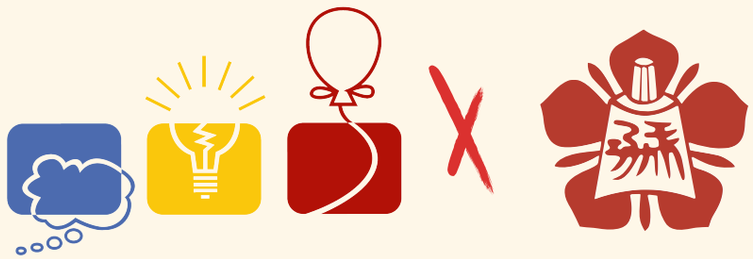
Traversal

郭至軒（KuoE0）

KuoE0.tw@gmail.com
KuoE0.ch

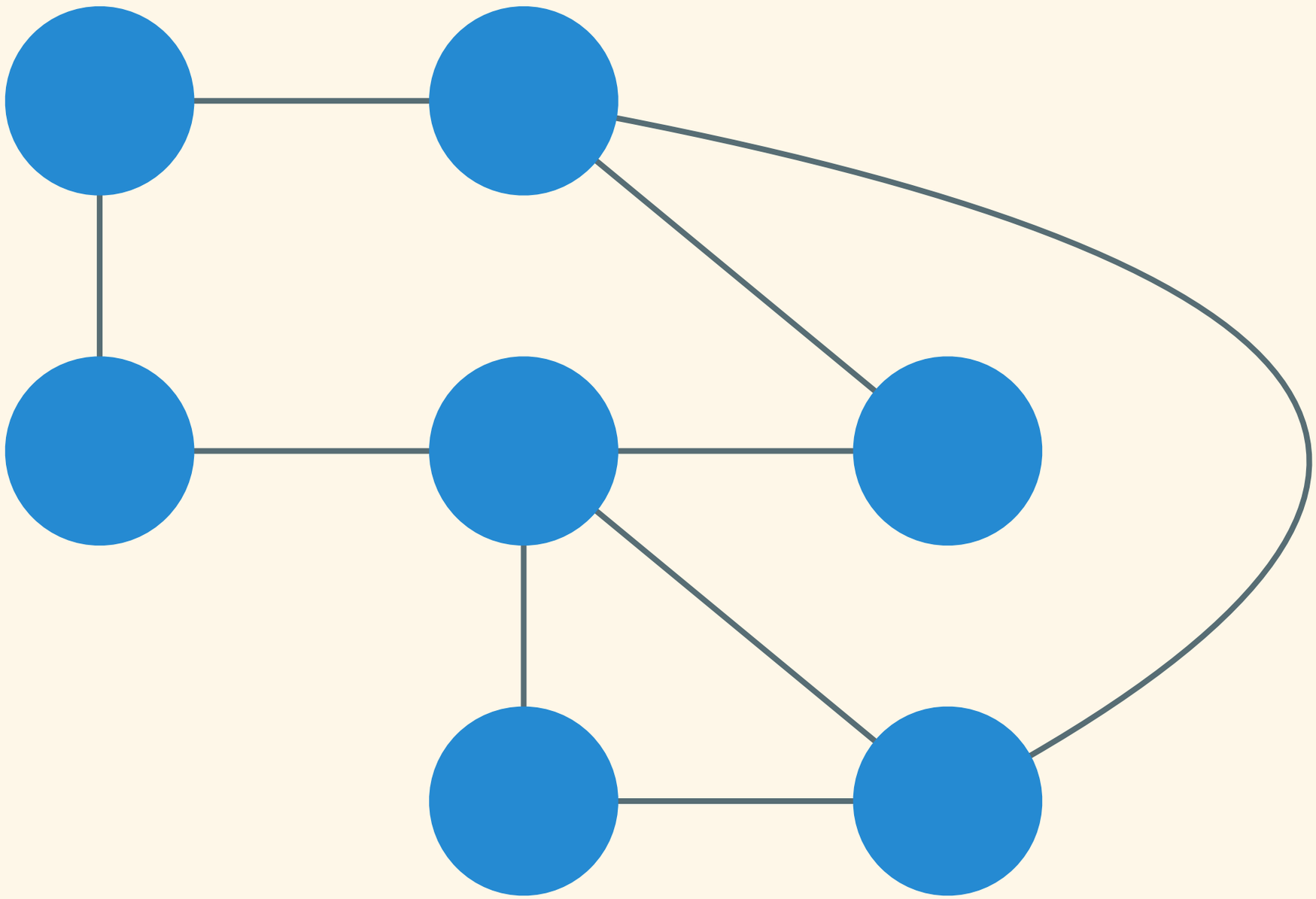


Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)

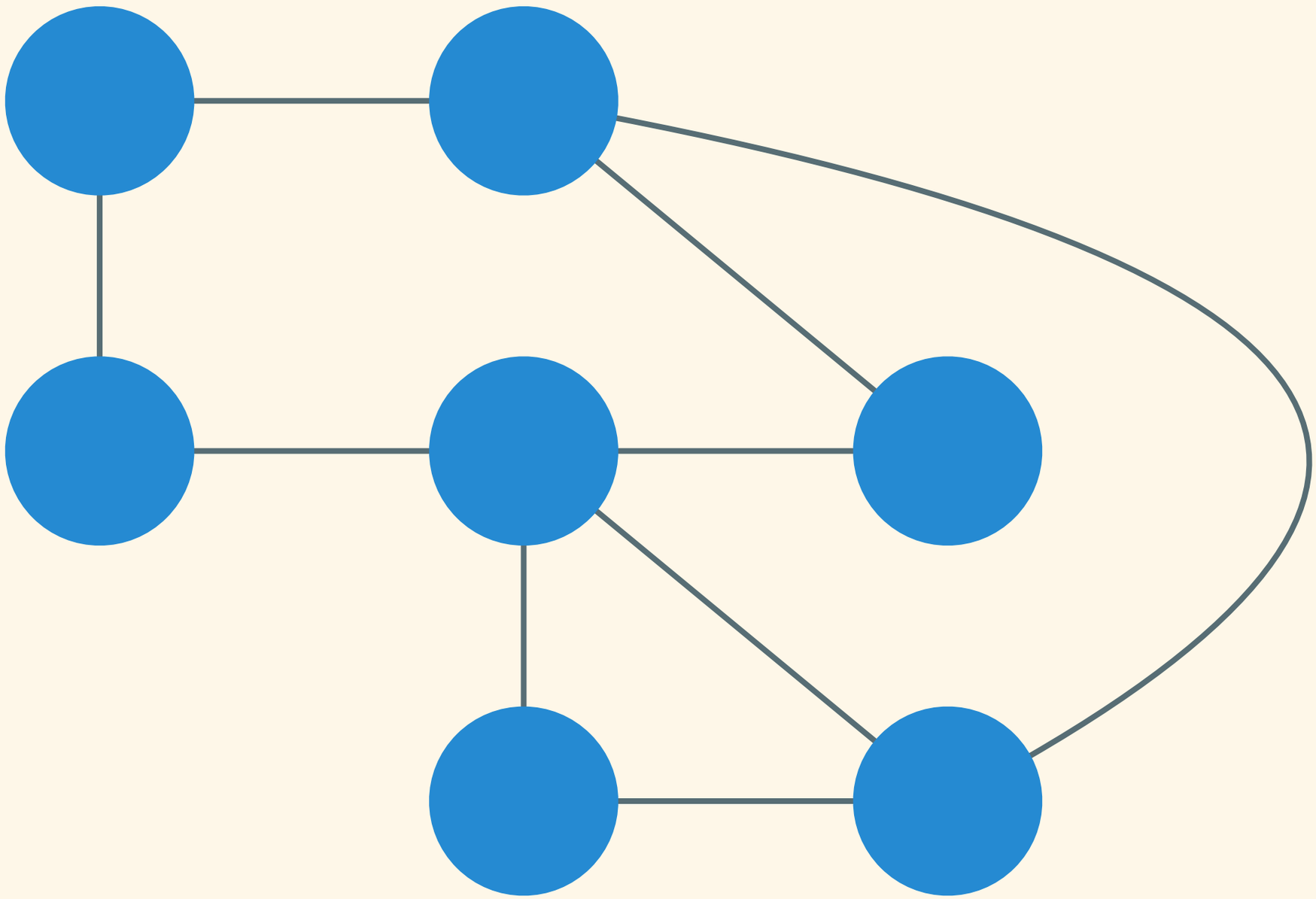
http://creativecommons.org/licenses/by-sa/3.0/

