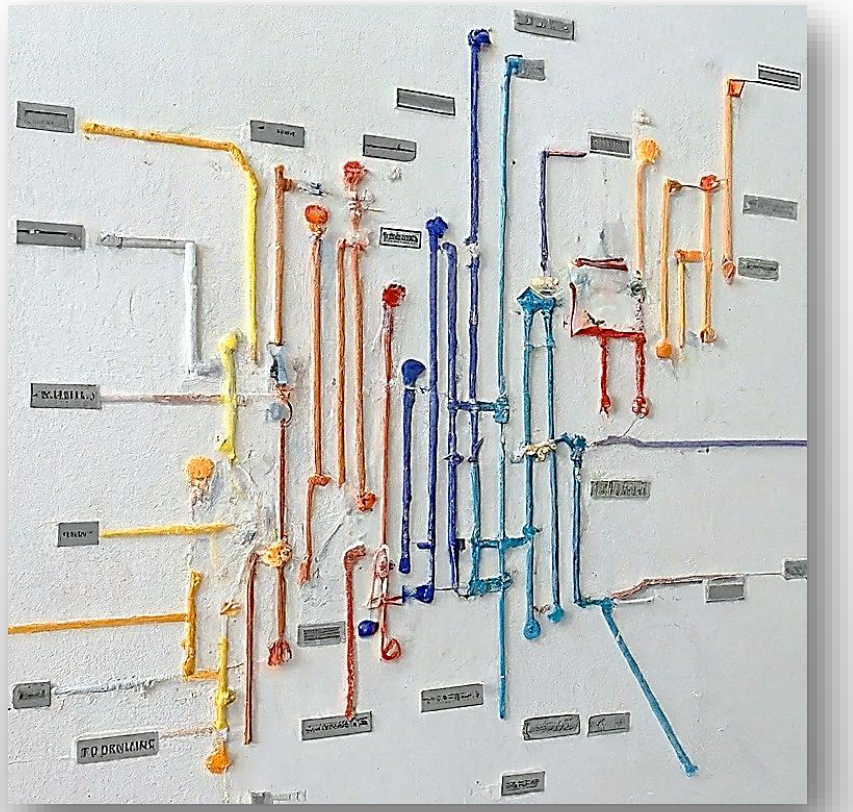
Bölüm 4: Çizge Algoritmaları

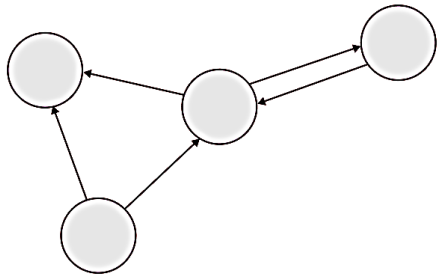
Algoritmalar

Çizge Algoritmaları

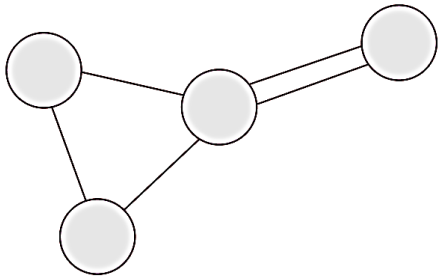
- Dünya aslında bir ağ gibidir.
  - Şehirler yollarla,
  - İnsanlar ilişkilerle,
  - Bilgisayarlar kablolarla birbirine bağlıdır.
- Çizge algoritmaları bu ağları inceler ve anlamlandırır.



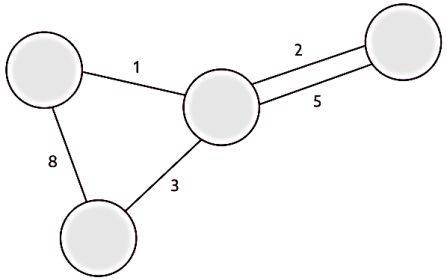
Çizge Türleri



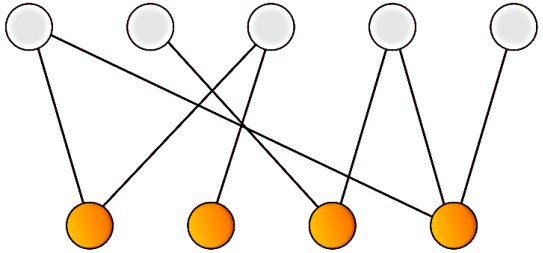
Directed graph



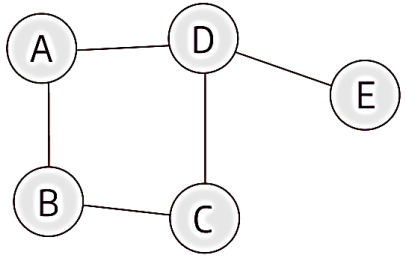
Undirected



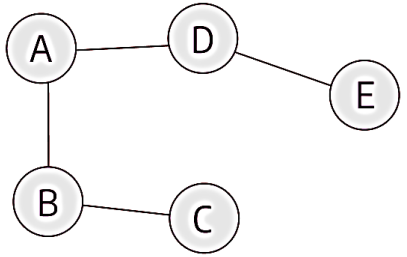
Weighted



Bipartite graph



Cyclic graph



Acyclic graph

Çizge Algoritmaları

- Birbirine bağlı noktalar (düğüm) ve bu noktaları birleştiren çizgiler (kenar) ile temsil edilen ağ yapılarını inceler.
- Ağlarda en kısa yolu hesaplama, gruplama gibi işlemleri gerçekleştirir.
- Sosyal ağlar, harita uygulamaları, navigasyon gibi birçok alanda kullanılır.

Çizge Algoritmalarının Çeşitleri

- Farklı çizge algoritmaları, farklı işlemler için kullanılır.
- Derinlik Öncelikli Arama (DFS):
  - Bir düğümden başlar, dallanarak tüm ağı gezer.
- Genişlik Öncelikli Arama (BFS):
  - Bir düğümden başlar, katman katman tüm ağı gezer.
- Dijkstra Algoritması:
  - Başlangıç düğümünden diğer düğümlere en kısa yolları bulur.
- Kruskal Algoritması:
  - Bir ağı minimum maliyetle birbirine bağlayan kenarları seçer.

Çizge Algoritmaları

- DFS bir labirentten çıkış yolu ararken kullanılabilir.
- BFS bir haberin tüm şehire yayılma sürecini modelleyebilir.
- Dijkstra en kısa sürede teslimat yapmak için kullanılabilir.

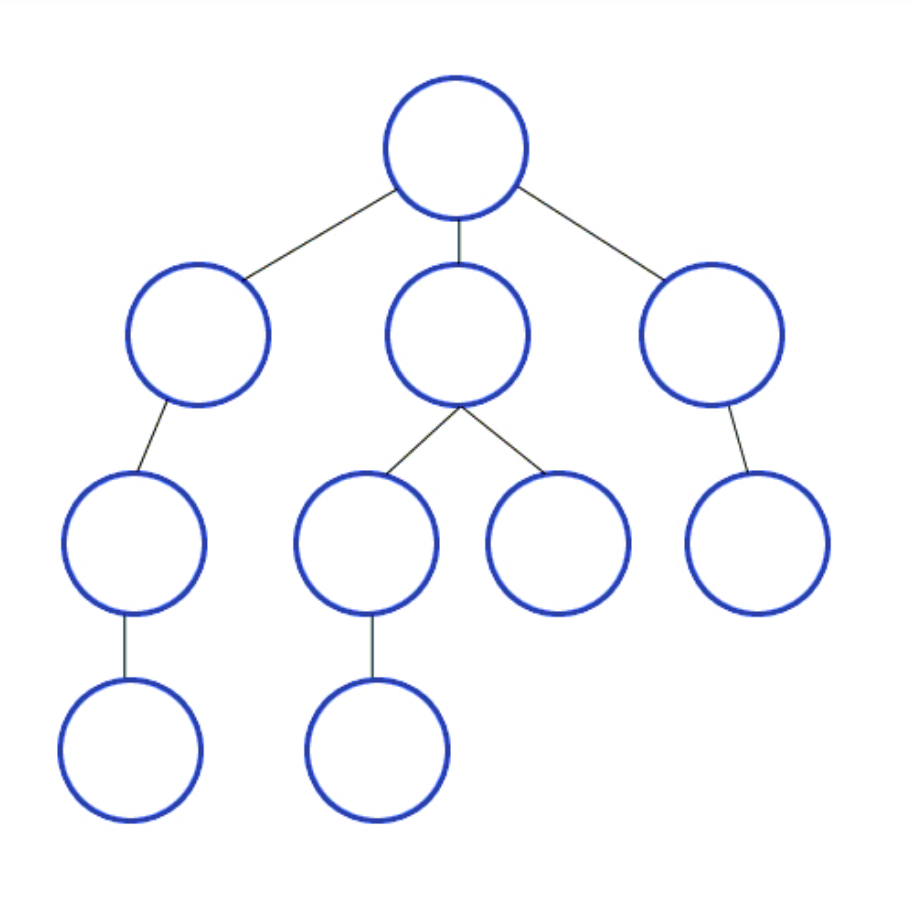
Çizge Algoritmaları

- Çizge gezinme algoritmaları (*Graph traversal*)
- En kısa yol algoritmaları (*Shortest path*)
- Minimum yayılan ağaç algoritmaları (*Minimum spanning tree*)
- Ağ akış algoritmaları (*Network flow*)

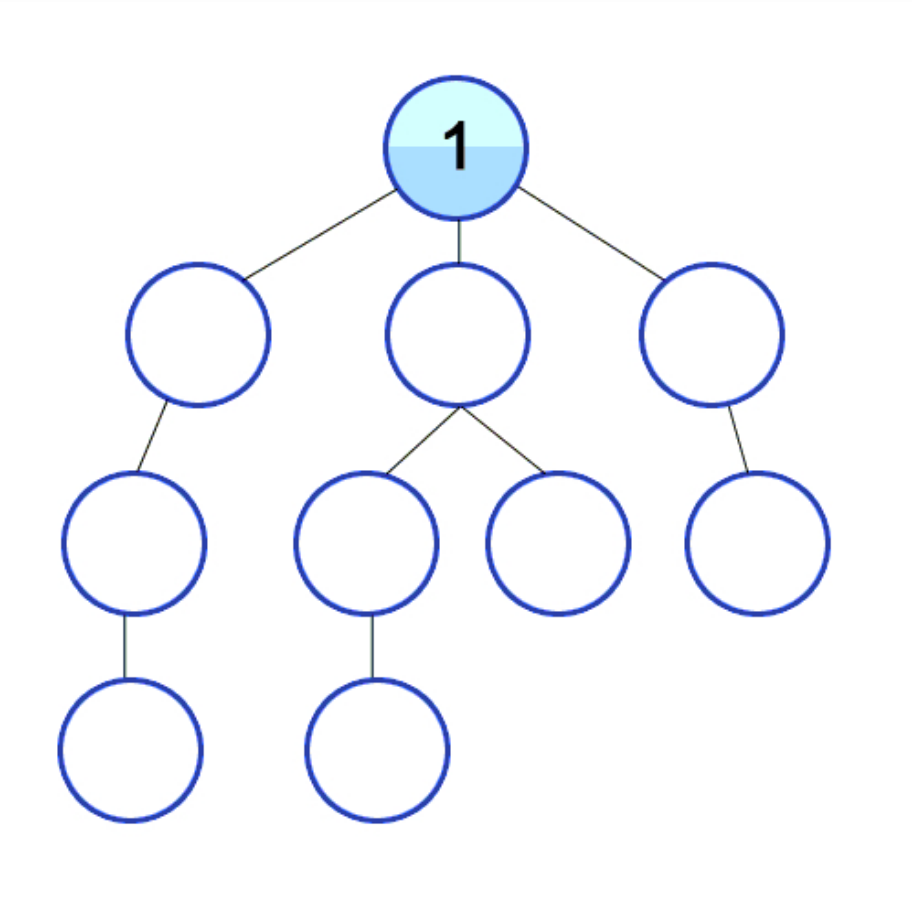
Çizge Gezinme Algoritmaları (Graph Traversal)

- Çizgenin yapısını sistematik bir şekilde keşfetmek için kullanılır.
- İki ana kategoriye ayrılır:
  - derinlik öncelikli arama (DFS) ve
  - genişlik öncelikli arama (BFS).
- *DFS*, yığıt veri yapısı kullanarak, bir düğümden başlar ve mümkün olduğunca derinlere iner, tüm komşularını ziyaret ettikten sonra geri döner.
- *BFS*, kuyruk veri yapısı kullanarak, seviye seviye tüm düğümleri ziyaret eder.

