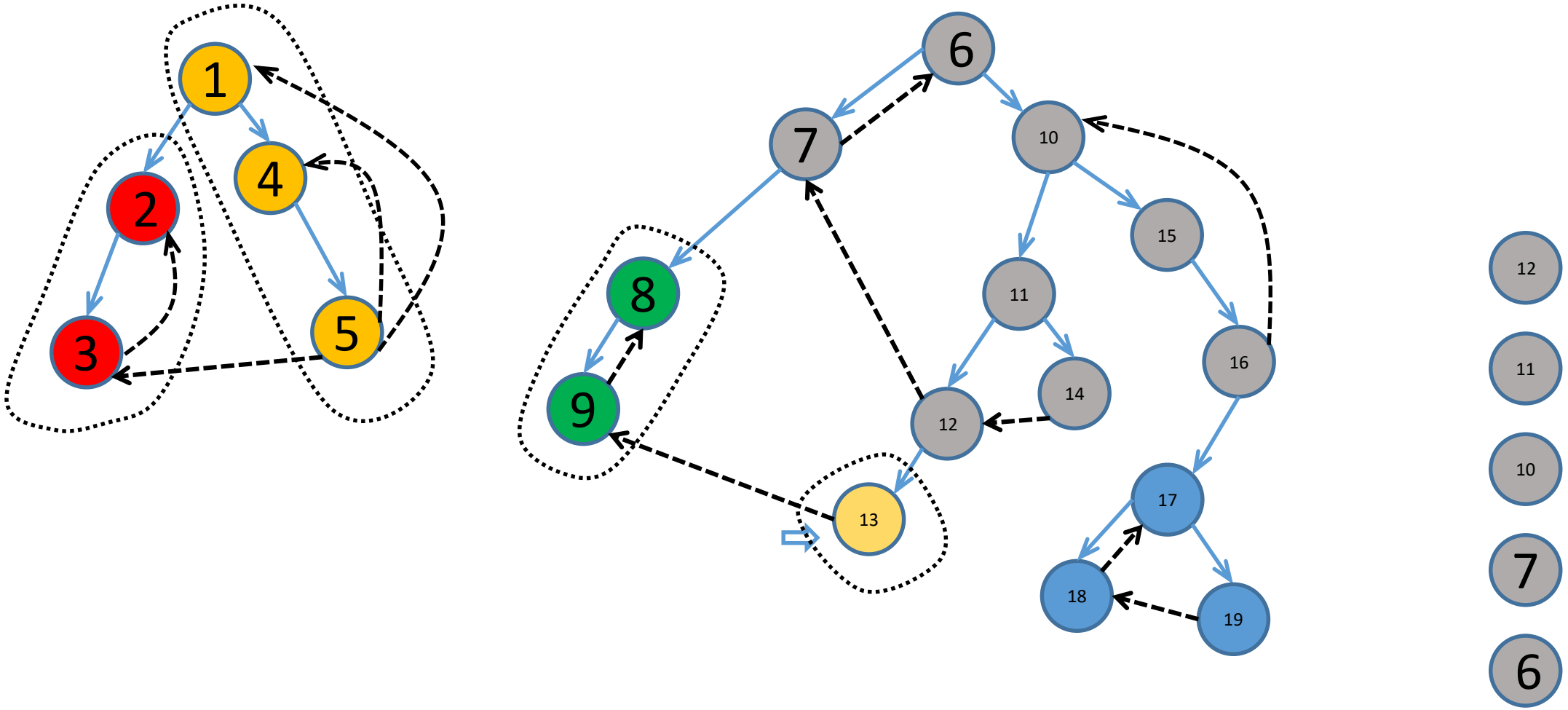
Use an additional stack