Latest update: Mar 13, 2013

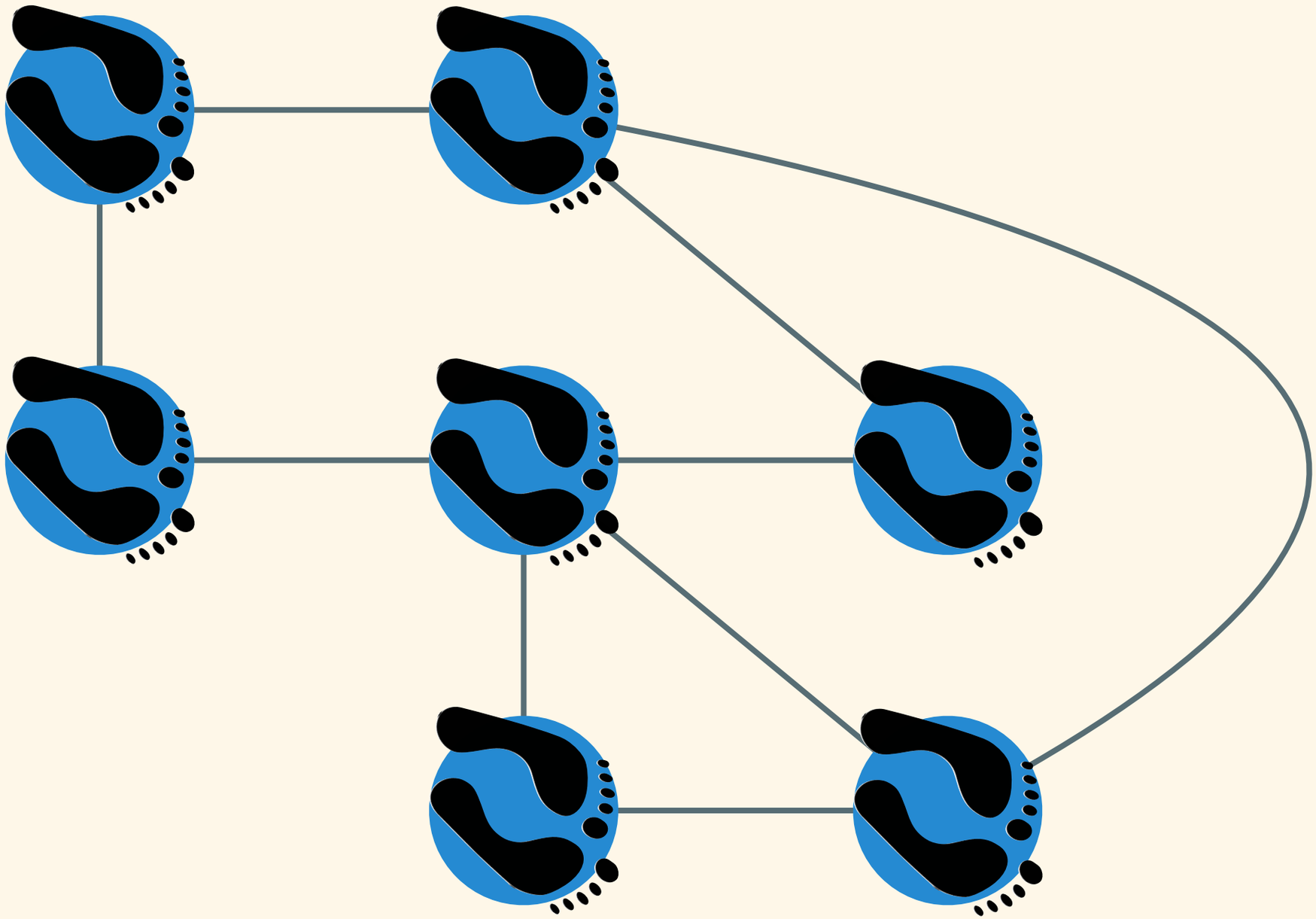
Graph



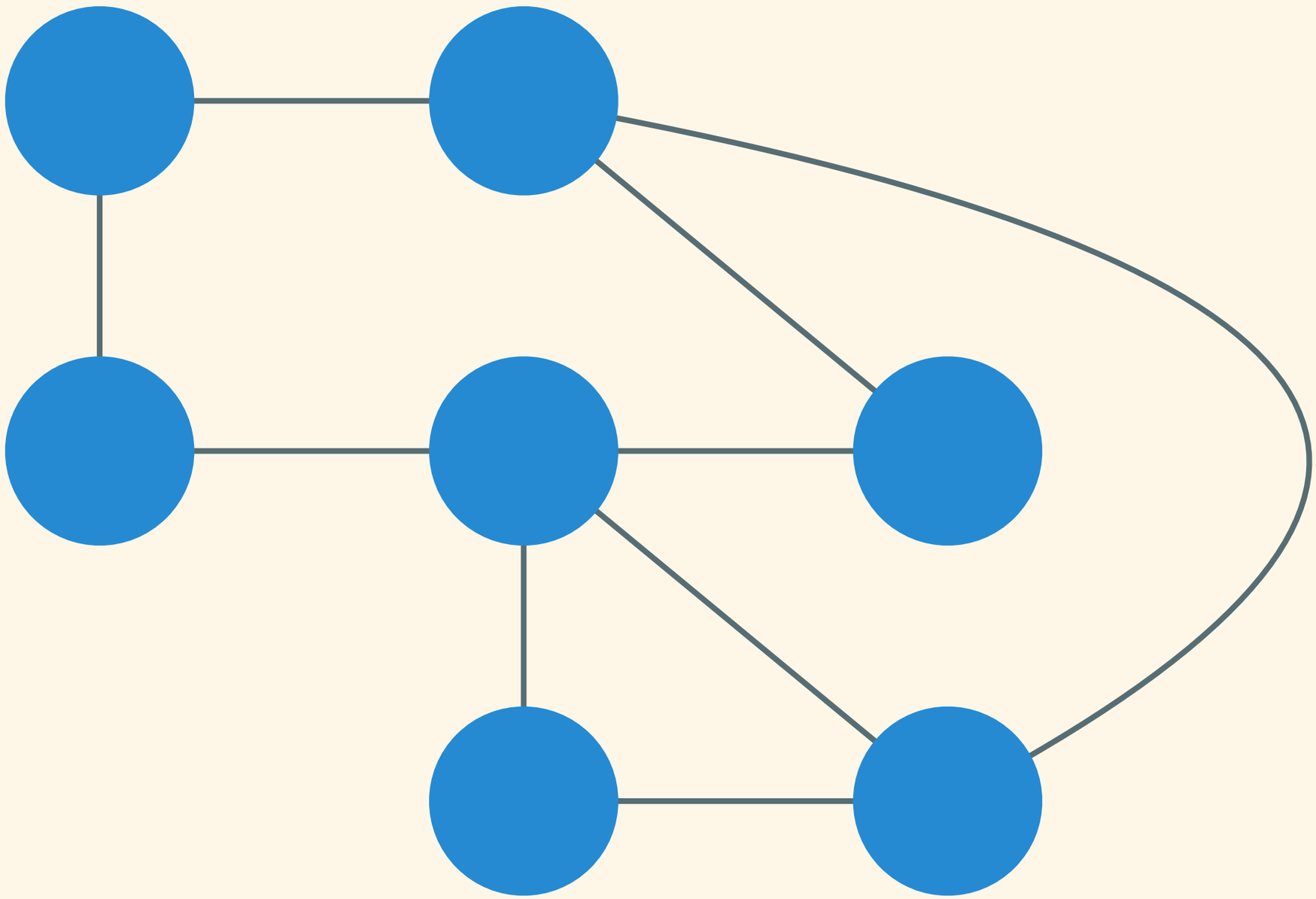