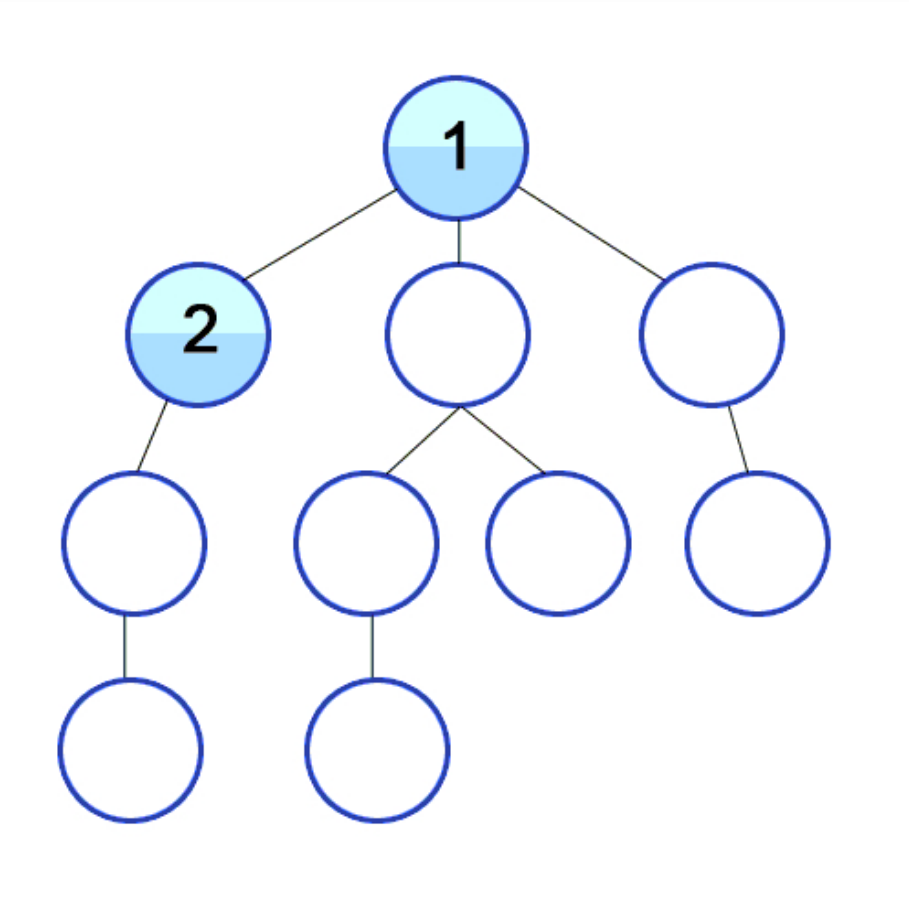
Derinlik Öncelikli Arama



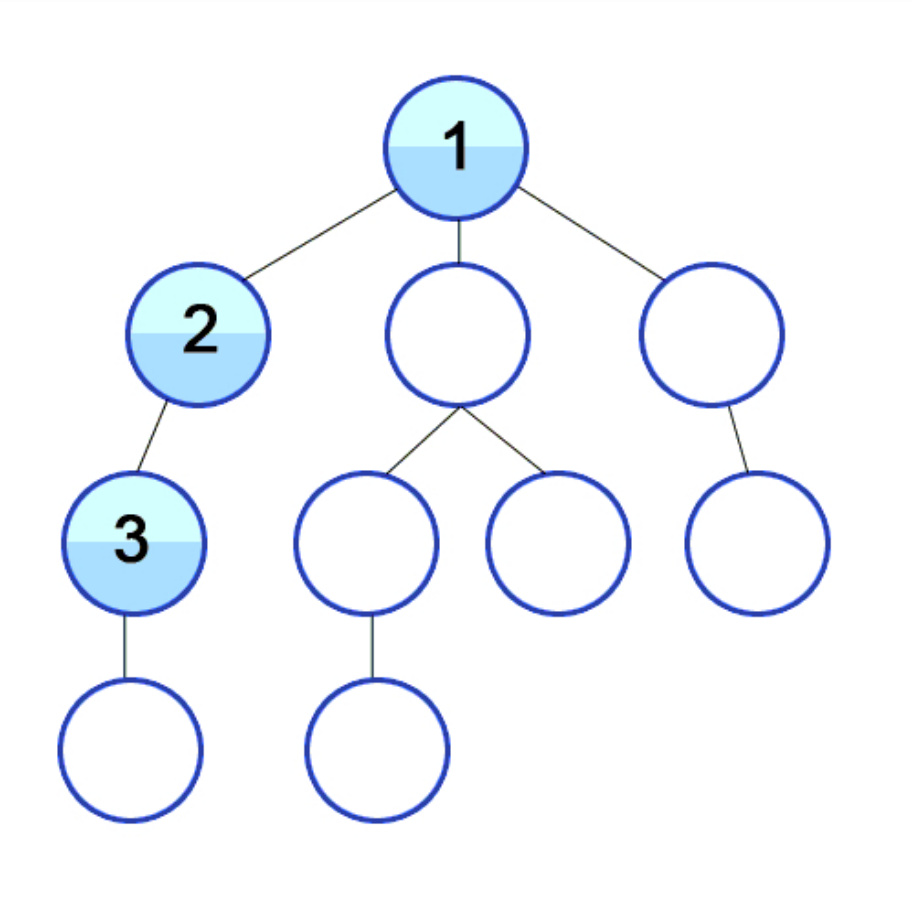
Derinlik Öncelikli Arama



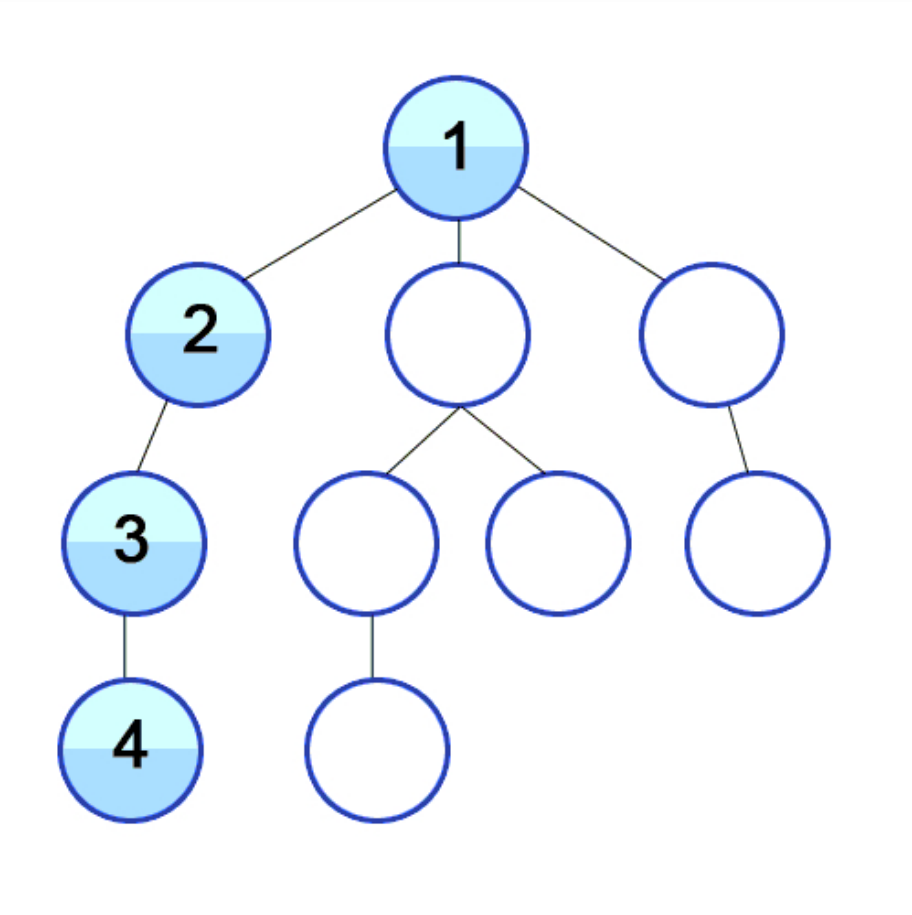
Derinlik Öncelikli Arama



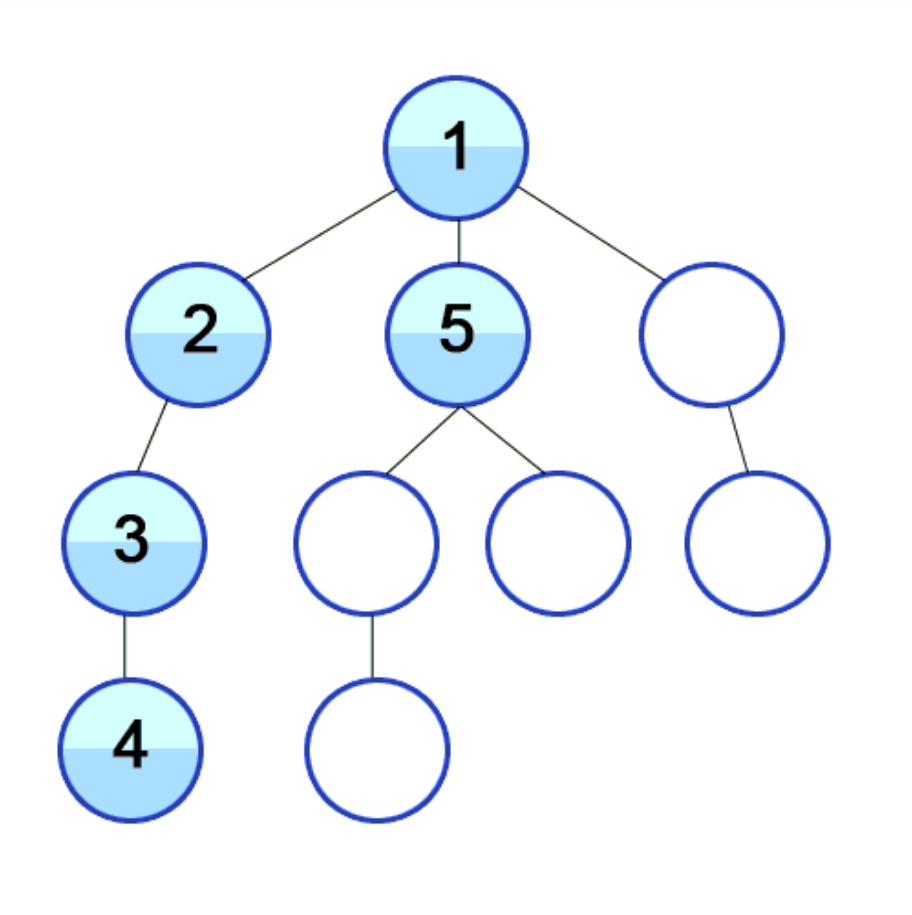
Derinlik Öncelikli Arama



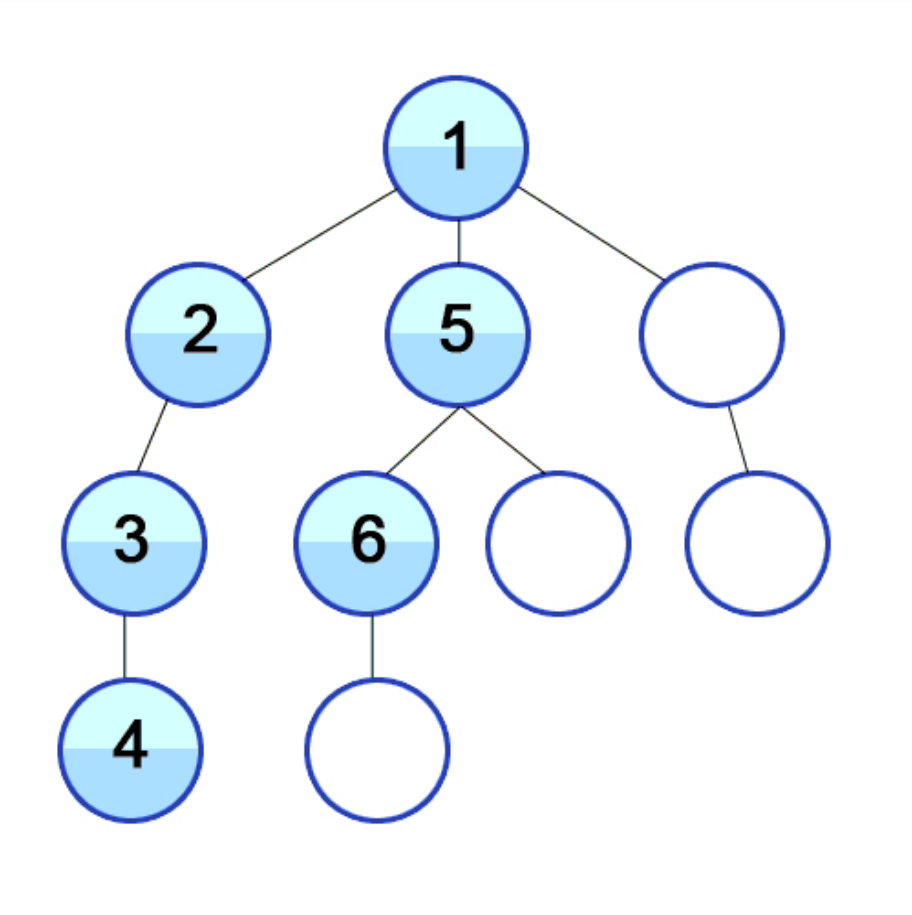
Derinlik Öncelikli Arama



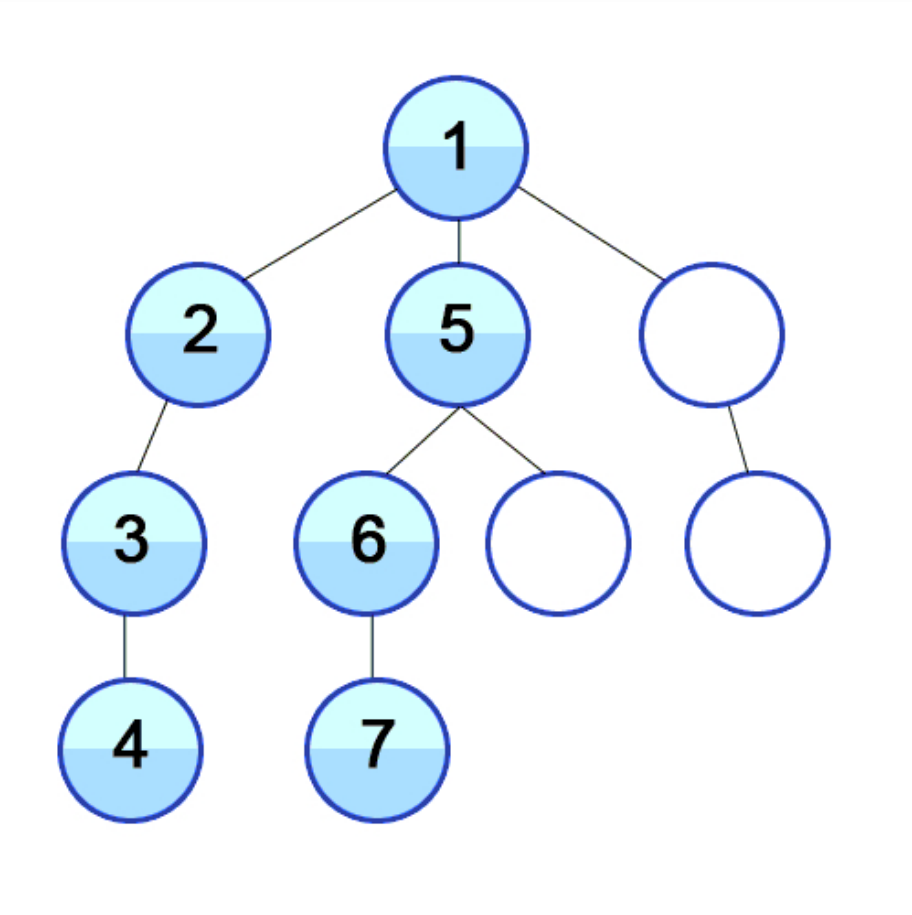
Derinlik Öncelikli Arama



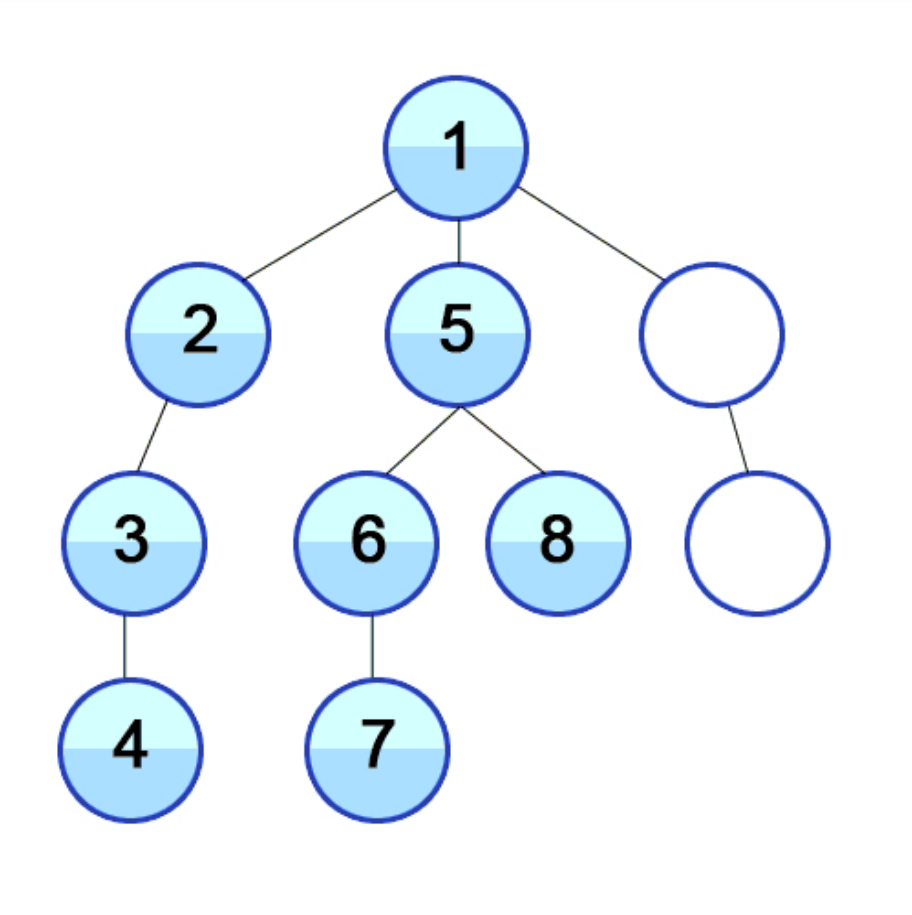
Derinlik Öncelikli Arama



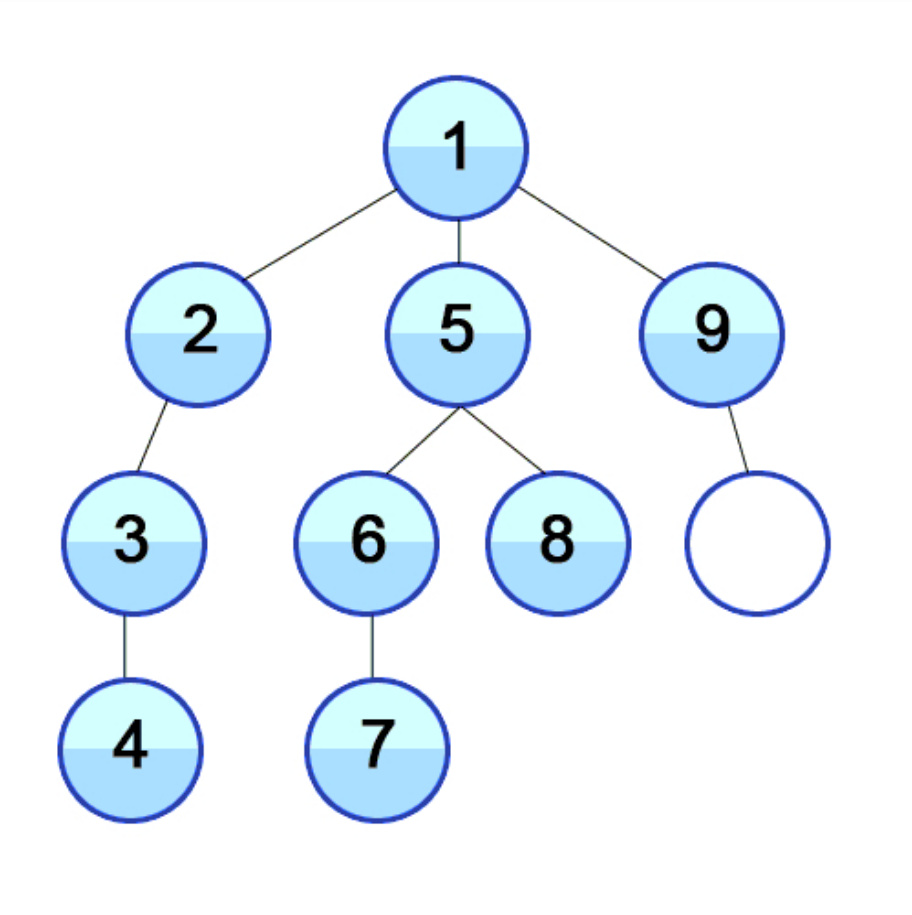
Derinlik Öncelikli Arama



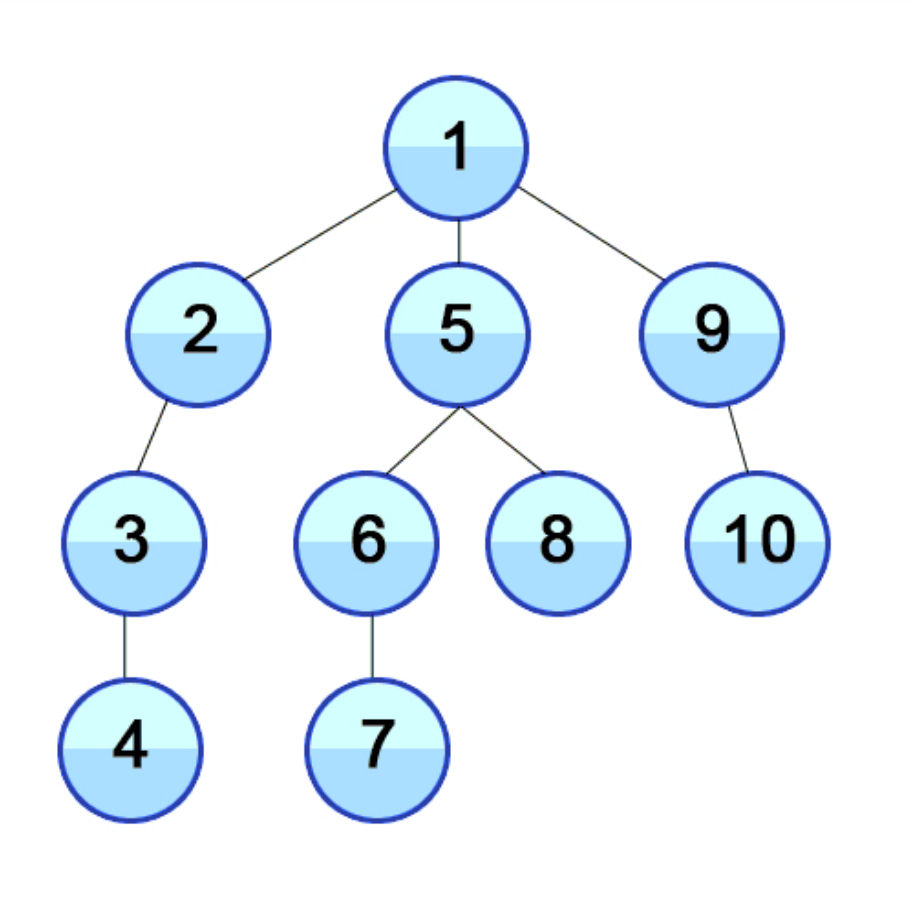
Derinlik Öncelikli Arama



Derinlik Öncelikli Arama



Derinlik Öncelikli Arama



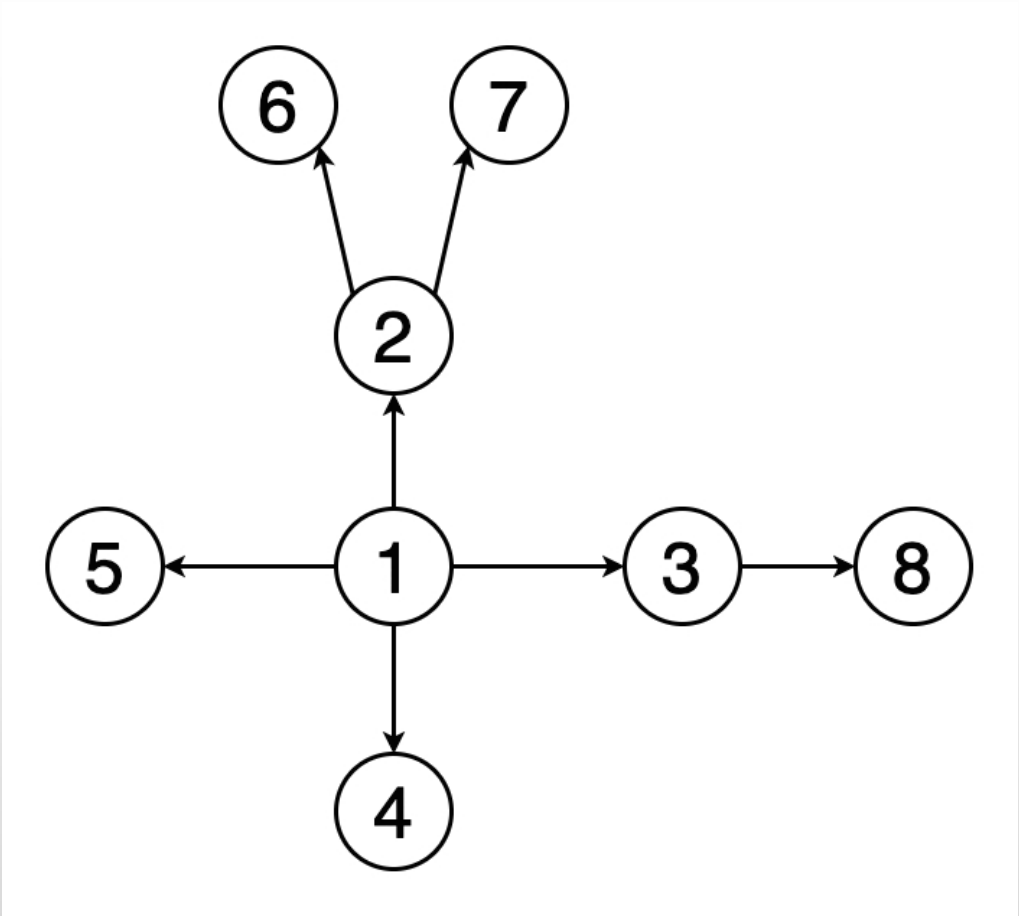
Recursive DFS

```
procedure DFS(G, v) is
    label v as discovered
    for all directed edges from v to w that are in G.adjacentEdges(v) do
        if vertex w is not labeled as discovered then
            recursively call DFS(G, w)
```