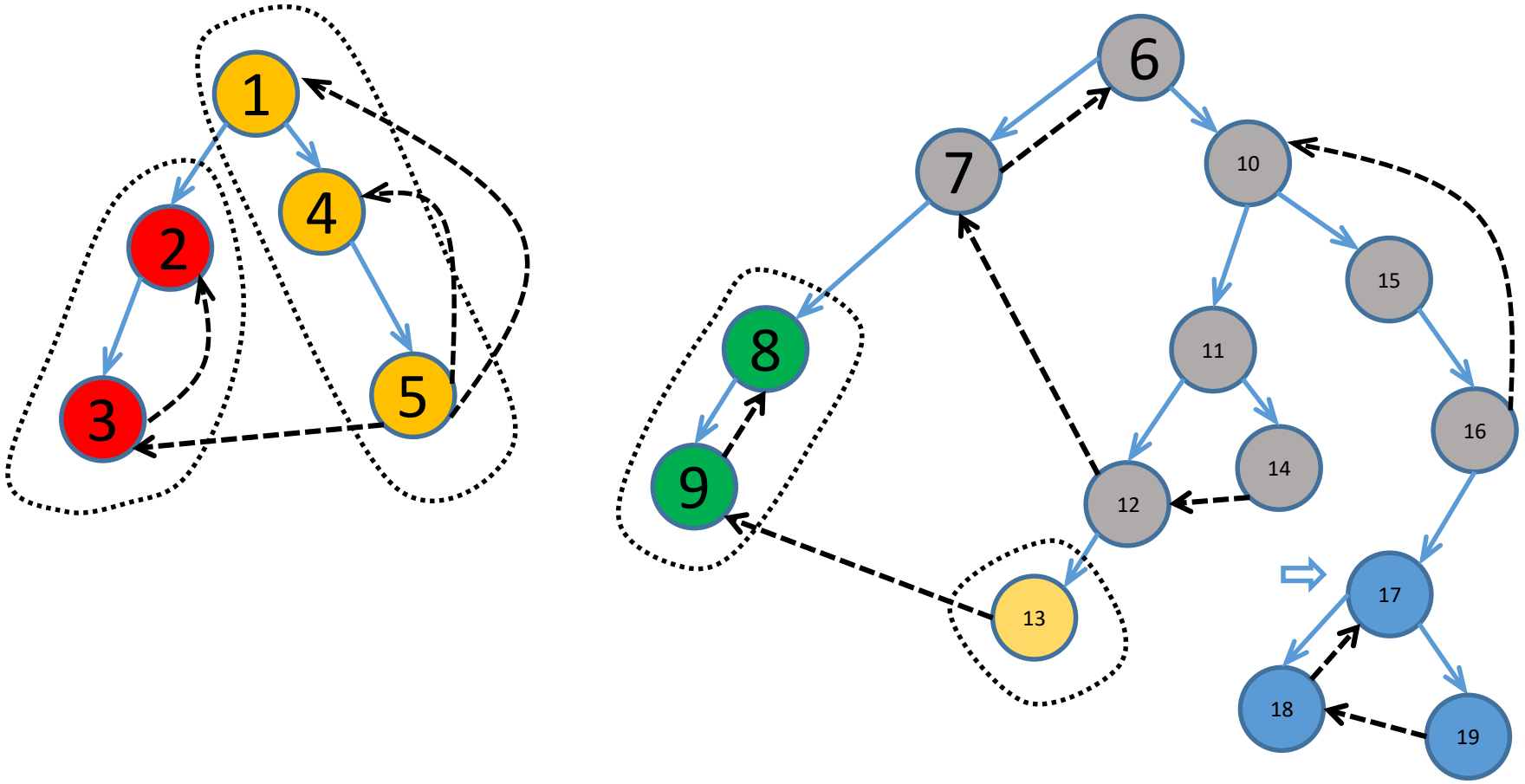
Use an additional stack



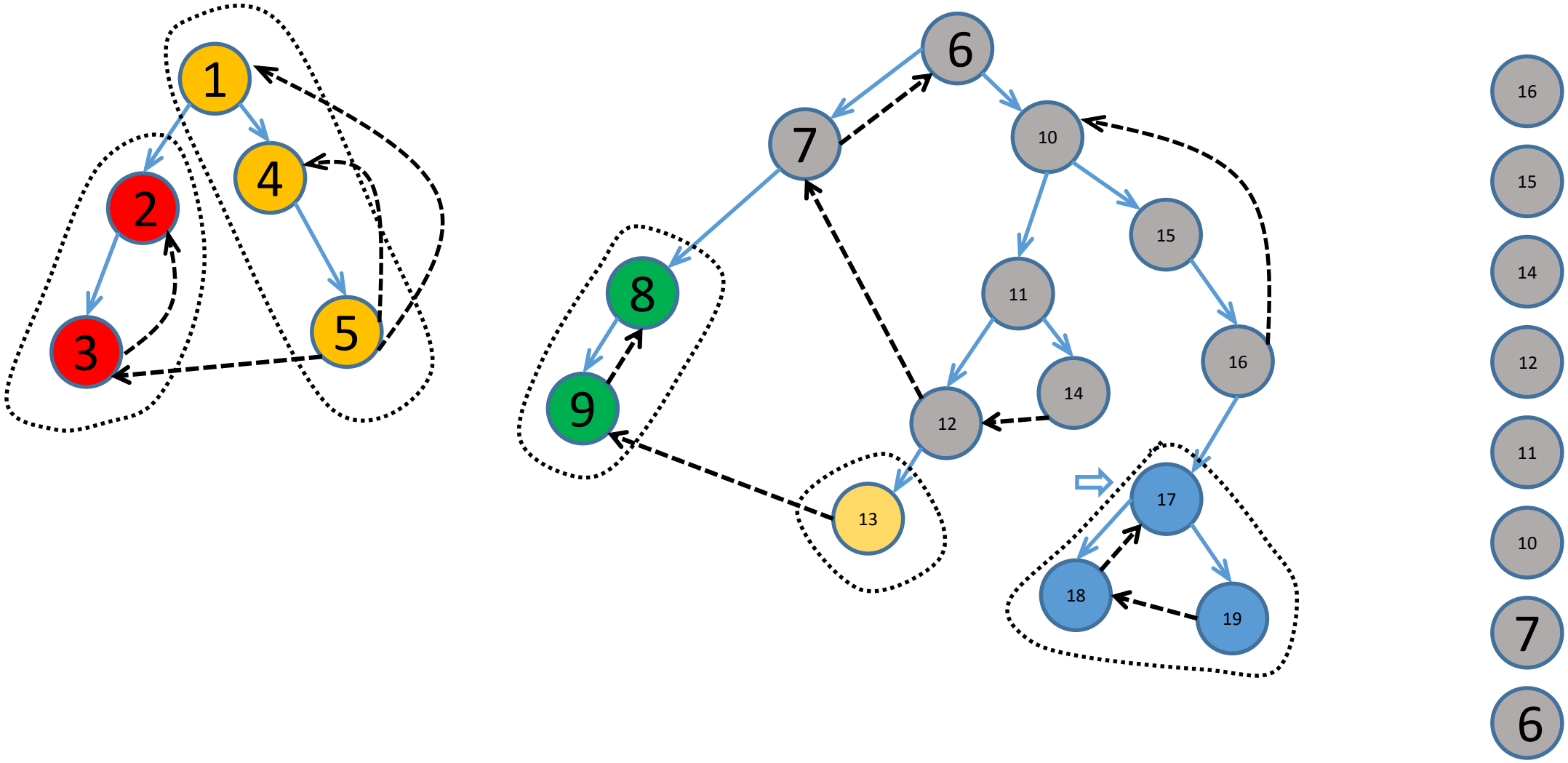