Traversal



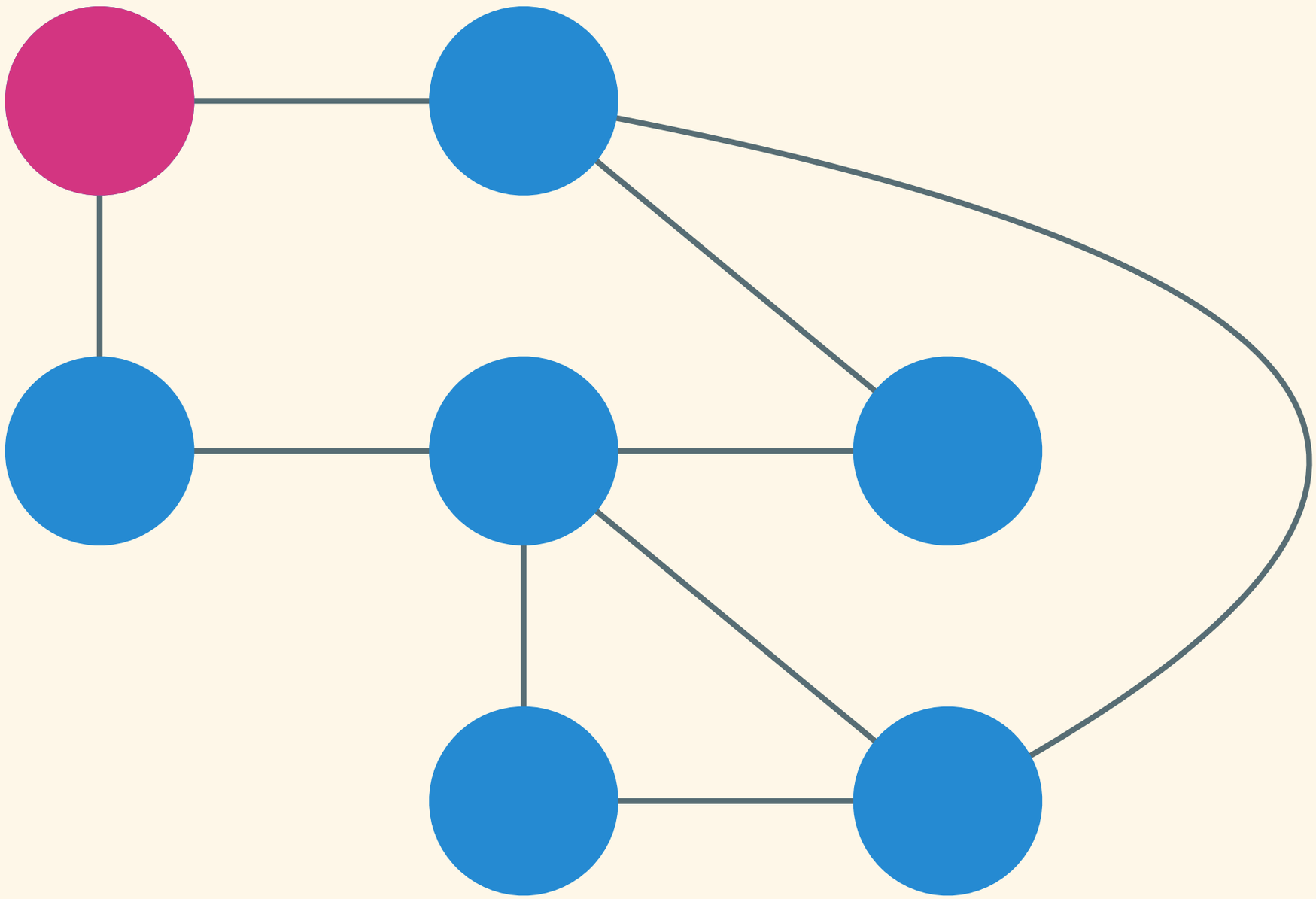
Traversal



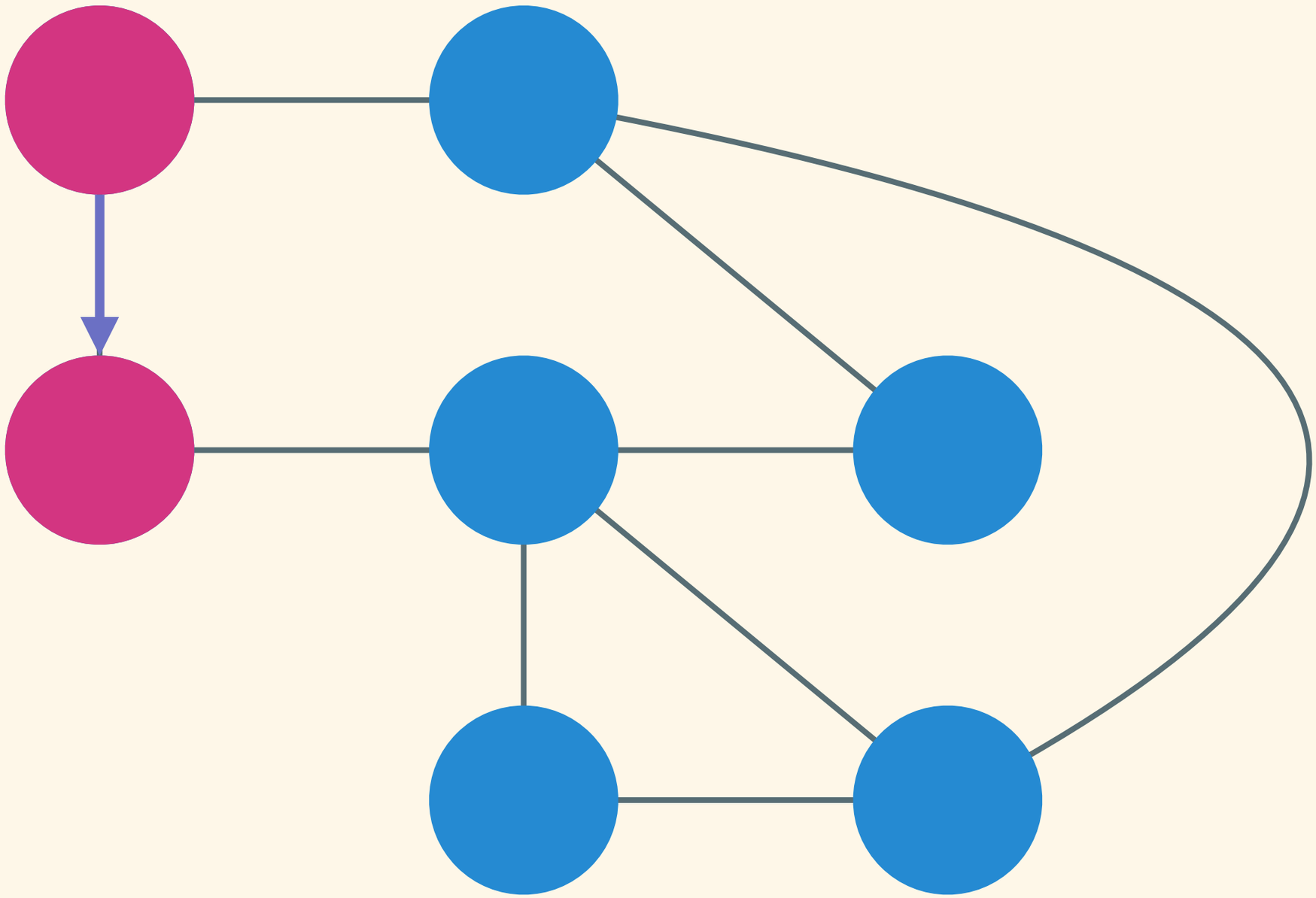