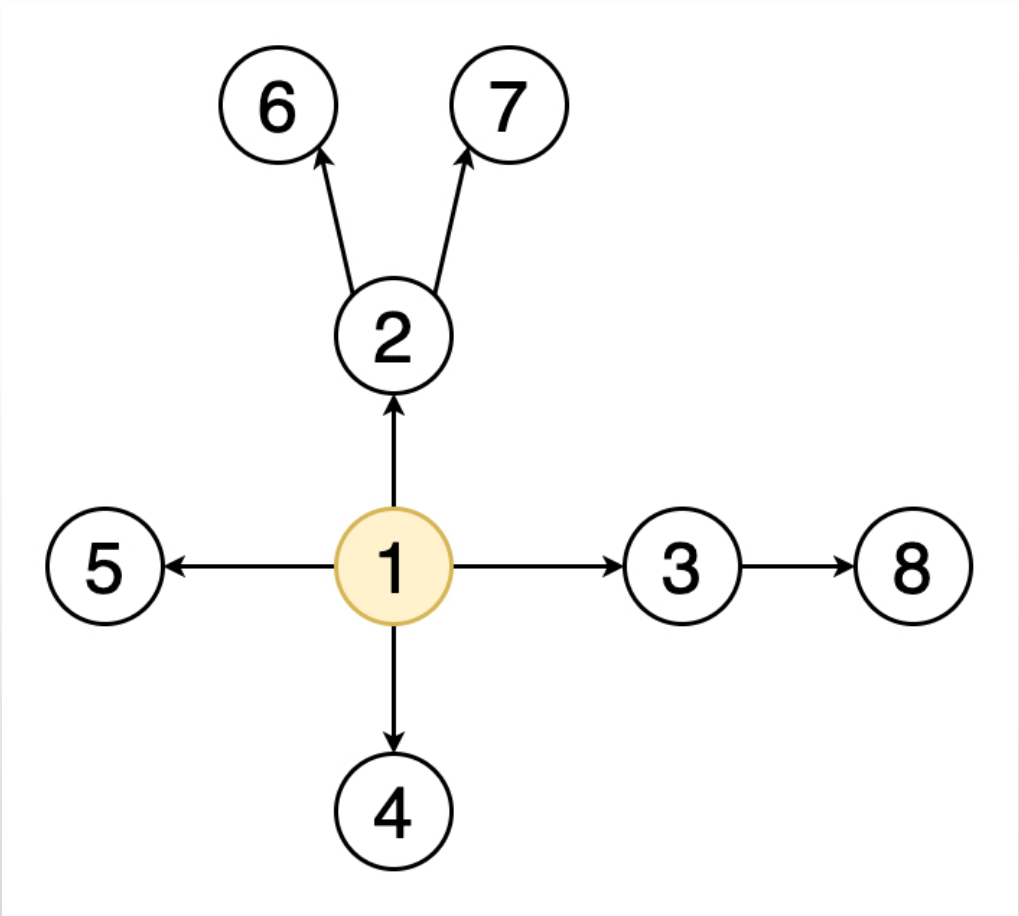
Iterative DFS

```
procedure DFS_iterative(G, v) is
    let S be a stack
    S.push(v)
    while S is not empty do
        v = S.pop()
        if v is not labeled as discovered then
            label v as discovered
            for all edges from v to w in G.adjacentEdges(v) do
                S.push(w)
```