Use an additional stack



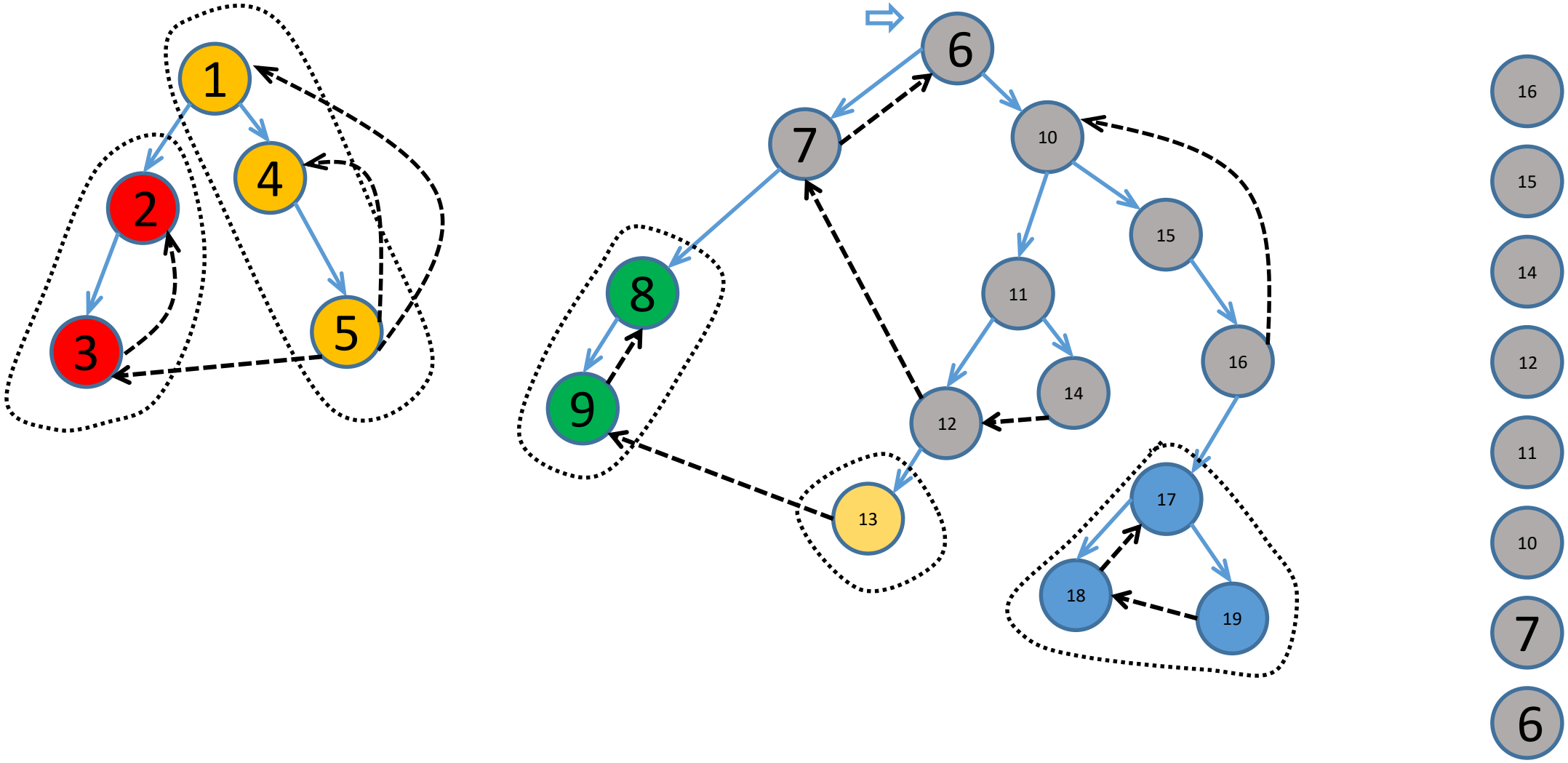
Use an additional stack



Use an additional stack



Use an additional stack



Use an additional stack