DFS (Depth-First Search)



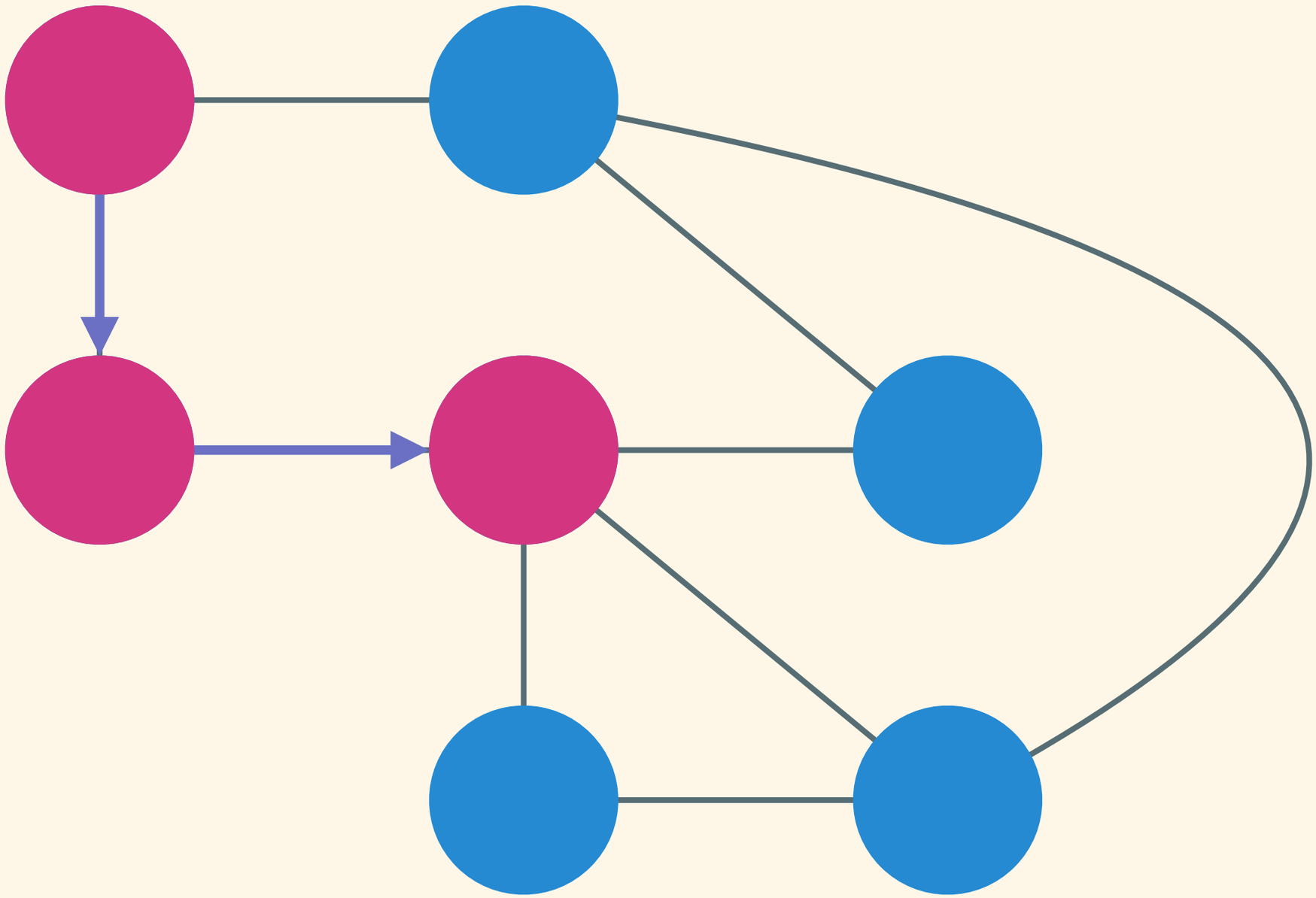
DFS (Depth-First Search)