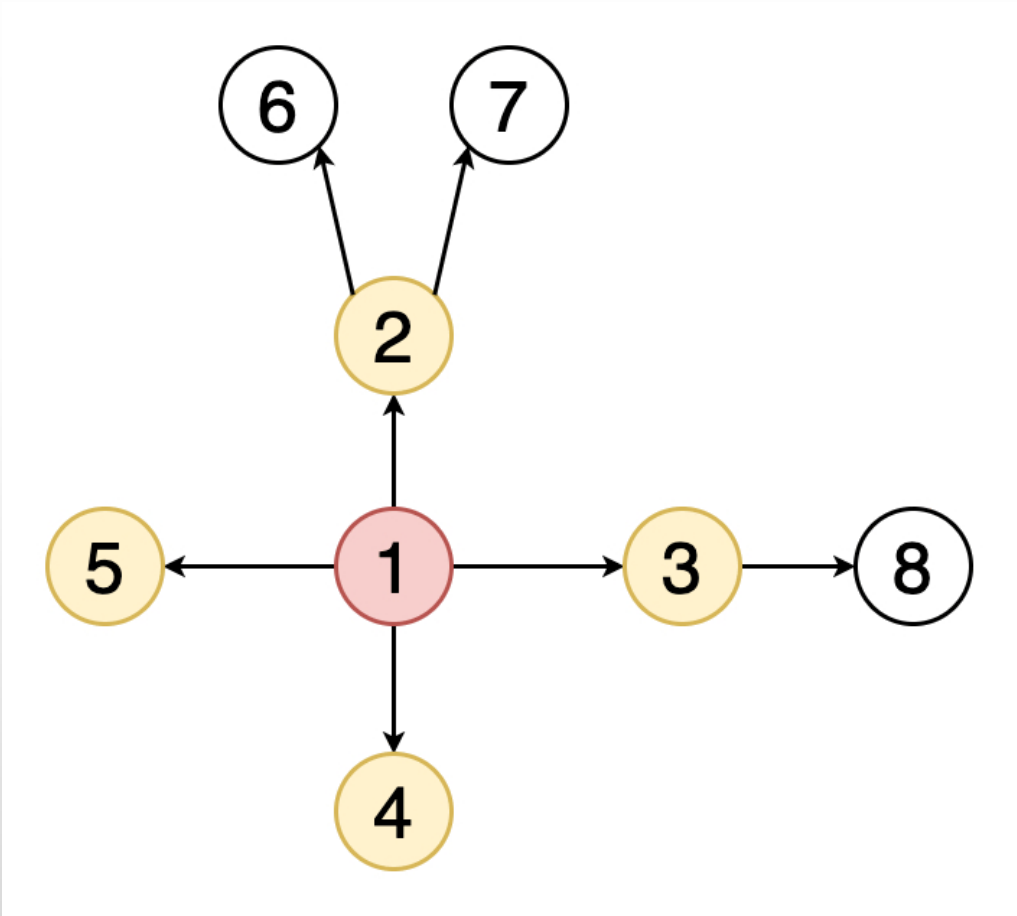
Genişlik Öncelikli Arama



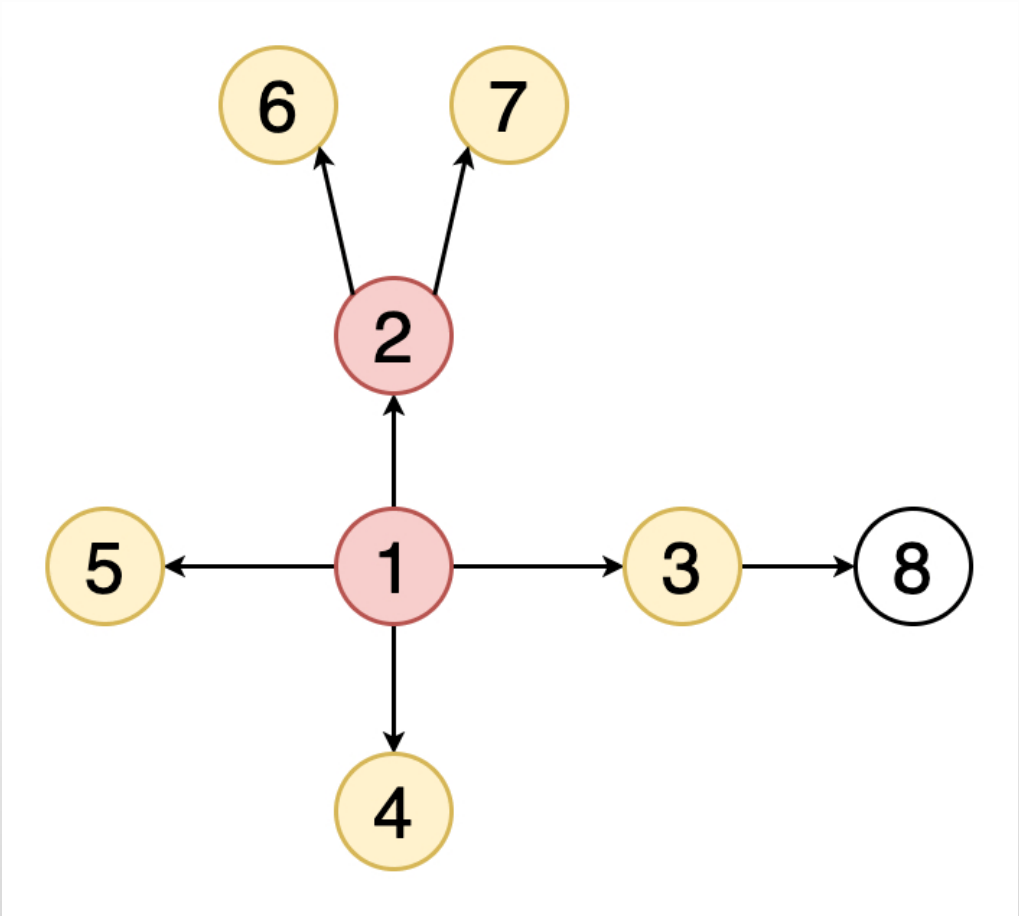
Genişlik Öncelikli Arama



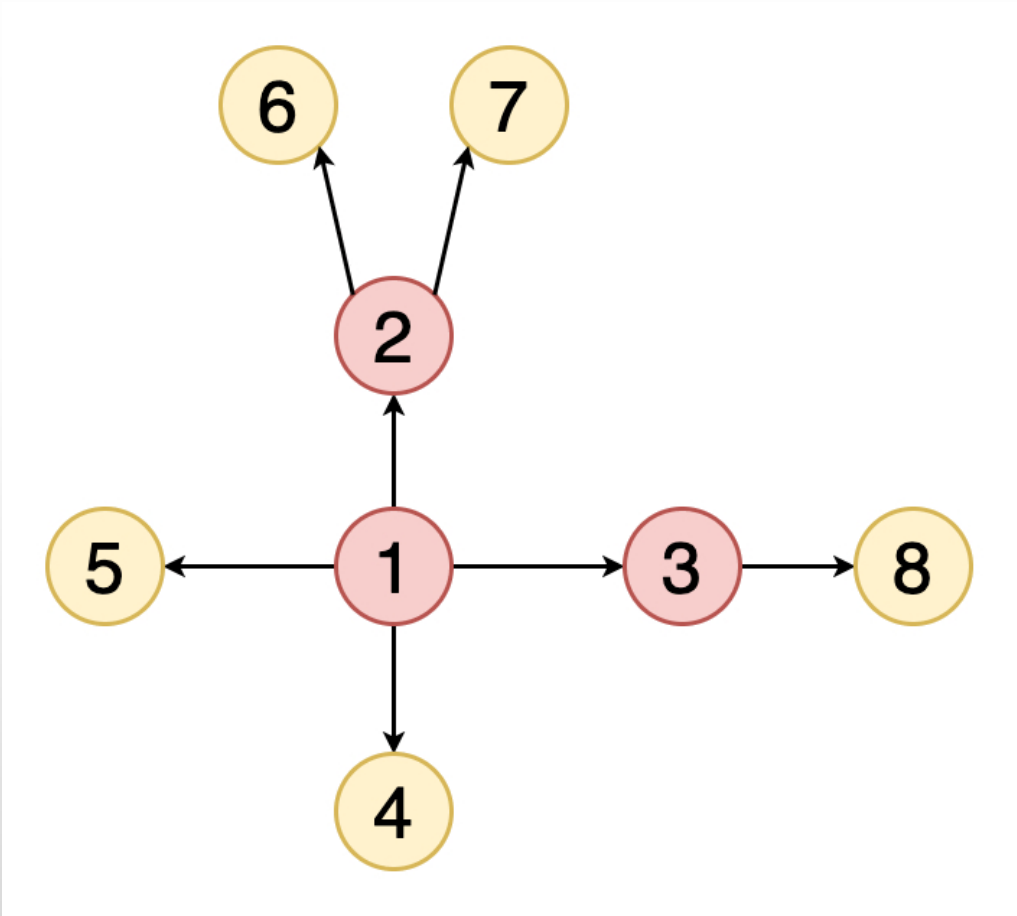
Genişlik Öncelikli Arama



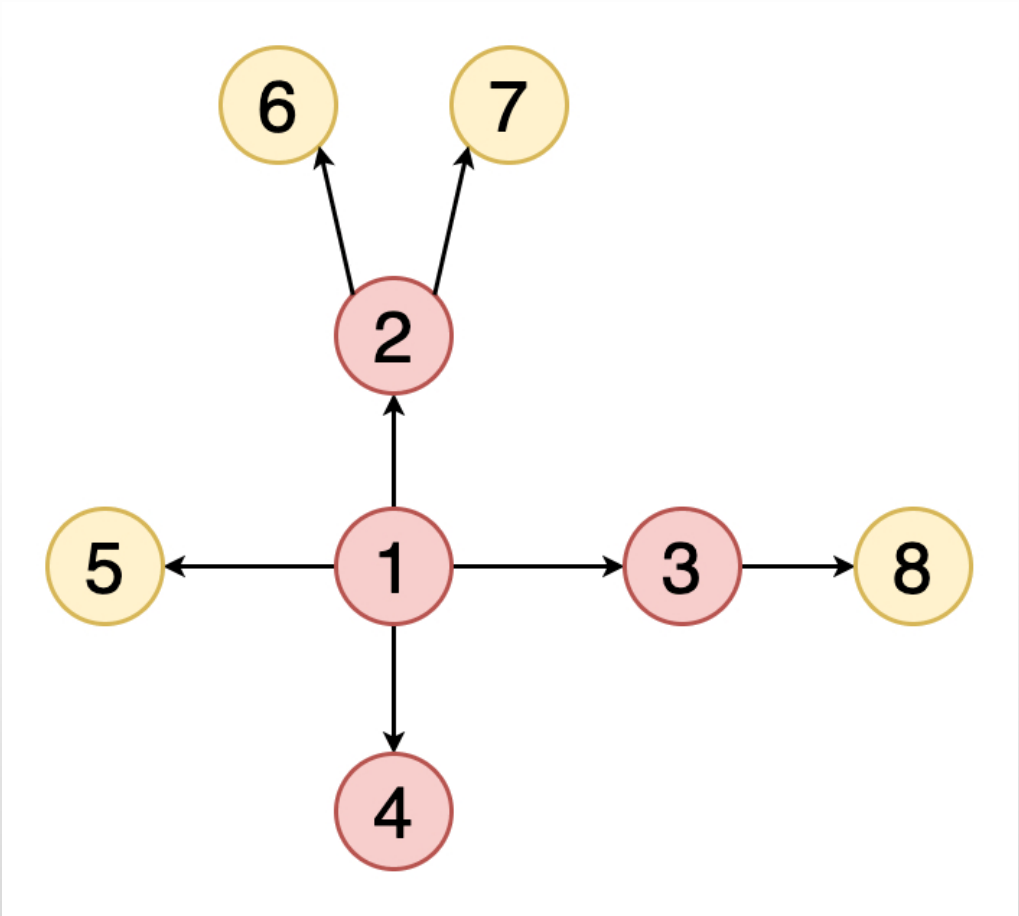
Genişlik Öncelikli Arama



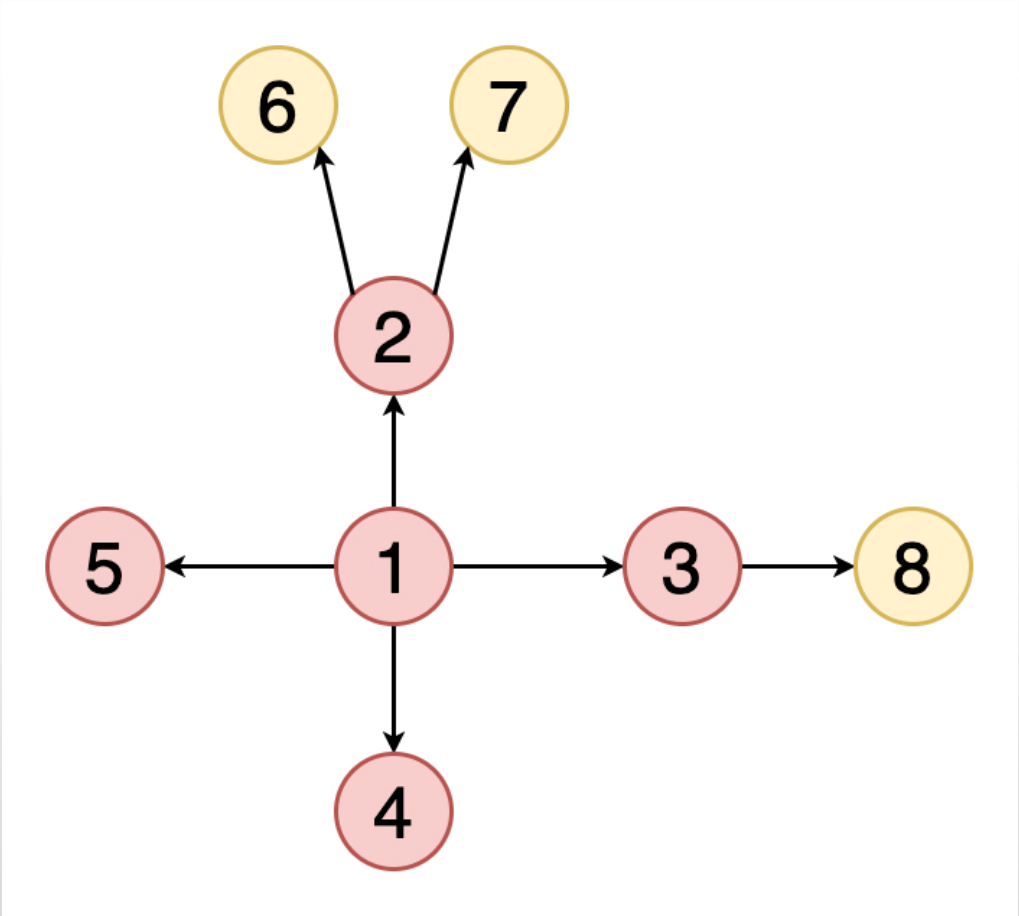
Genişlik Öncelikli Arama



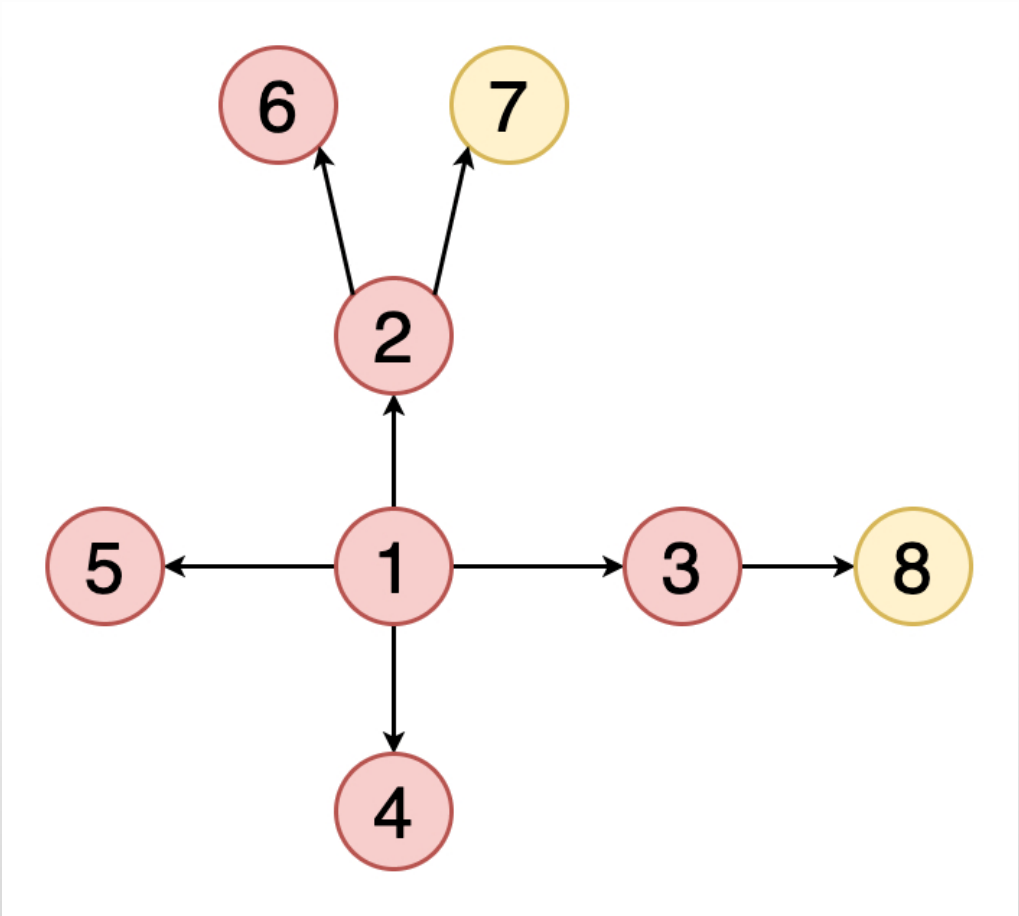
Genişlik Öncelikli Arama



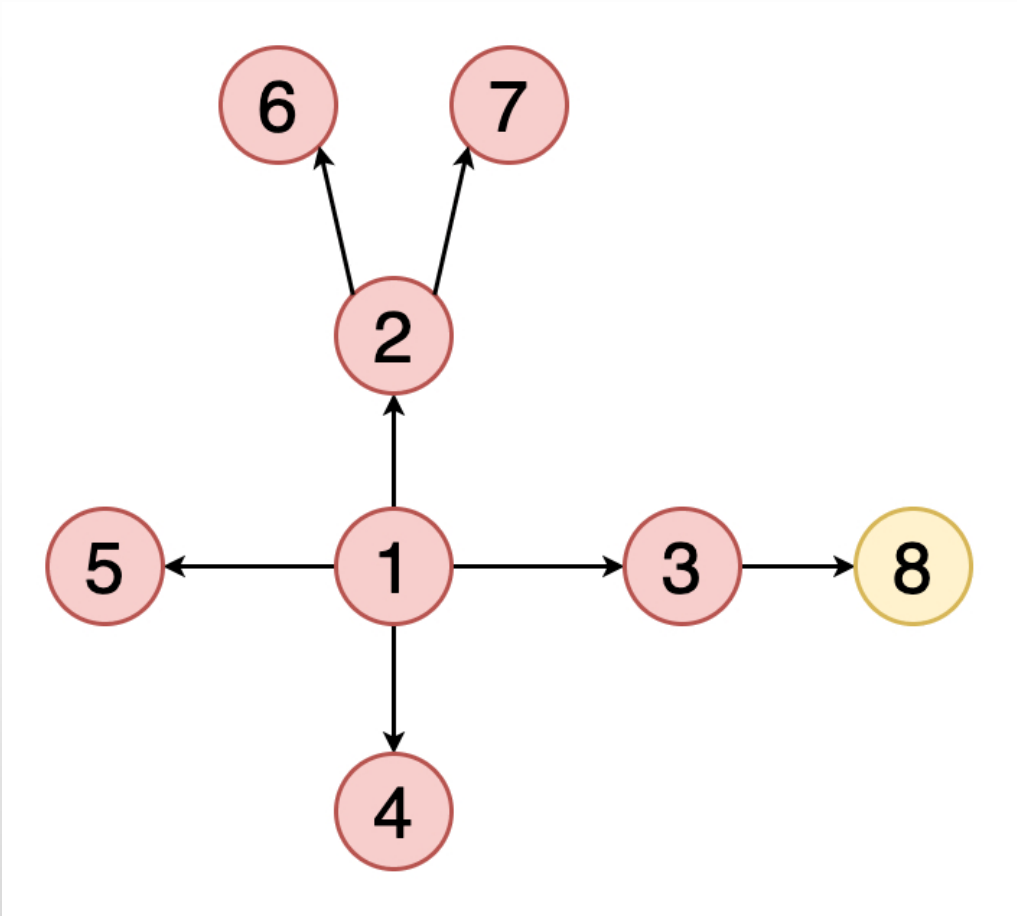
Genişlik Öncelikli Arama



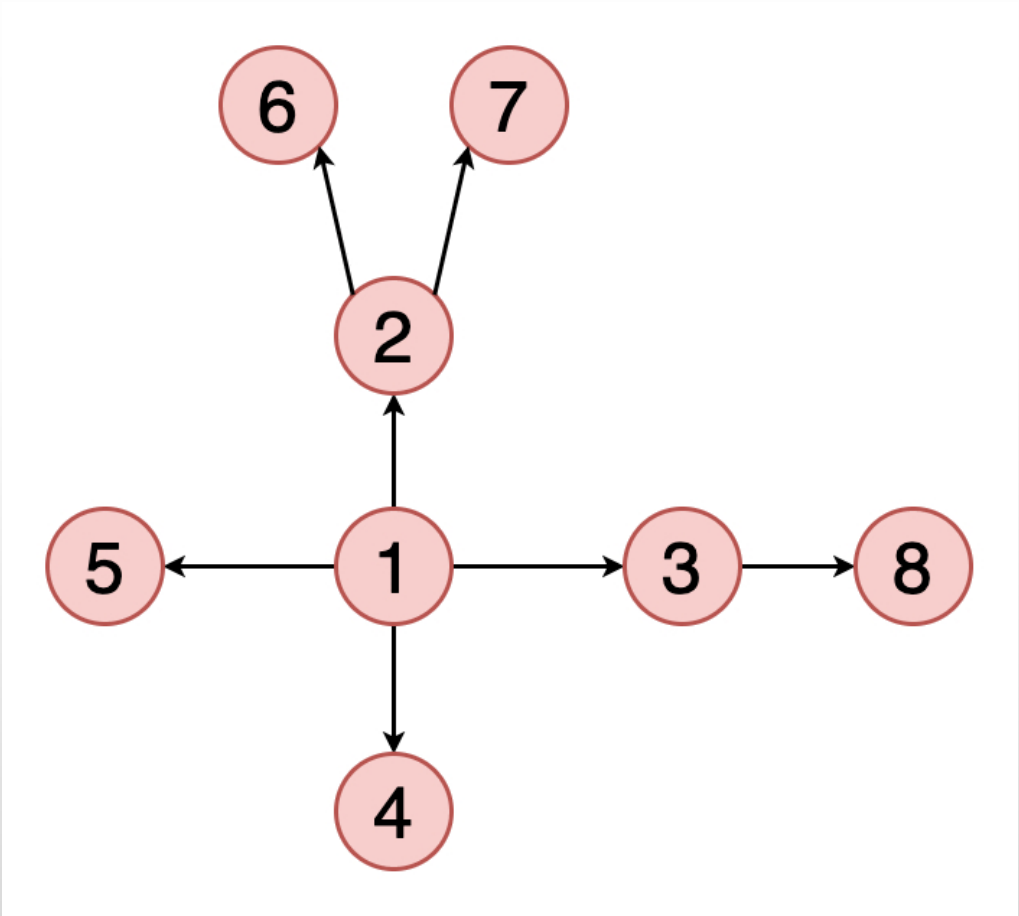
Genişlik Öncelikli Arama



Genişlik Öncelikli Arama



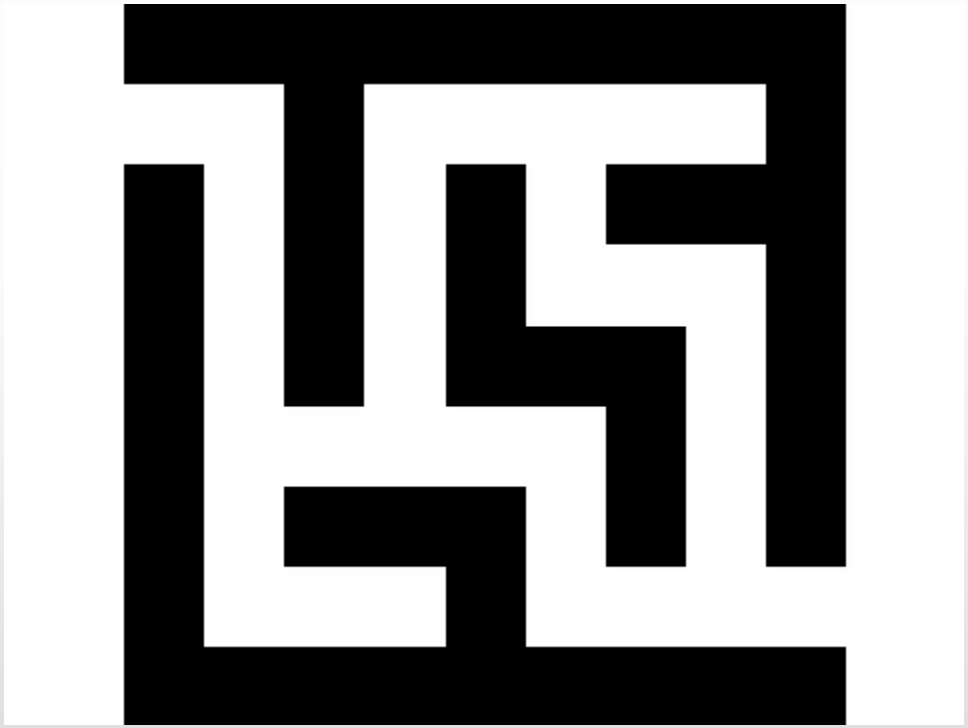
Genişlik Öncelikli Arama



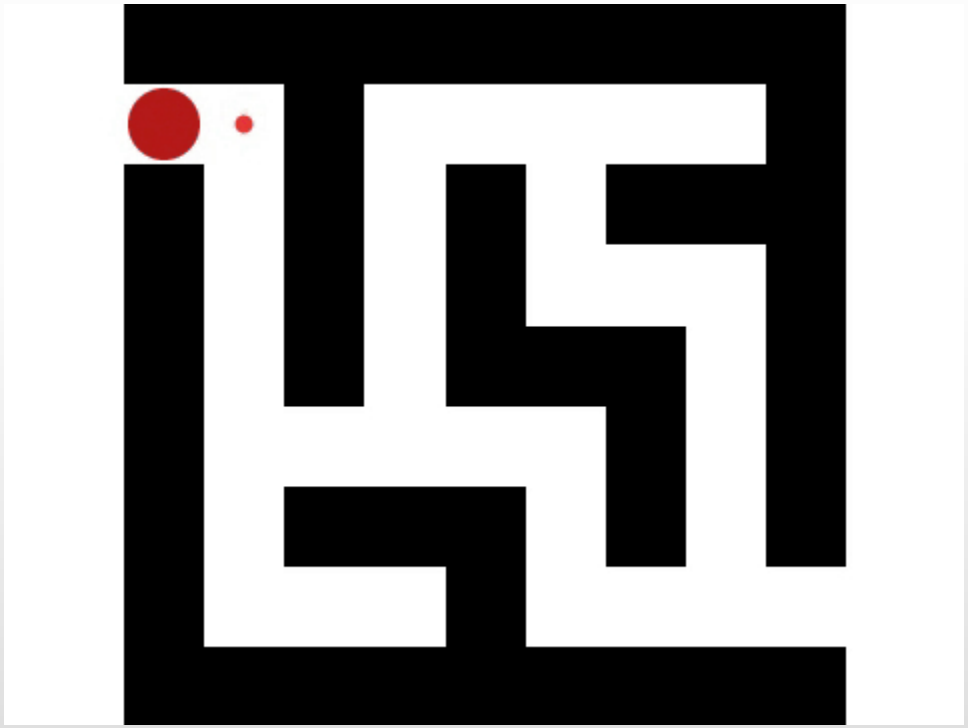
Iterative BFS

```
procedure BFS(G, root) is
    let Q be a queue
    label root as explored
    Q.enqueue(root)
    while Q is not empty do
        v := Q.dequeue()
        if v is the goal then
            return v
        for all edges from v to w in G.adjacentEdges(v) do
            if w is not labeled as explored then
                label w as explored
                w.parent := v
                Q.enqueue(w)
```

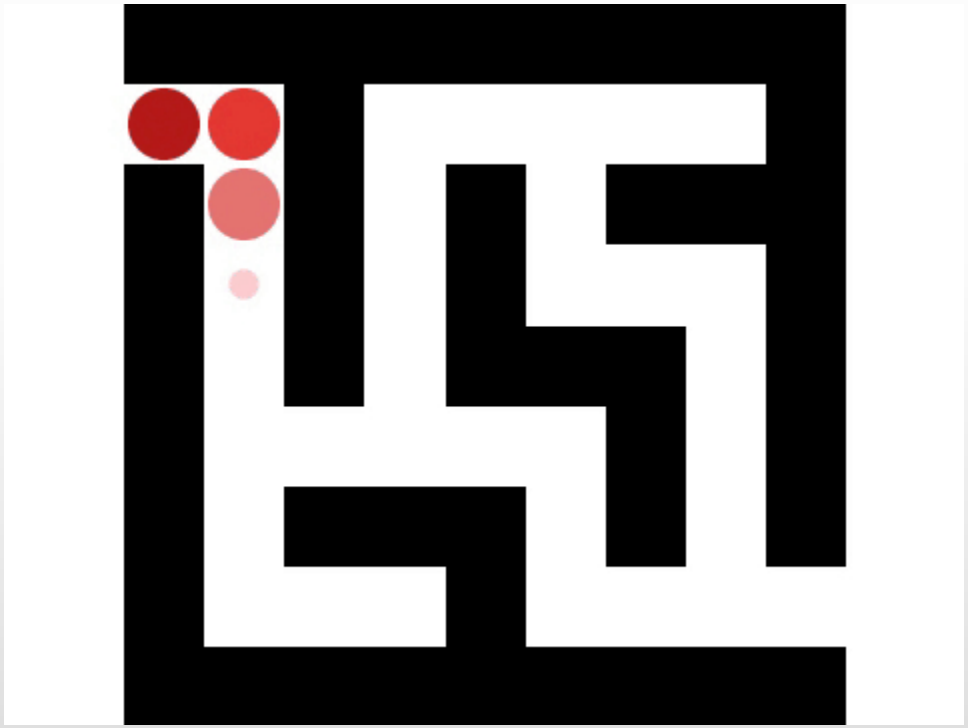
BFS Search Way



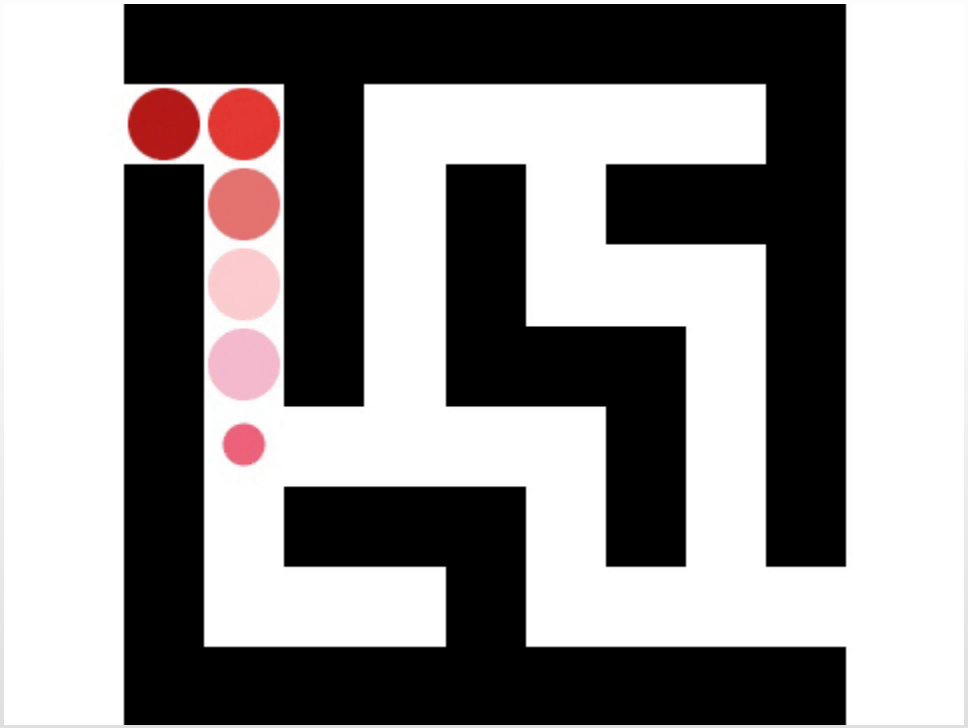
BFS Search Way



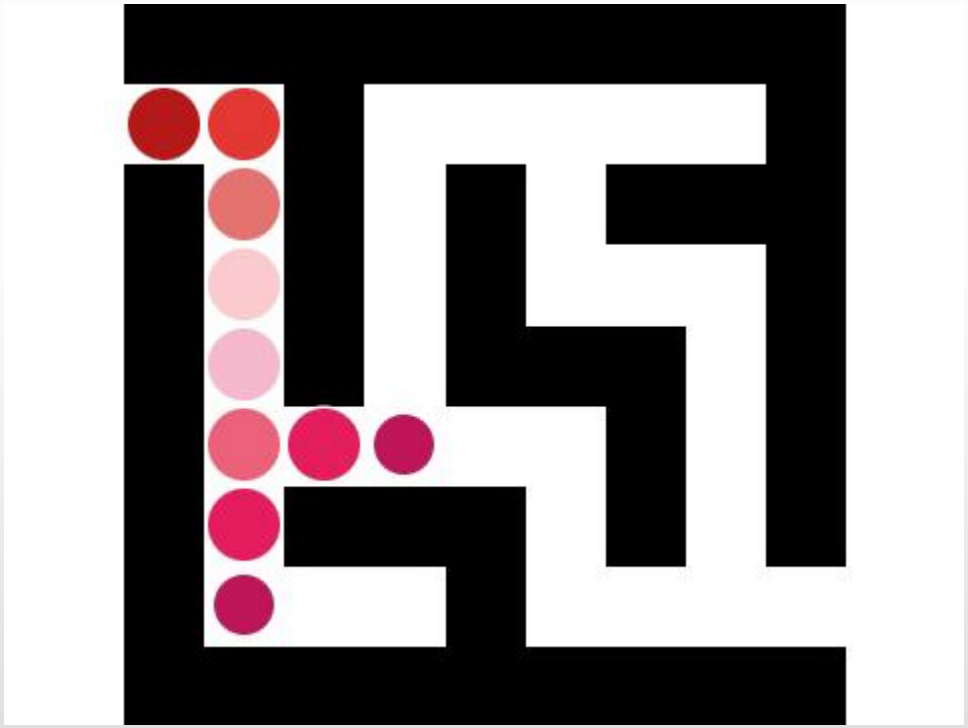
BFS Search Way



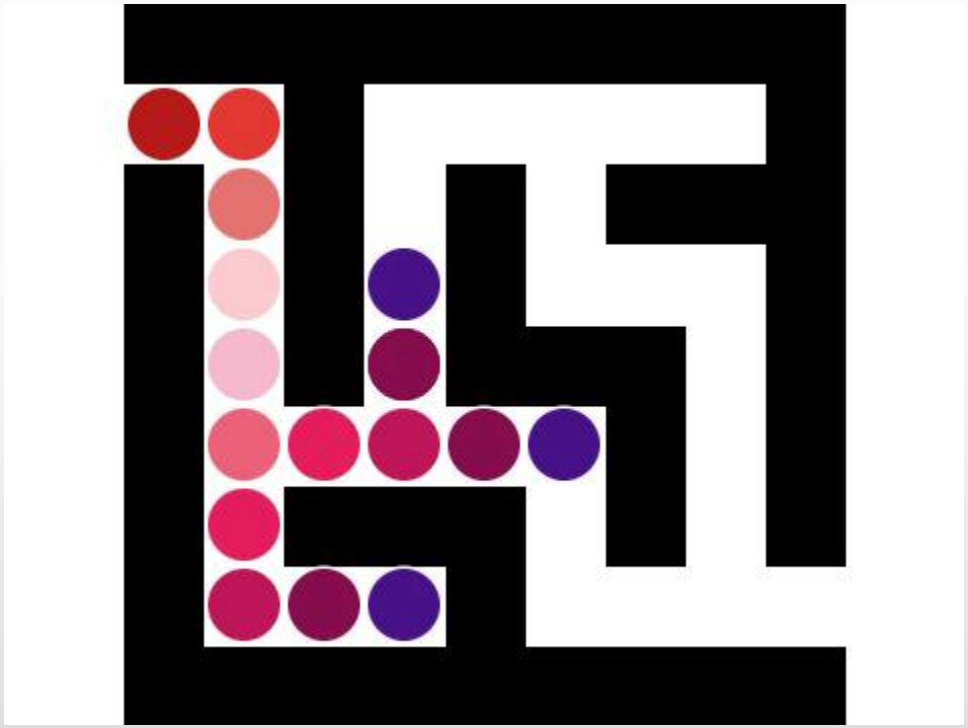
BFS Search Way



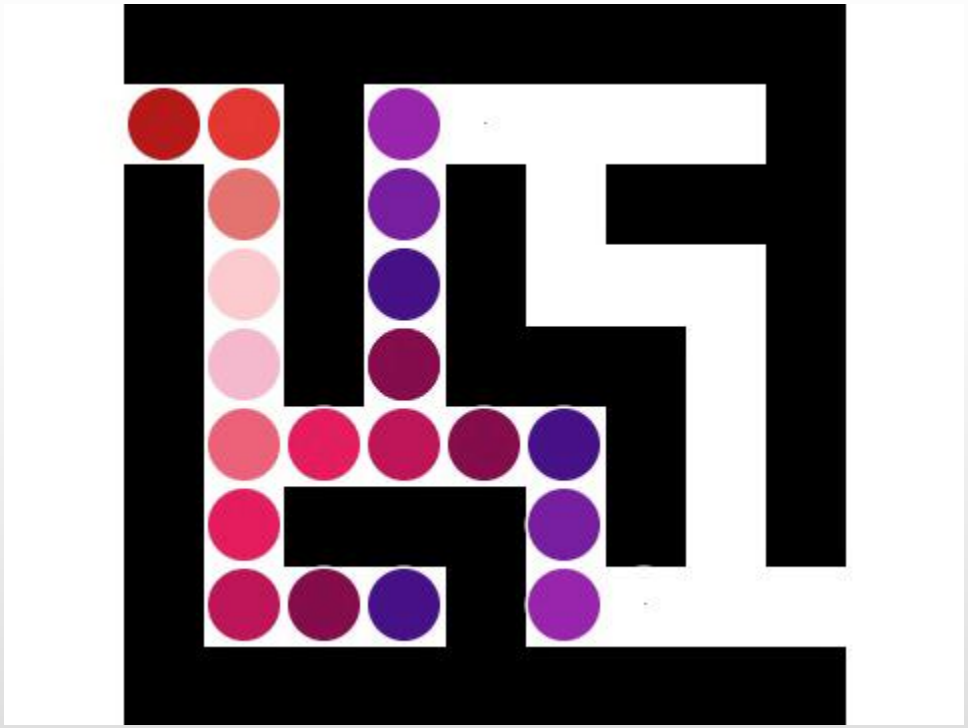
BFS Search Way



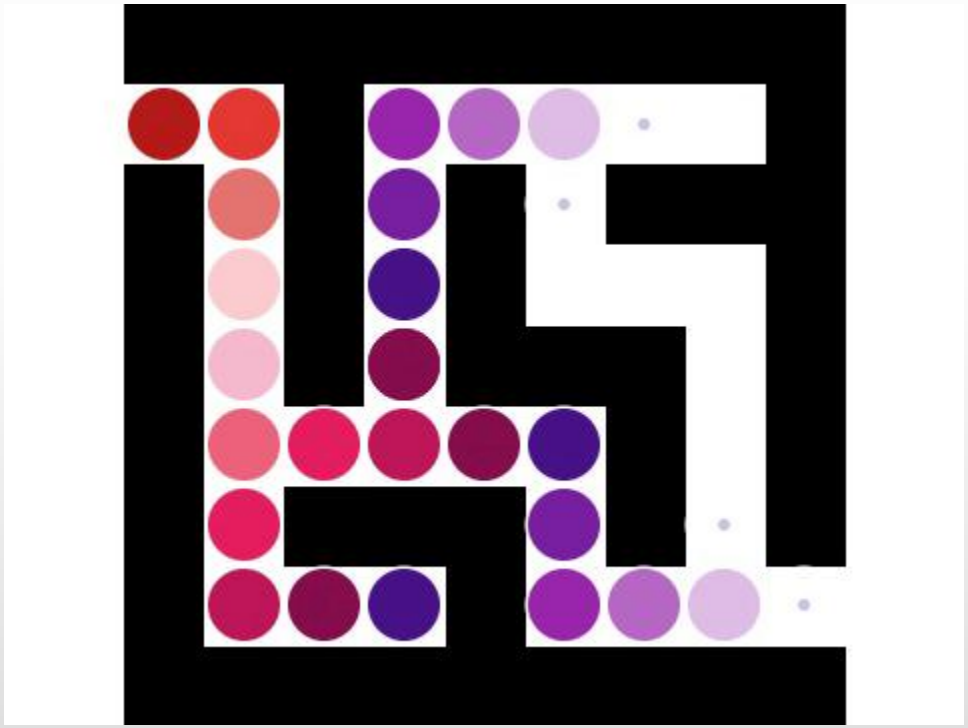
BFS Search Way



BFS Search Way



BFS Search Way



Tarjan'ın Güçlü Bağlantılı Bileşenler Algoritması

- Yönlü çizgede güçlü bağlantılı bileşenleri bulur.
- Güçlü bağlantılı bileşenler, (*strongly connected components*)
  - iki düğüm arasında hem ileri hem de geri yönlü yolların bulunduğu bileşenlerdir.
  - bir düğüm ve ona erişilebilen tüm düğümleri içeren alt çizge.

Algoritma İlkeleri

- DFS (Depth-First Search) tabanlı bir algoritmadır.
- Çizgenin düğümlerini sıra ile ziyaret eder.
- Her ziyaret edilen düğüm için, o düğümden ulaşılabilen en düşük düğüm numarası (index) hesaplanır.

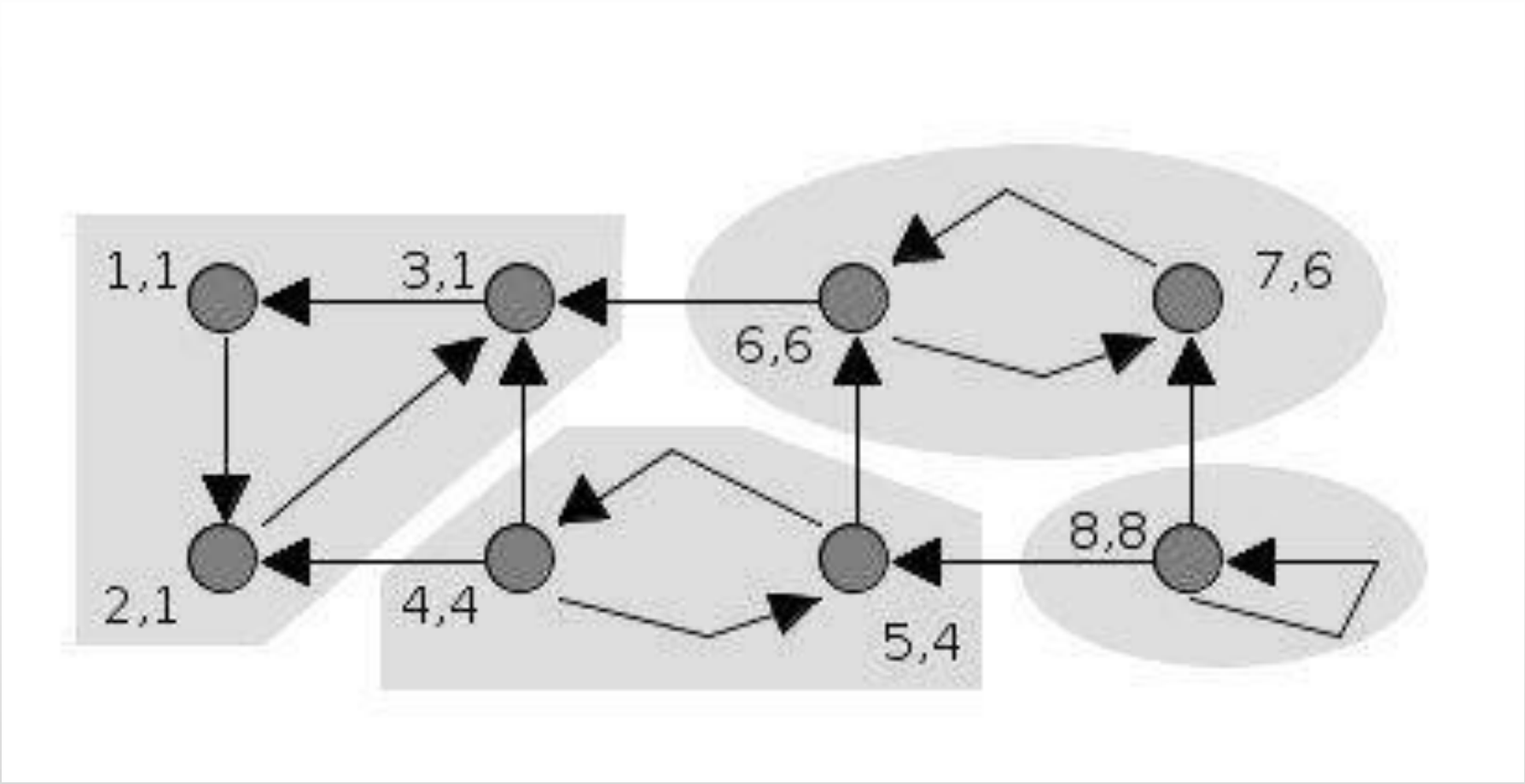
Algoritma Adımları

- Adım 1: Her bir düğüm DFS algoritması ile ziyaret edilir.
- Adım 2: DFS sırasında, ziyaret edilen her düğüm bir numara alır. Bu numara, düğümün çizge içindeki konumunu temsil eder.
- Adım 3: Her düğüm için, düğümden ulaşılabilen en düşük düğüm numarası hesaplanır. Bu, çizgedeki ileri ve geri yönlü yolları kontrol eder.
- Adım 4: Düğüm numaraları ve en düşük düğüm numaraları kullanılarak güçlü bağlantılı bileşenler bulunur.

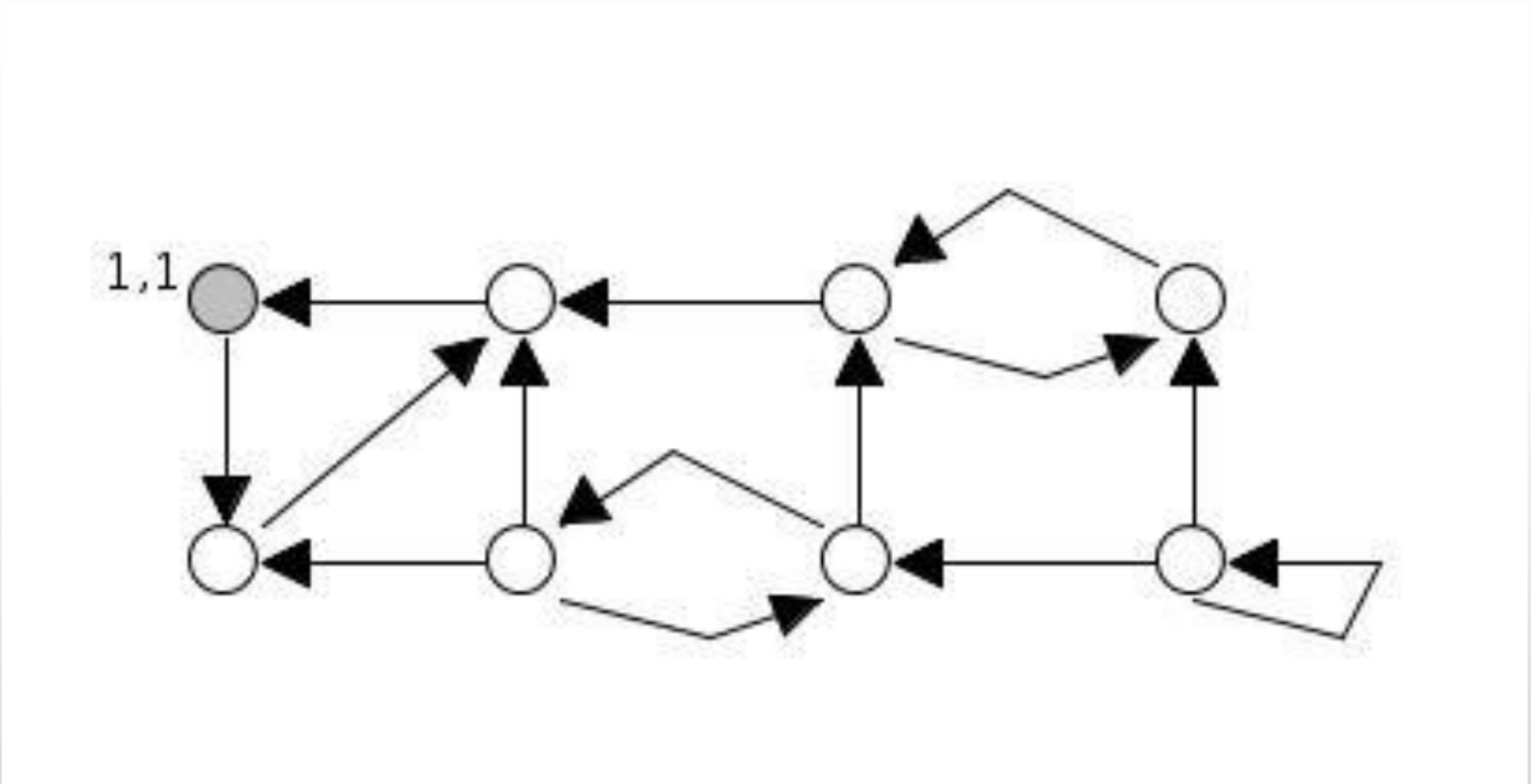
Karmaşıklık Analizi

- Tarjan'ın Algoritması'nın karmaşıklığı,
  - *O(V+E)*
  - V düğüm sayısı
  - E kenar sayısı