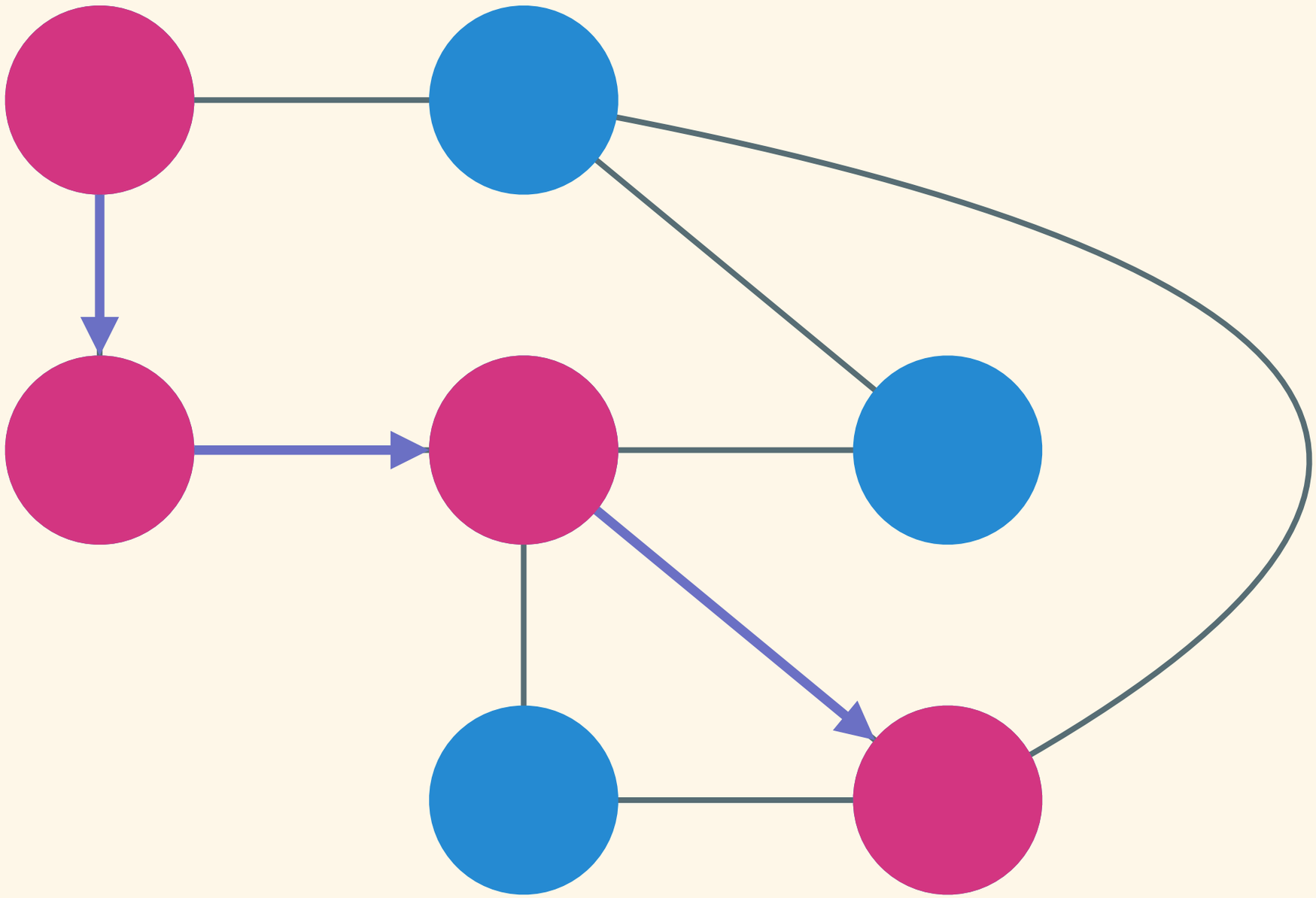
DFS (Depth-First Search)



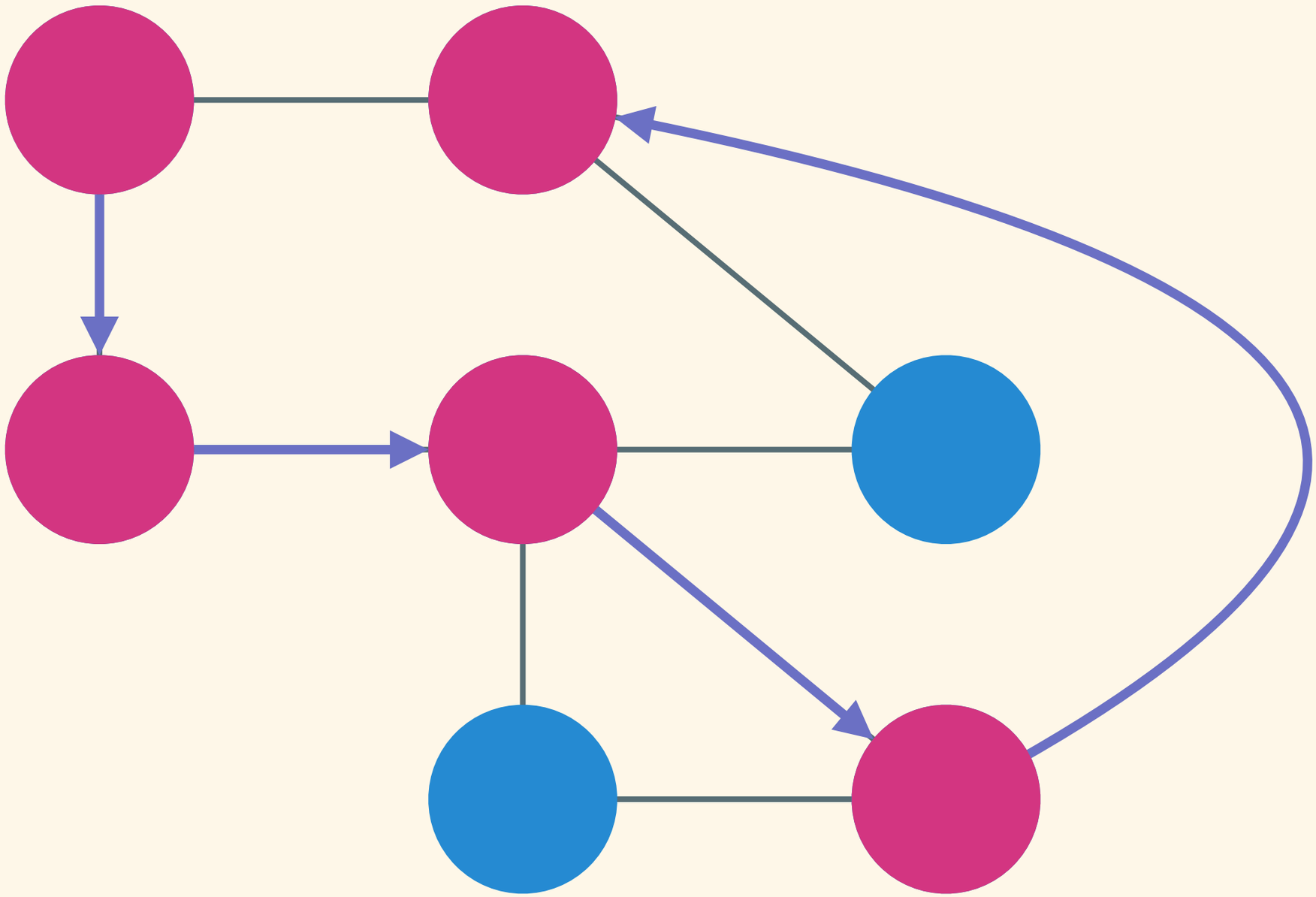
DFS (Depth-First Search)