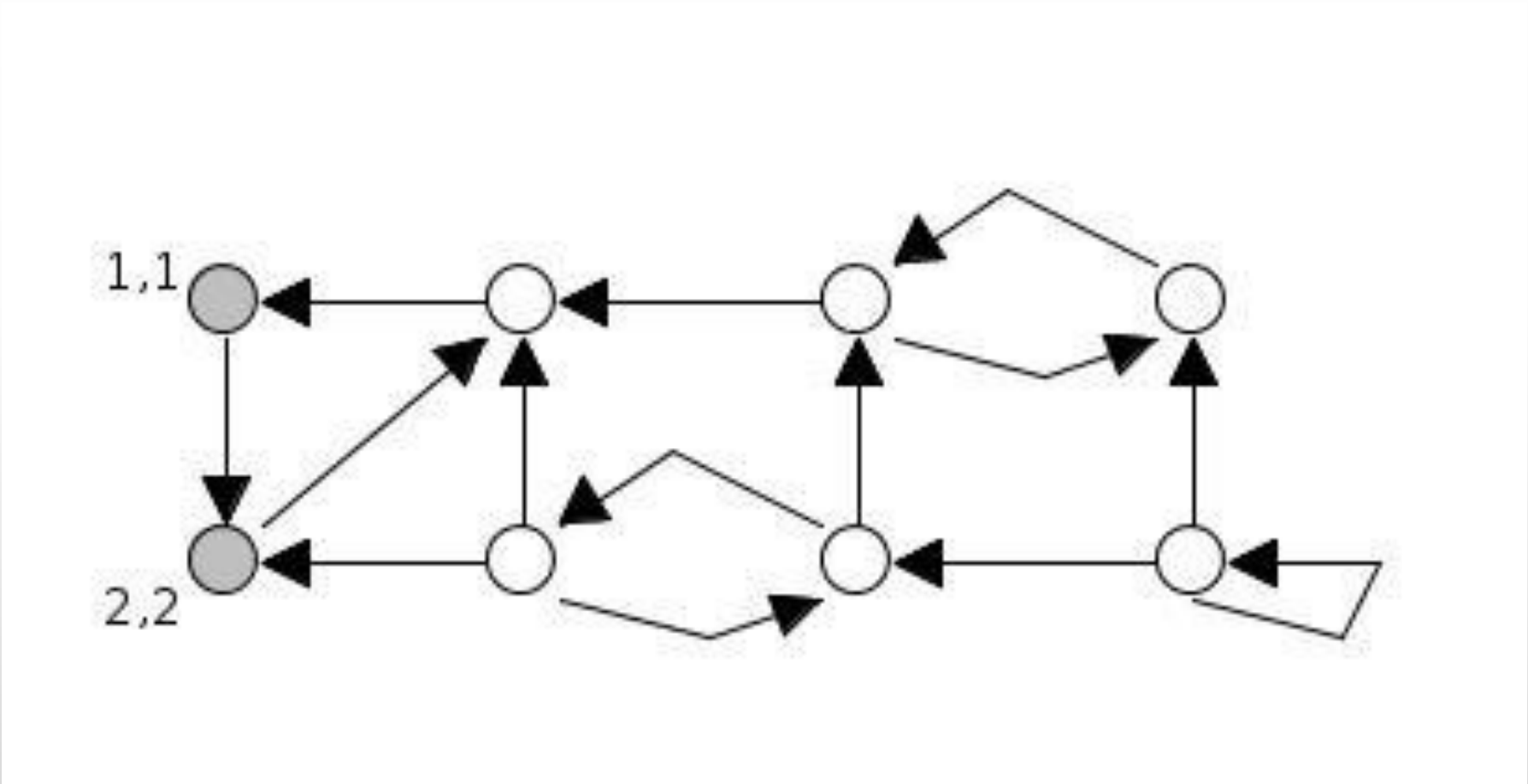
Tarjan's Strongly Connected Components



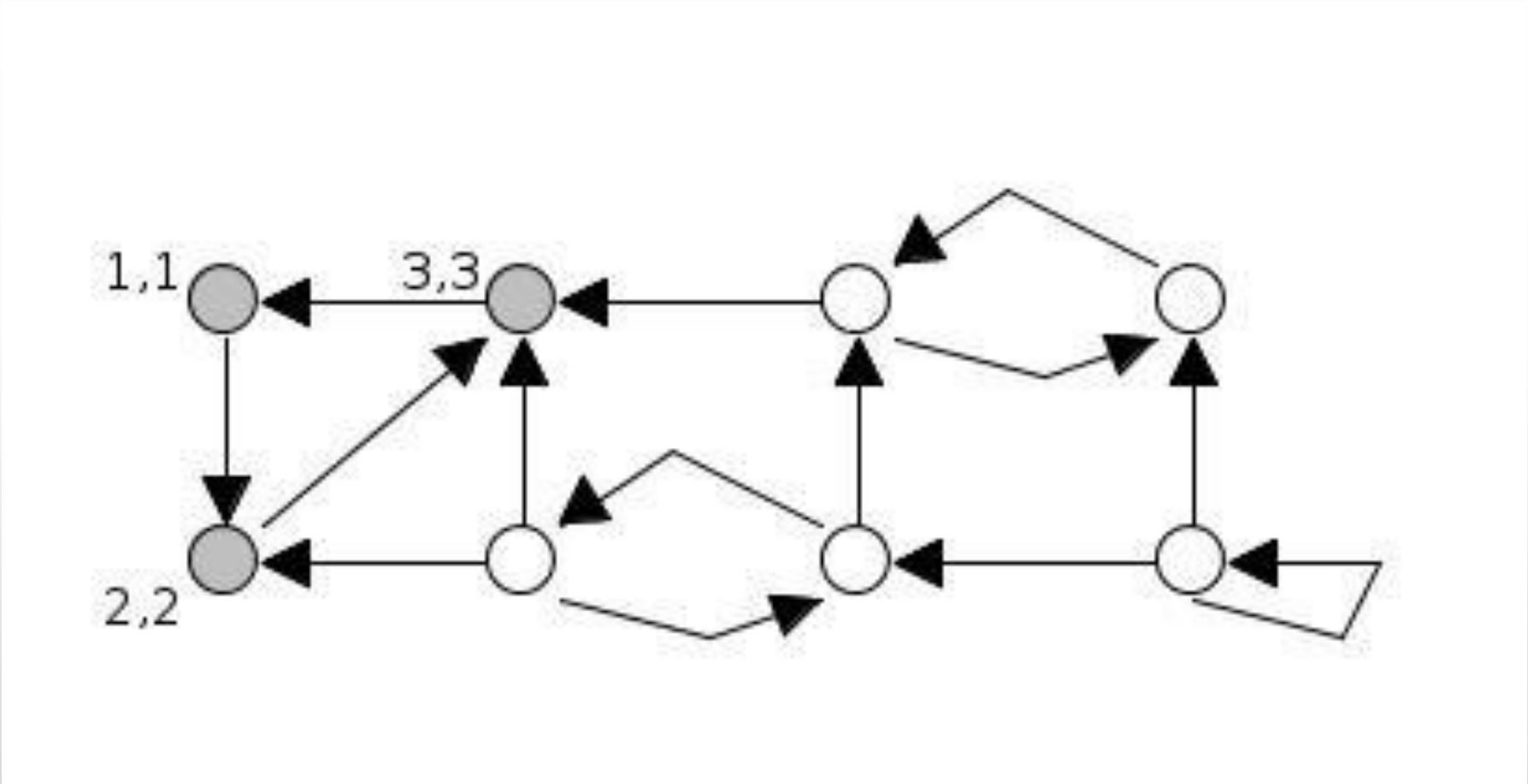
Tarjan's Strongly Connected Components



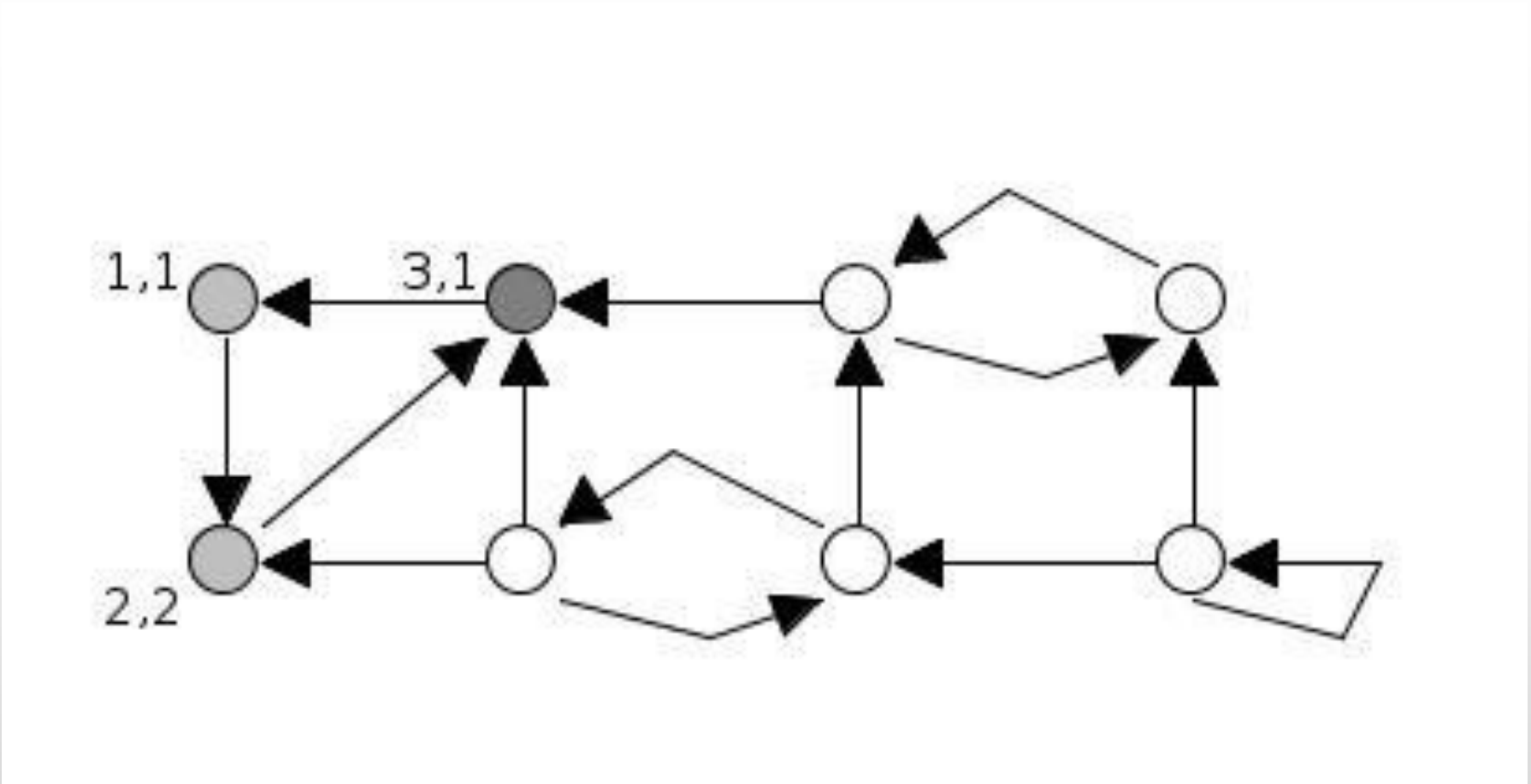
Tarjan's Strongly Connected Components



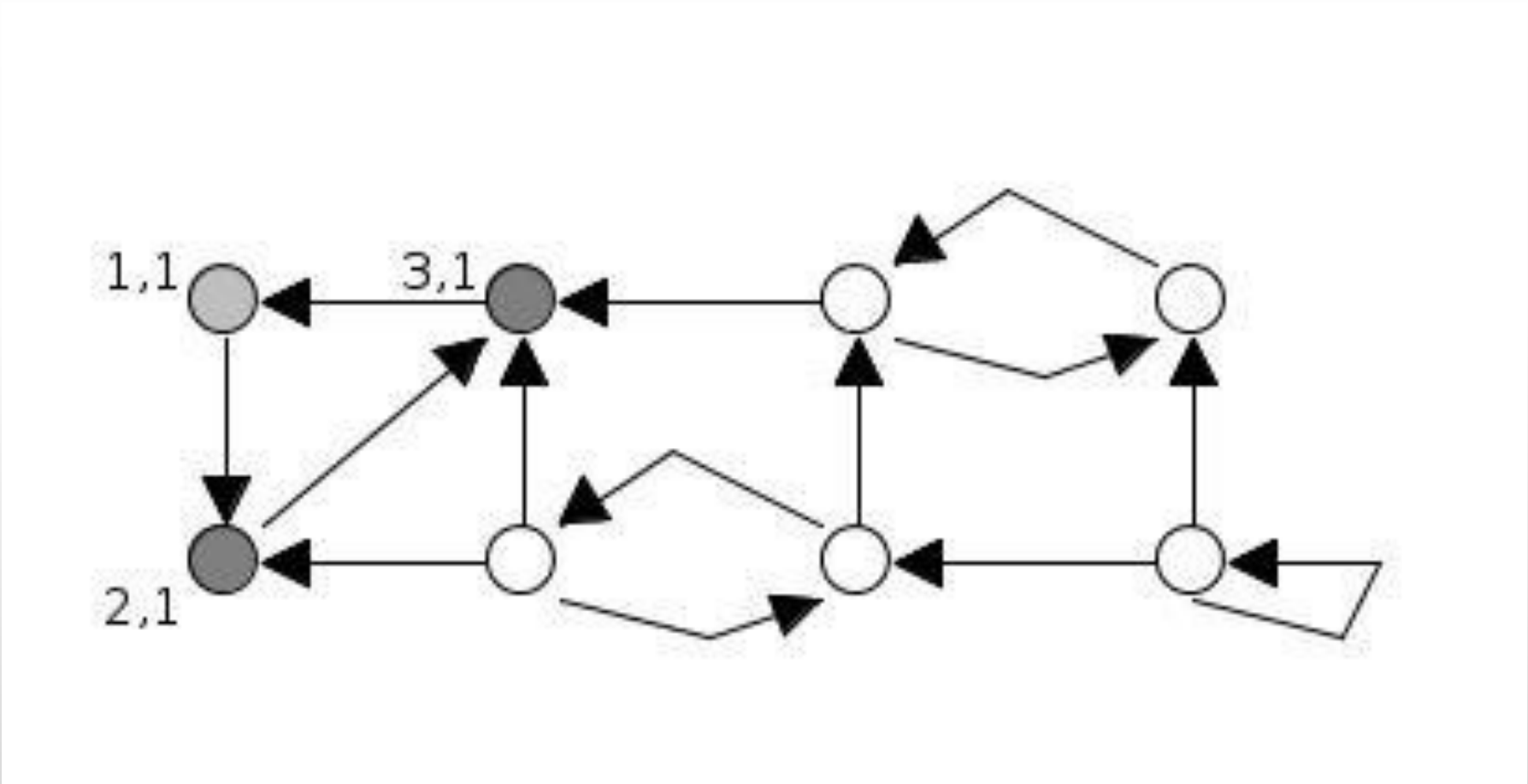
Tarjan's Strongly Connected Components



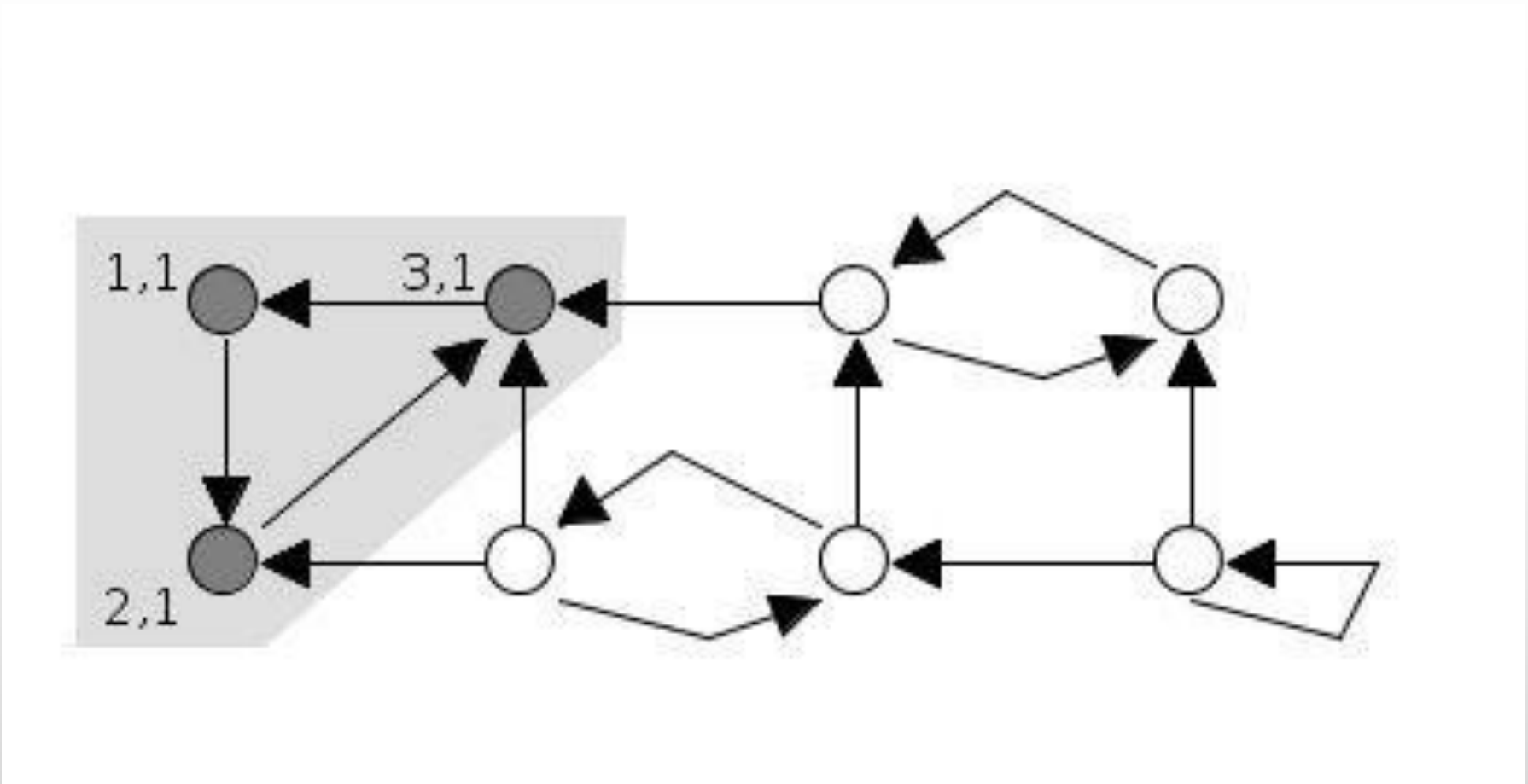
Tarjan's Strongly Connected Components



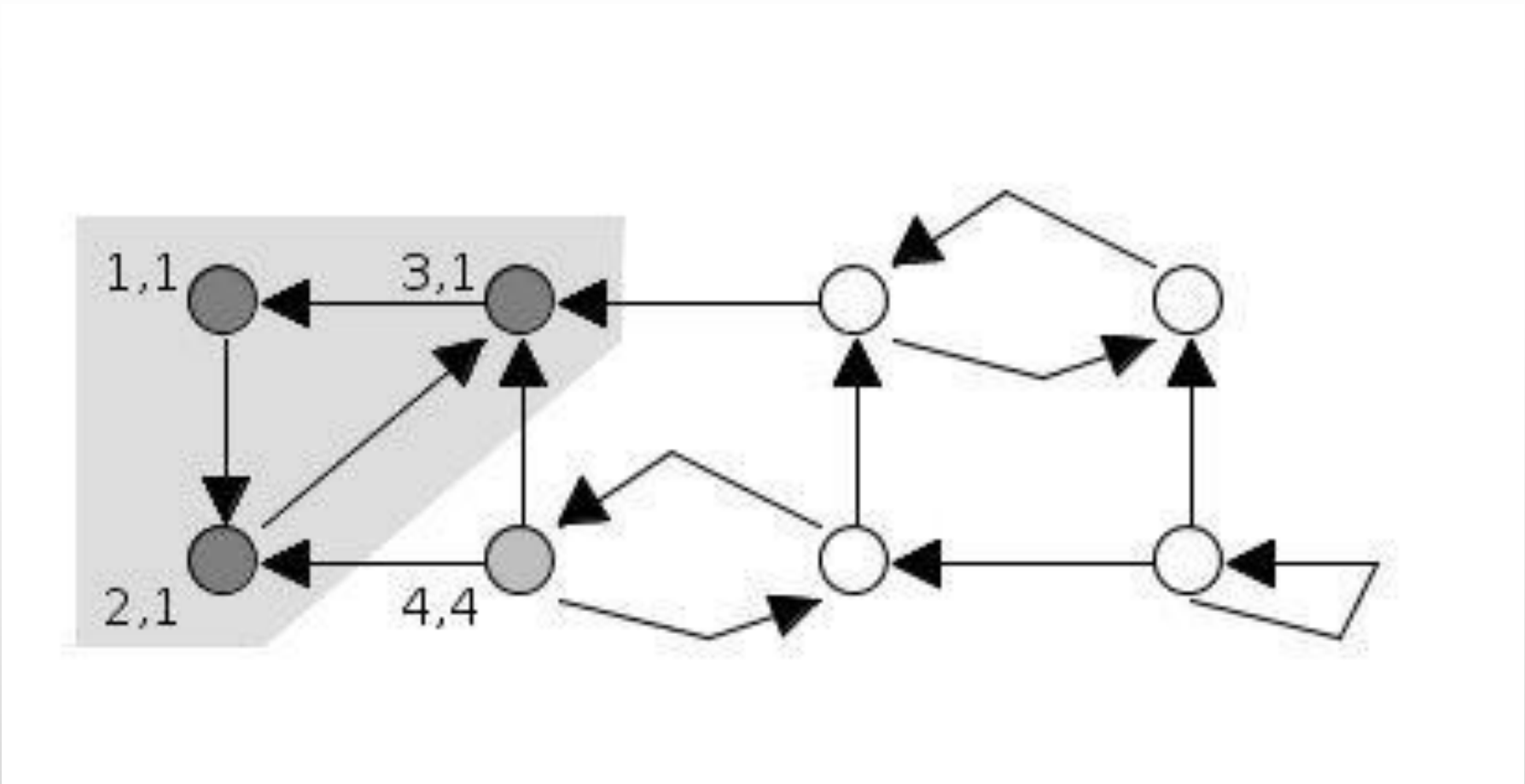
Tarjan's Strongly Connected Components



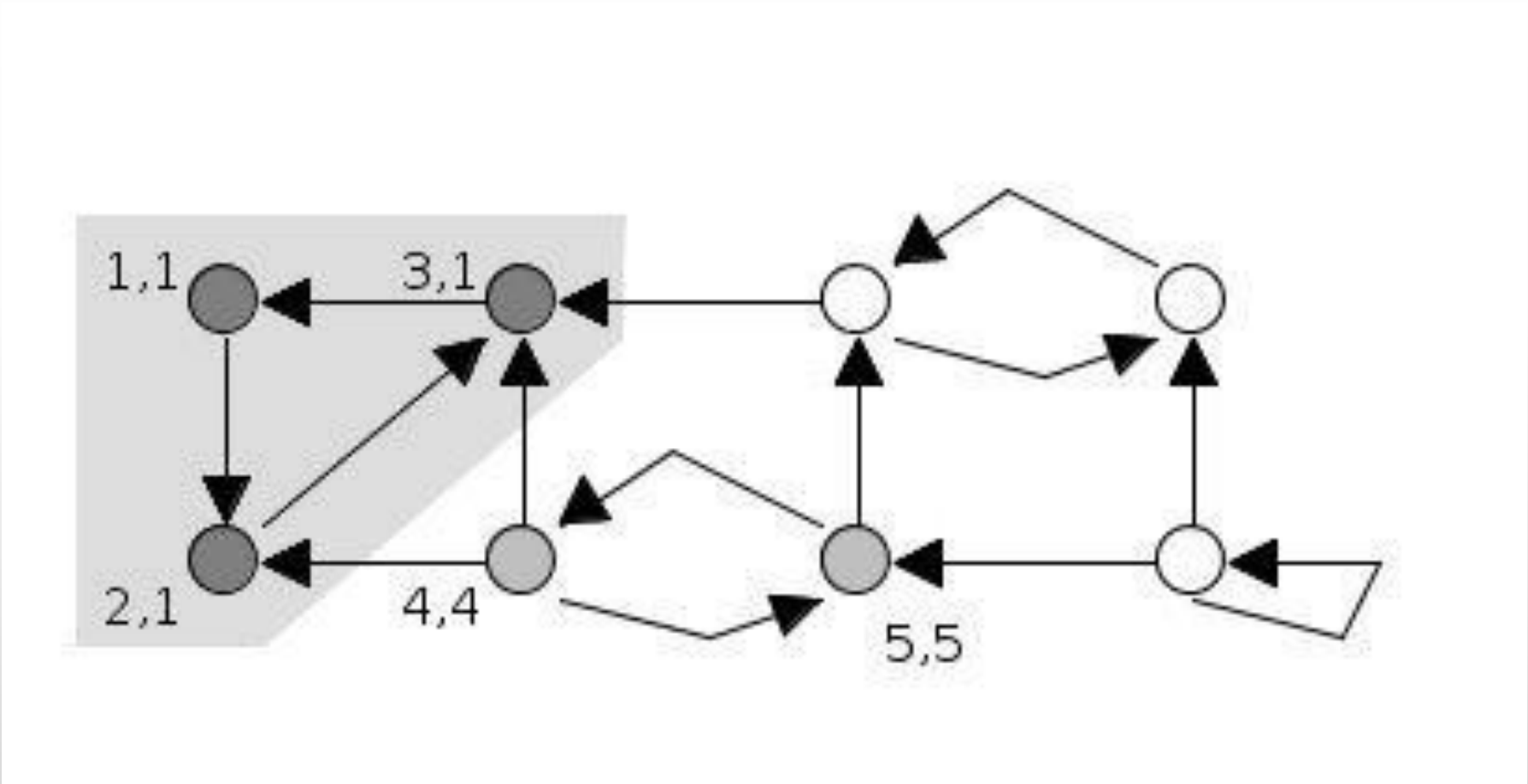
Tarjan's Strongly Connected Components



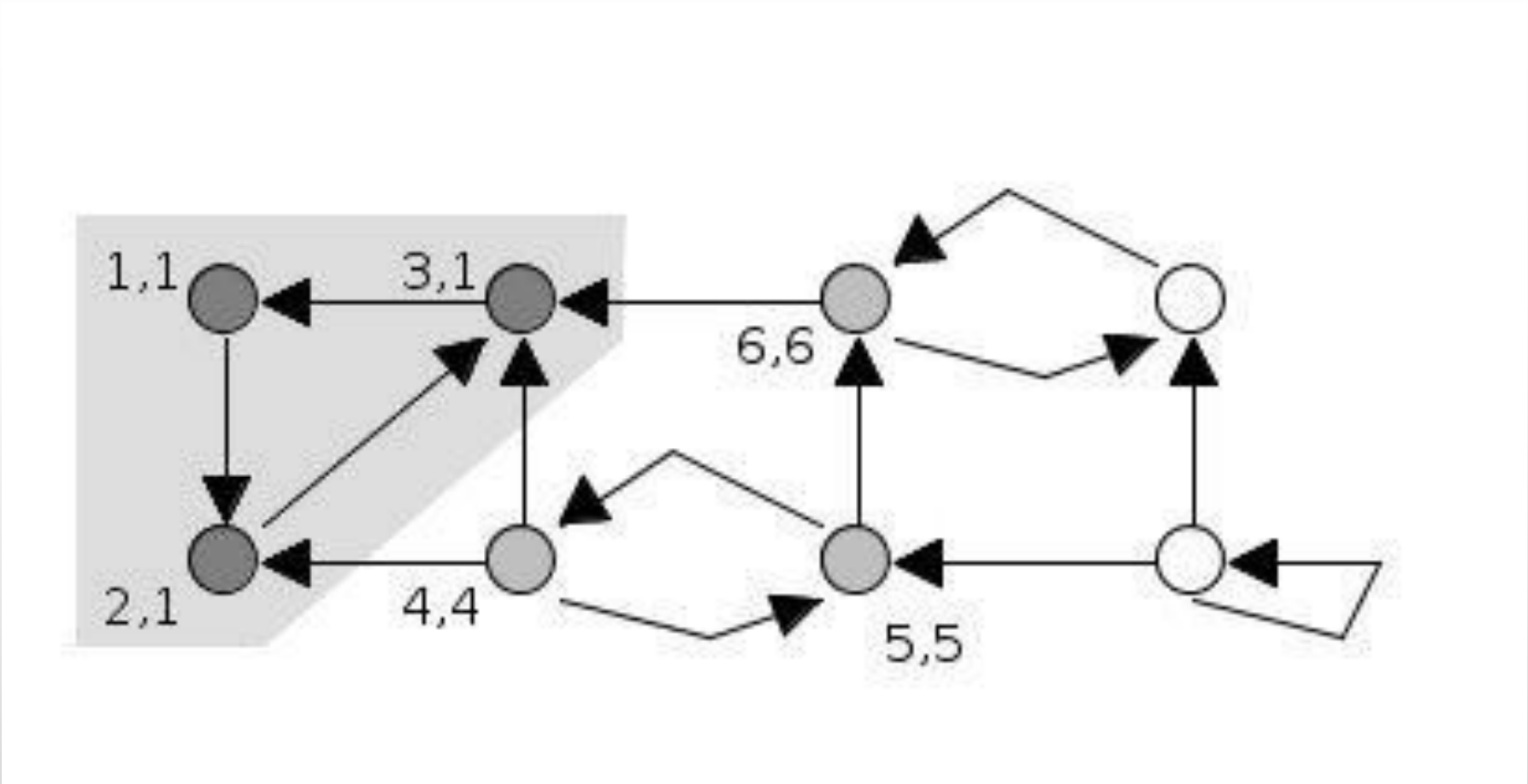
Tarjan's Strongly Connected Components



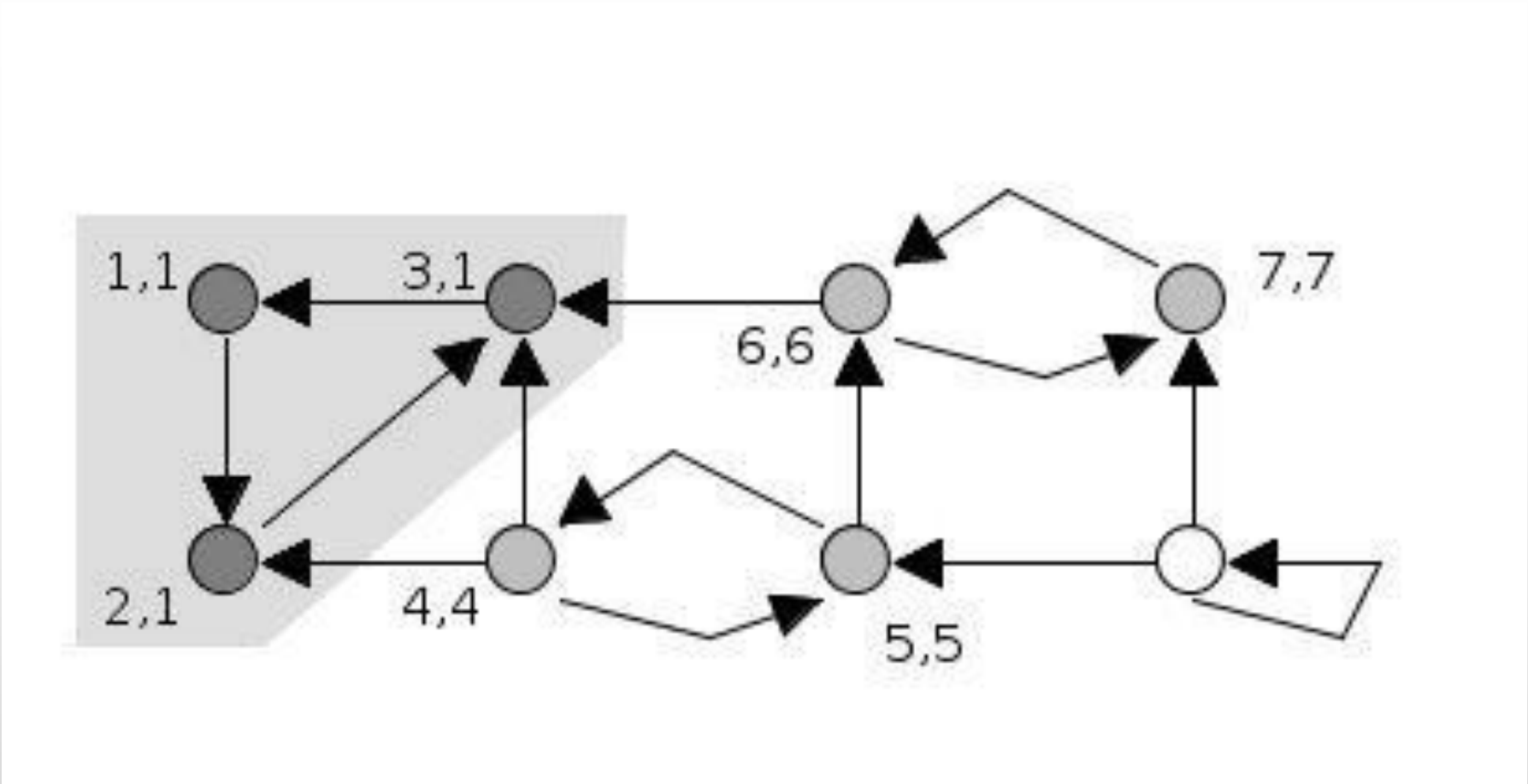
Tarjan's Strongly Connected Components



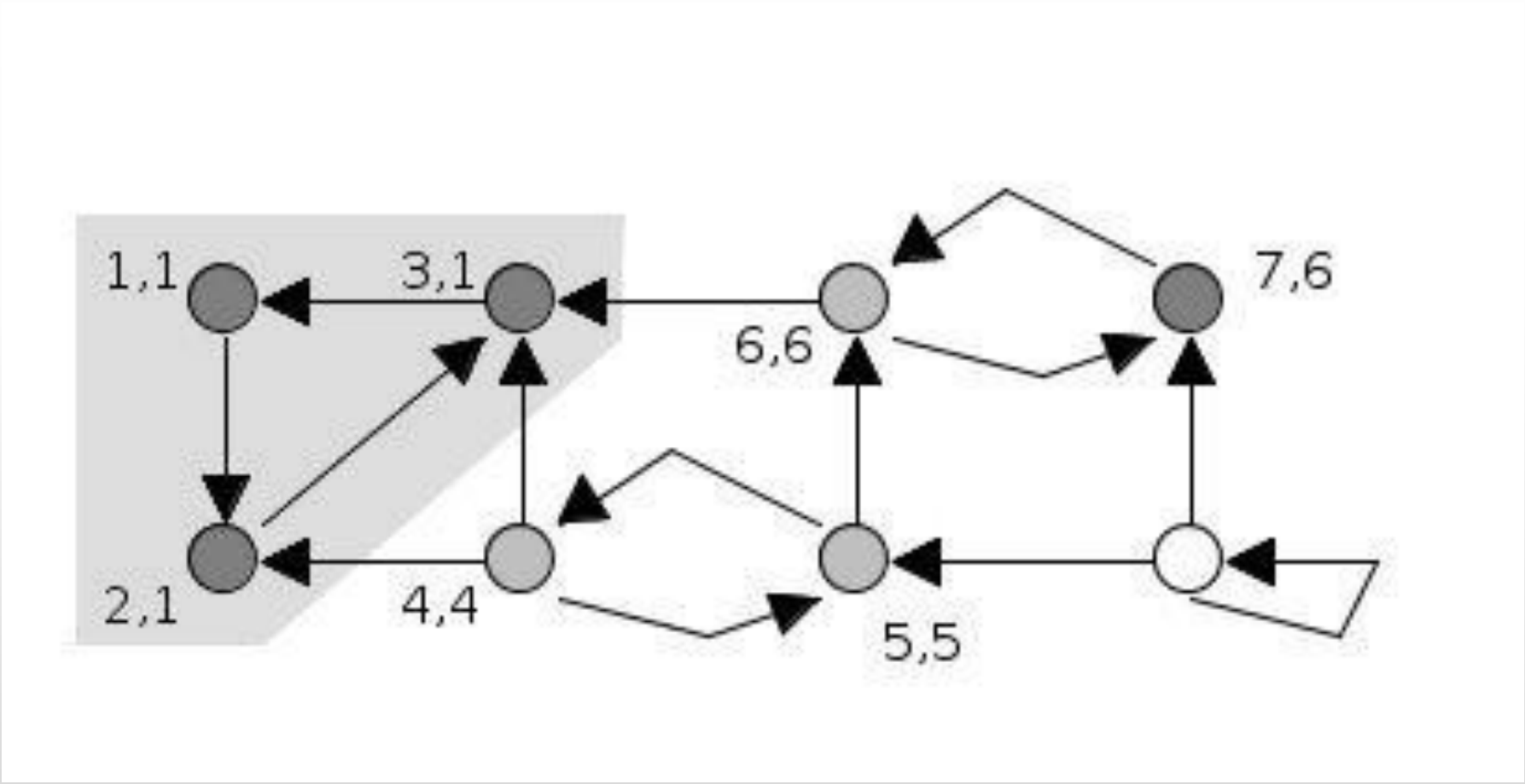
Tarjan's Strongly Connected Components



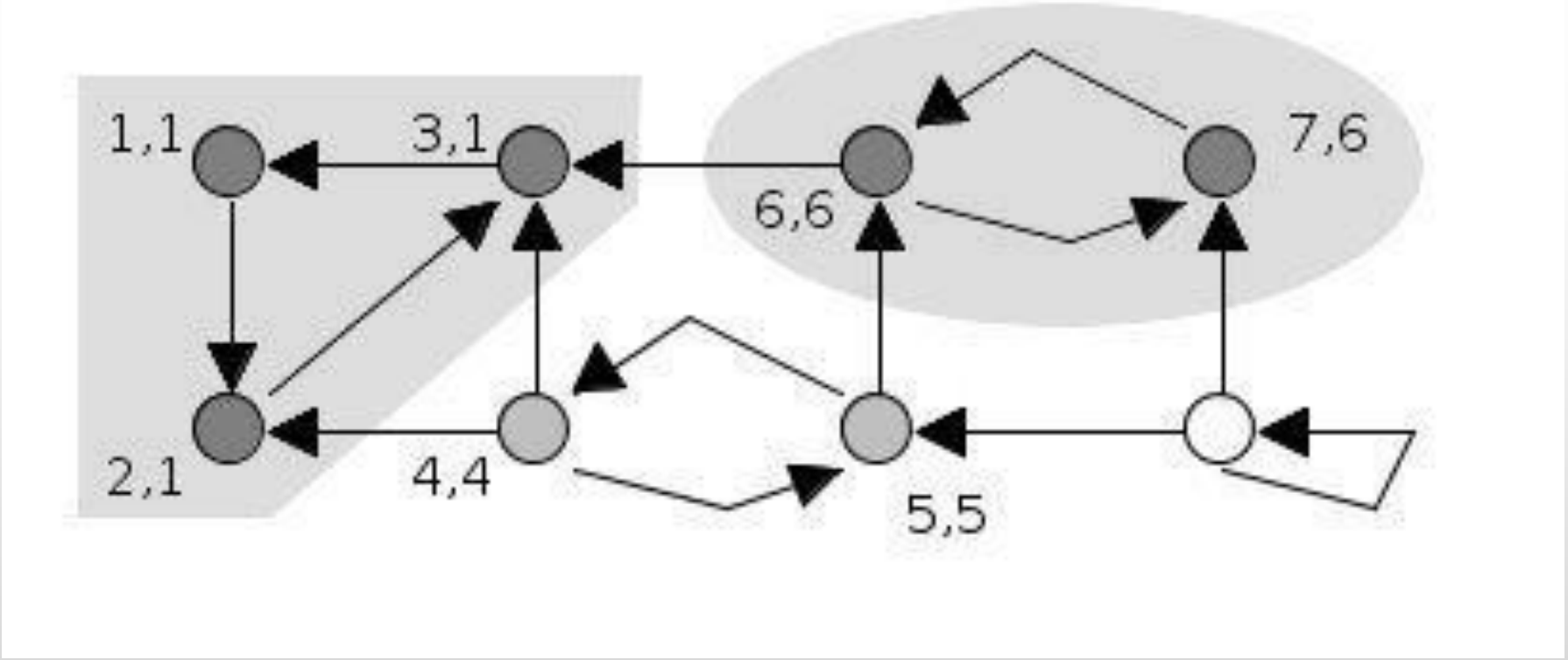