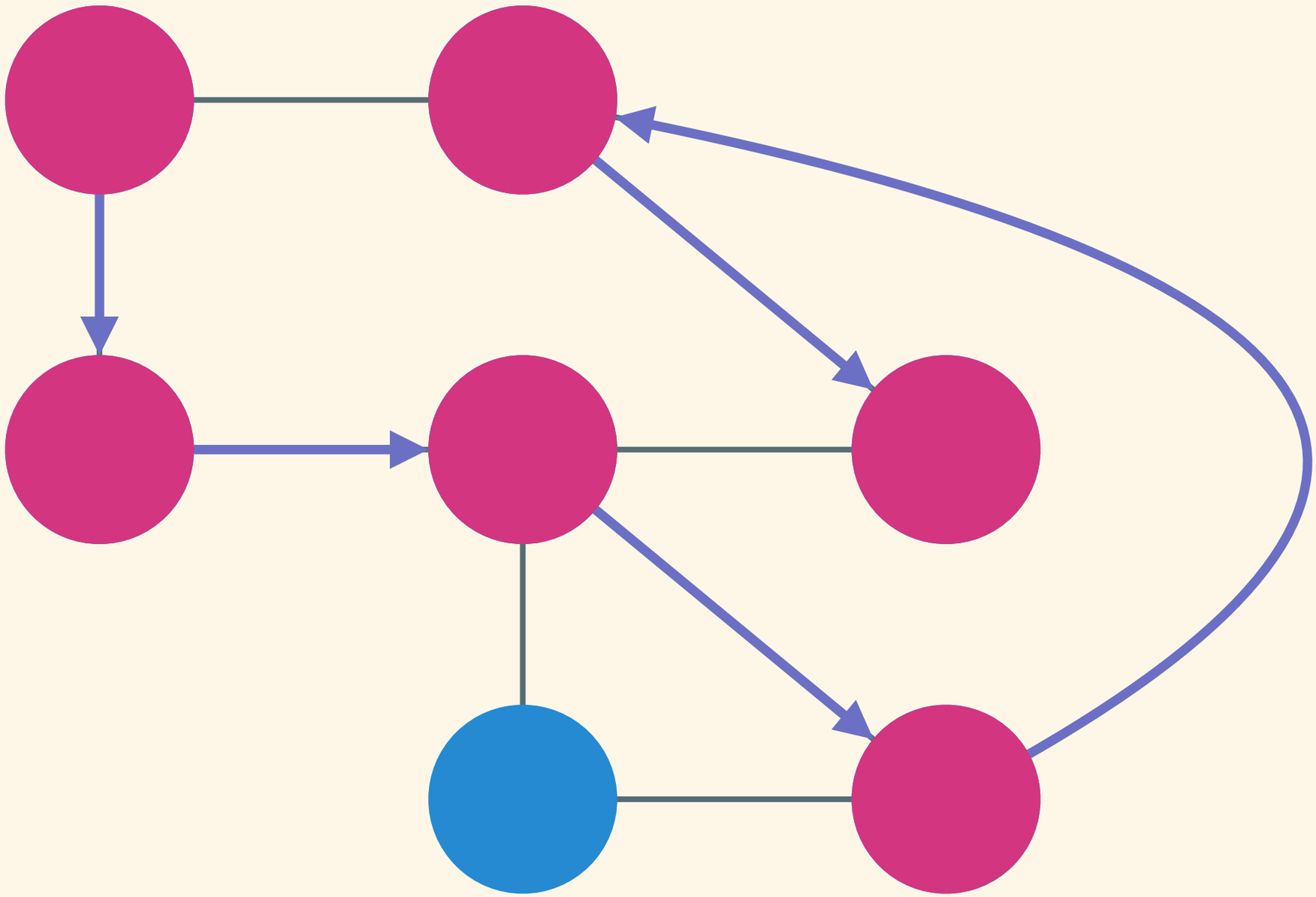
DFS (Depth-First Search)