Tarjan's Strongly Connected Components



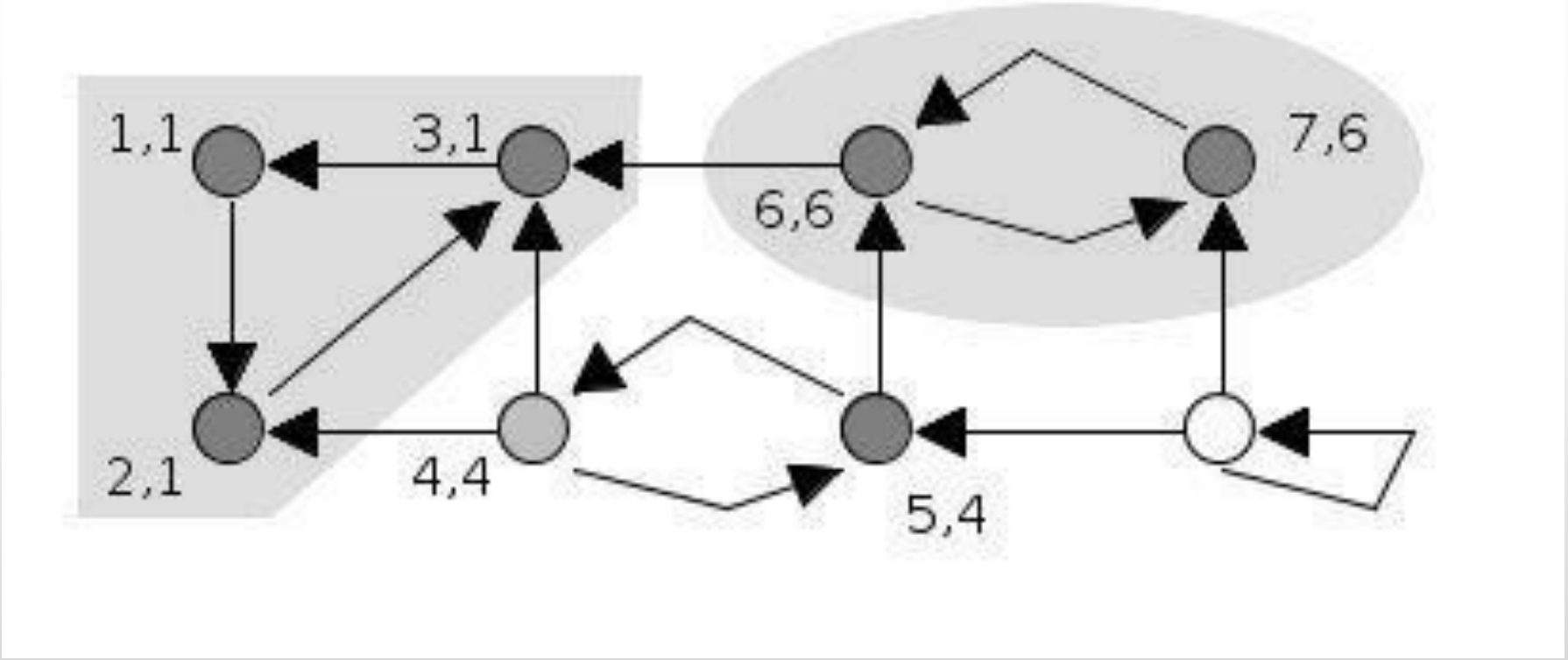
Tarjan's Strongly Connected Components



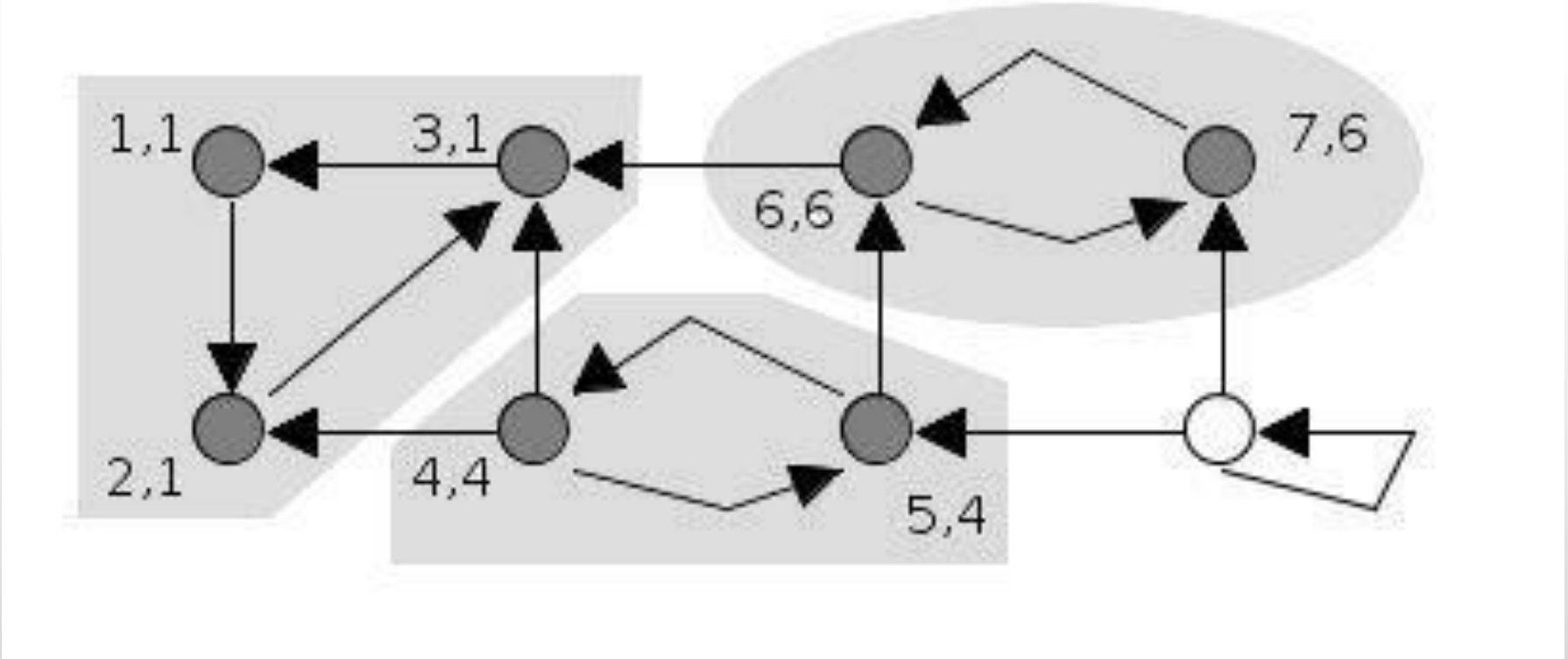
Tarjan's Strongly Connected Components



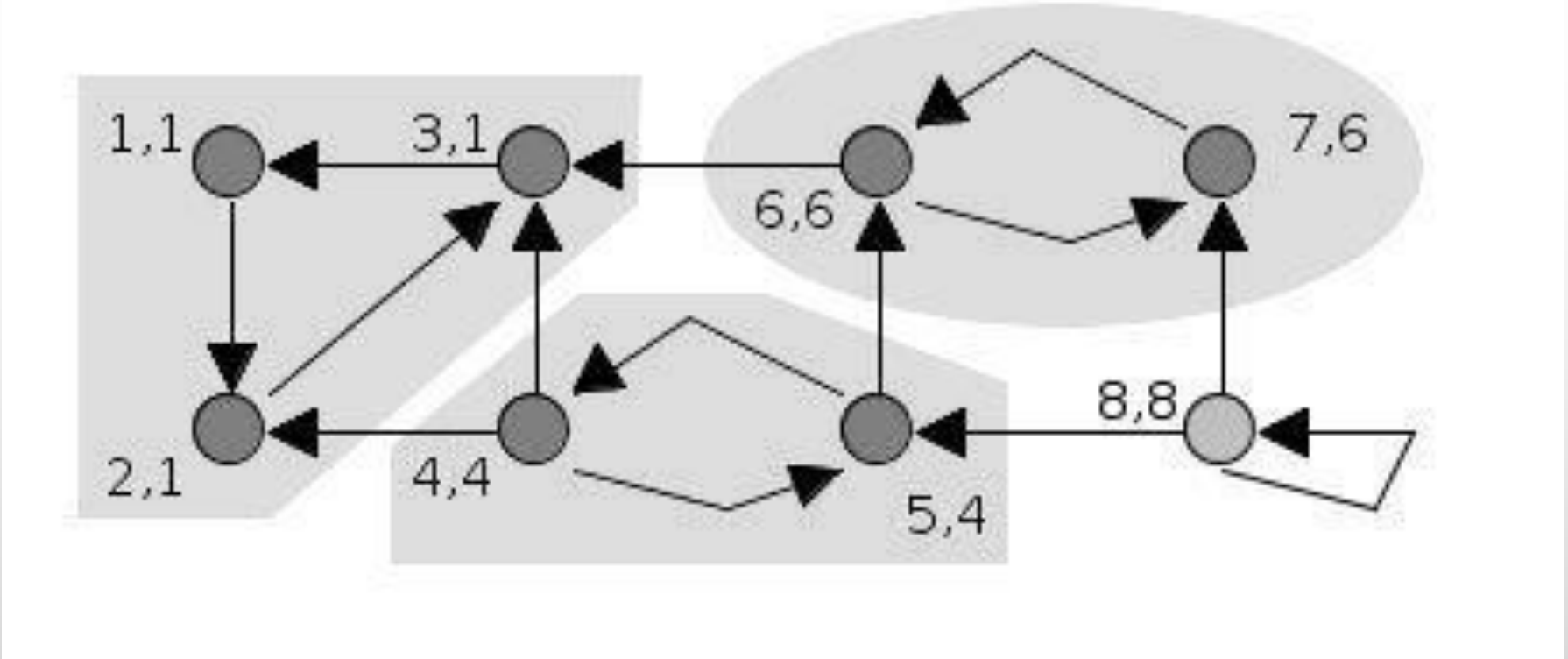
Tarjan's Strongly Connected Components



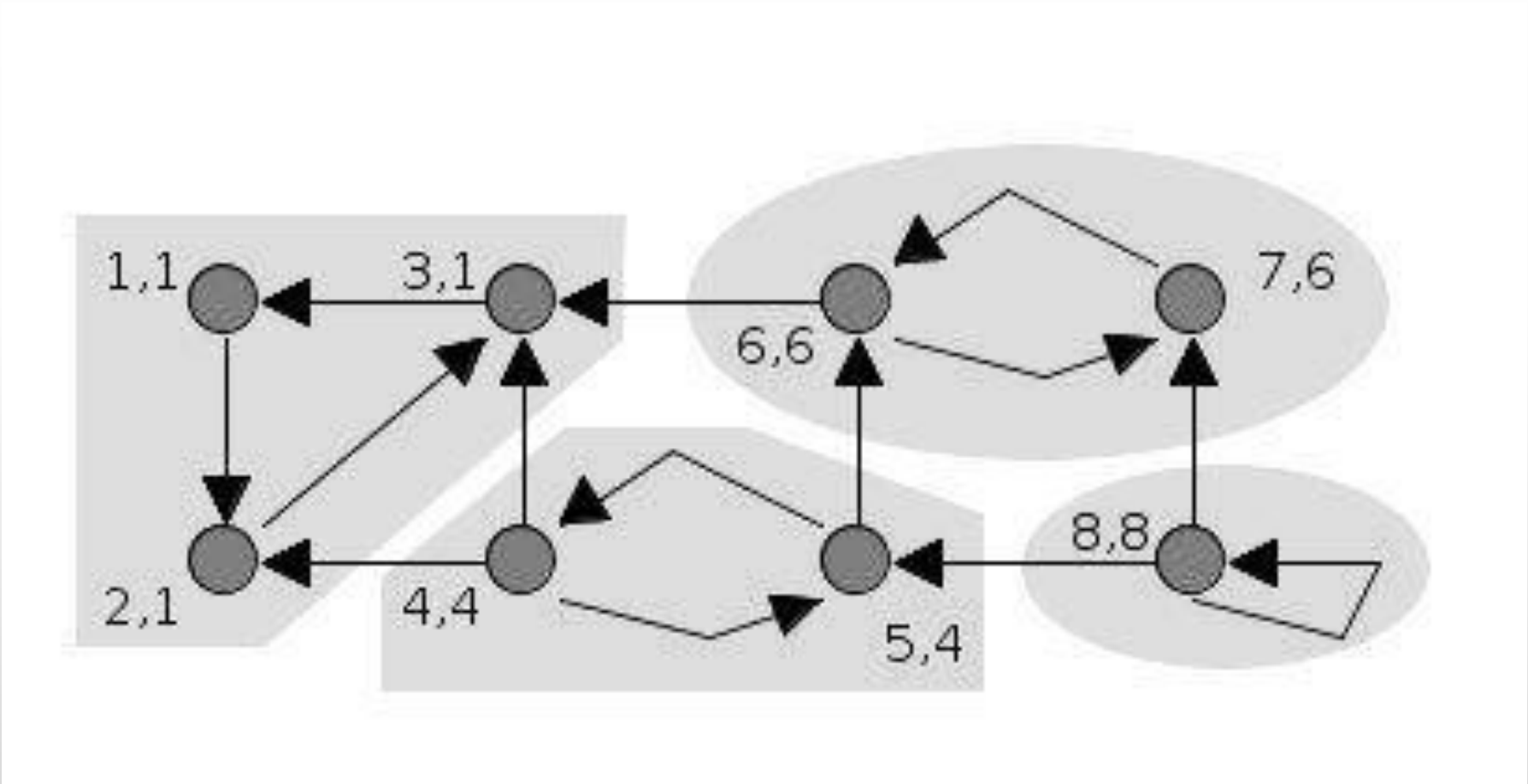
Tarjan's Strongly Connected Components



Tarjan's Strongly Connected Components



Tarjan's Strongly Connected Components



SON