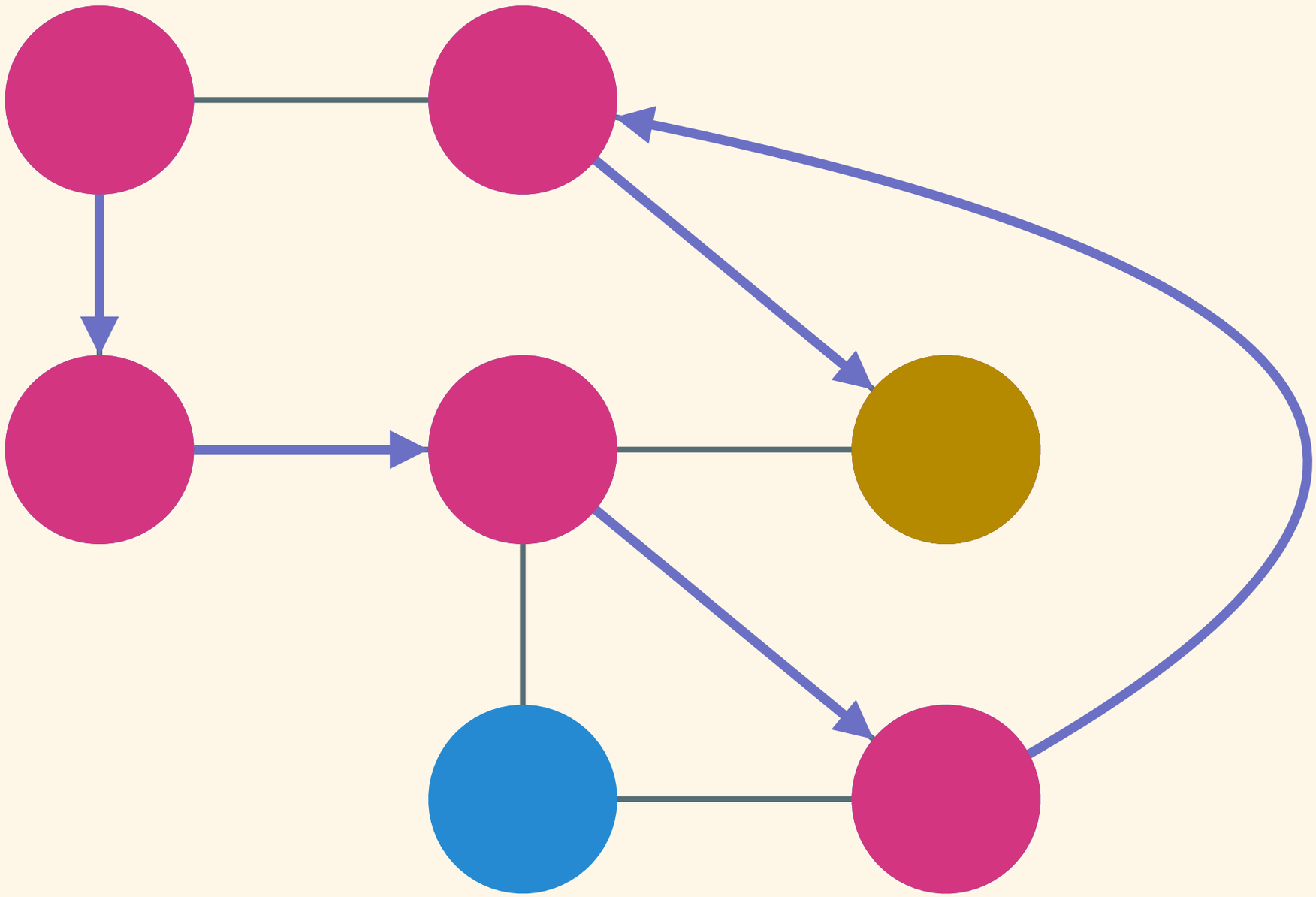
DFS (Depth-First Search)



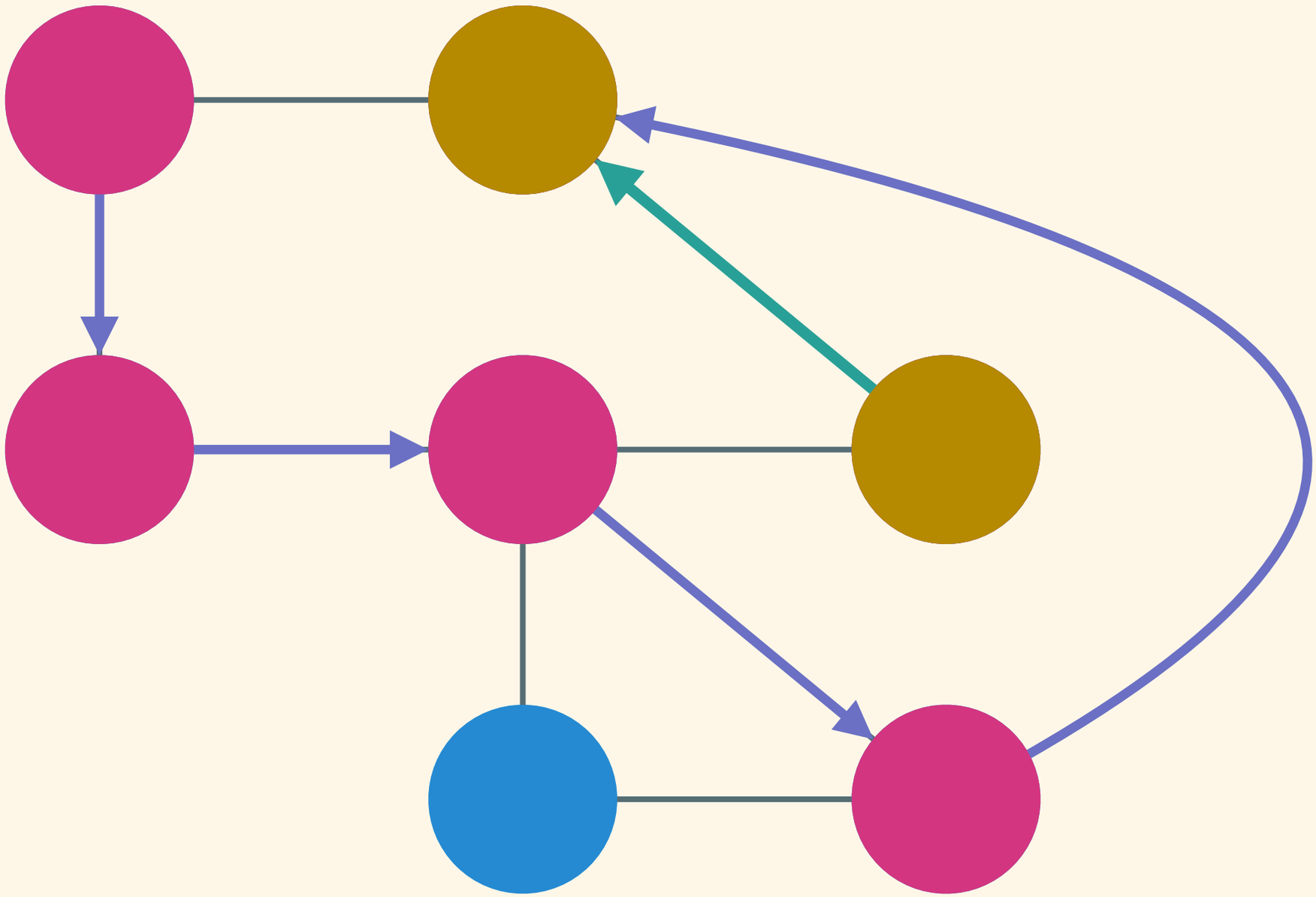
DFS (Depth-First Search)



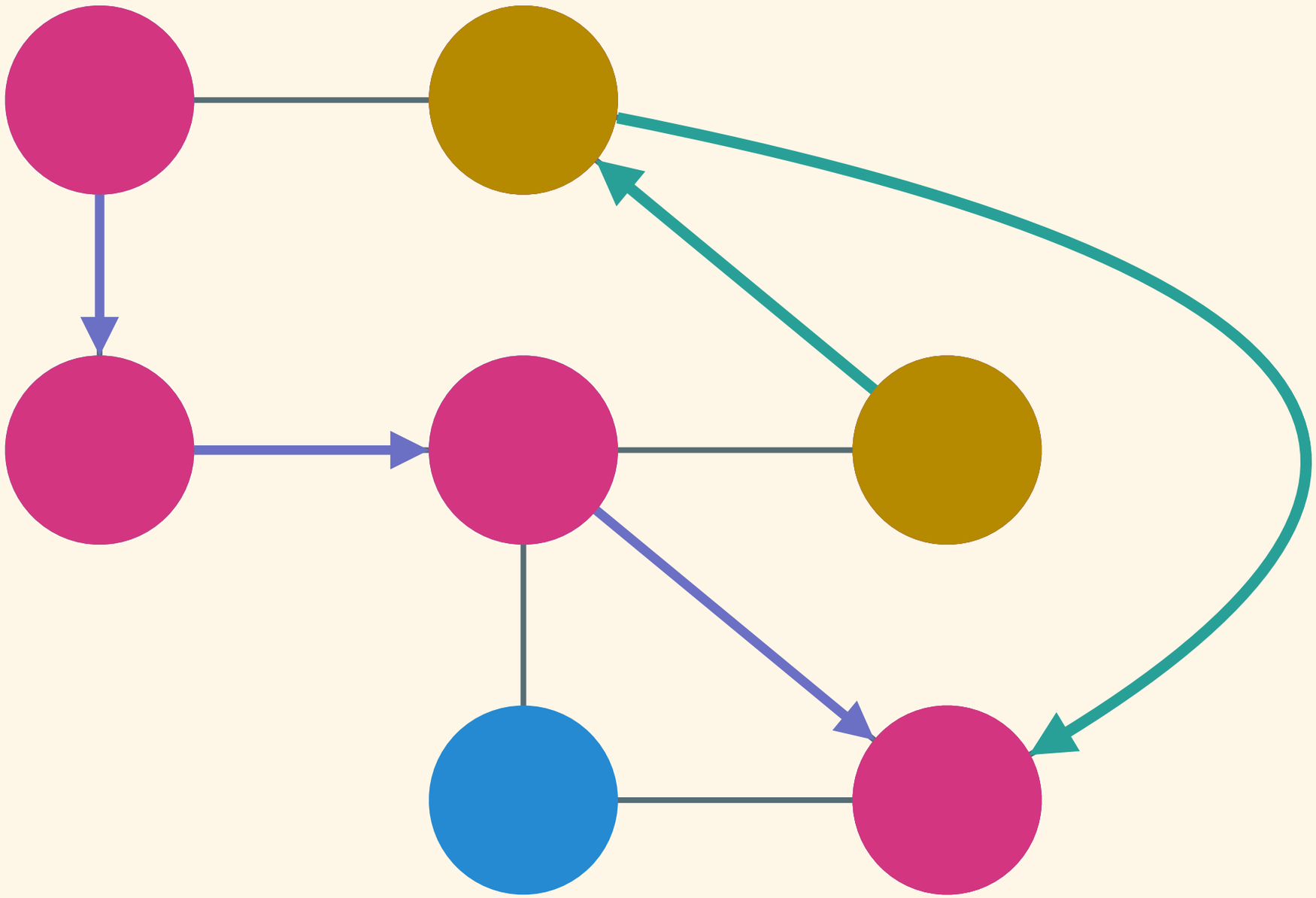
DFS (Depth-First Search)



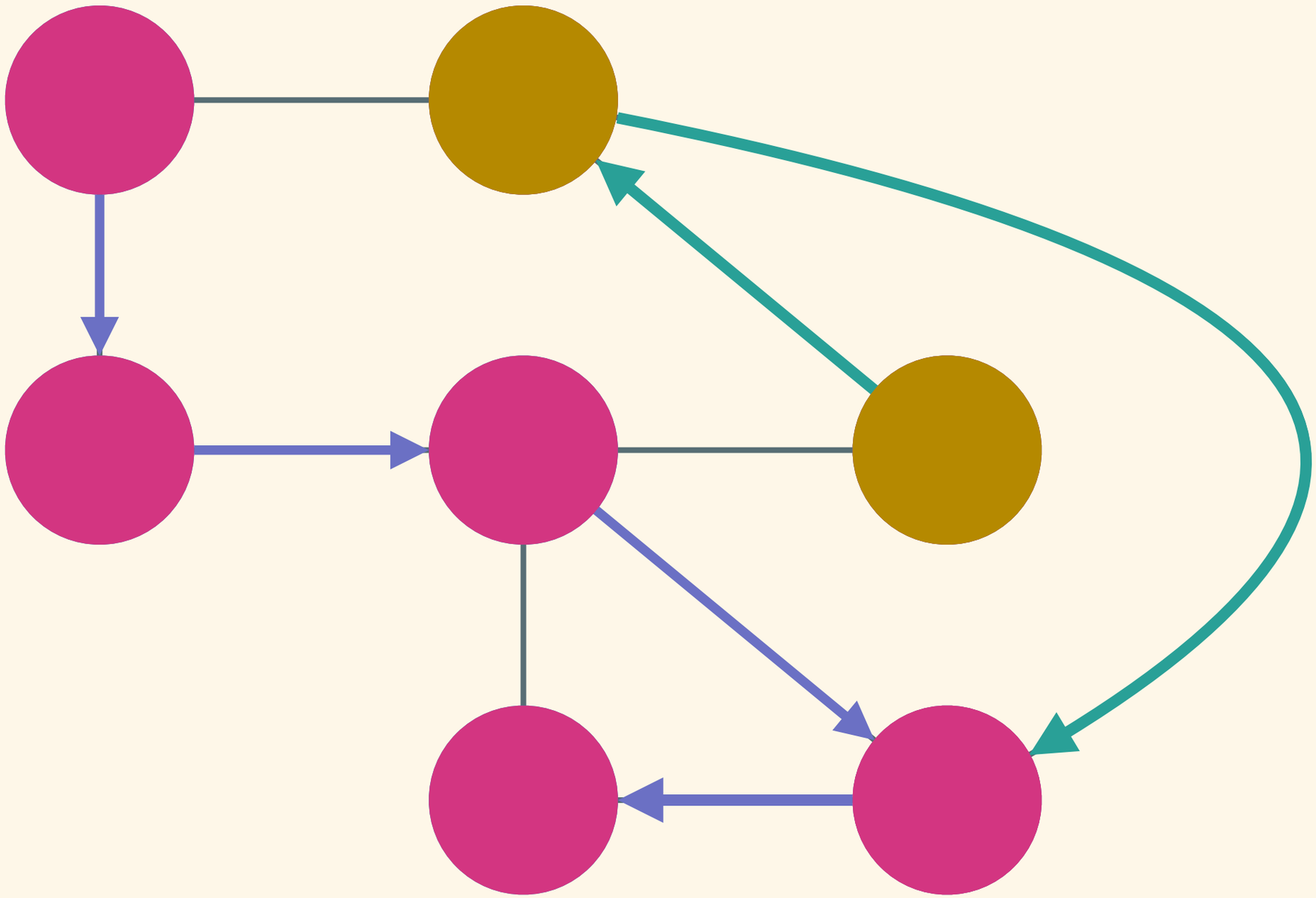
DFS (Depth-First Search)



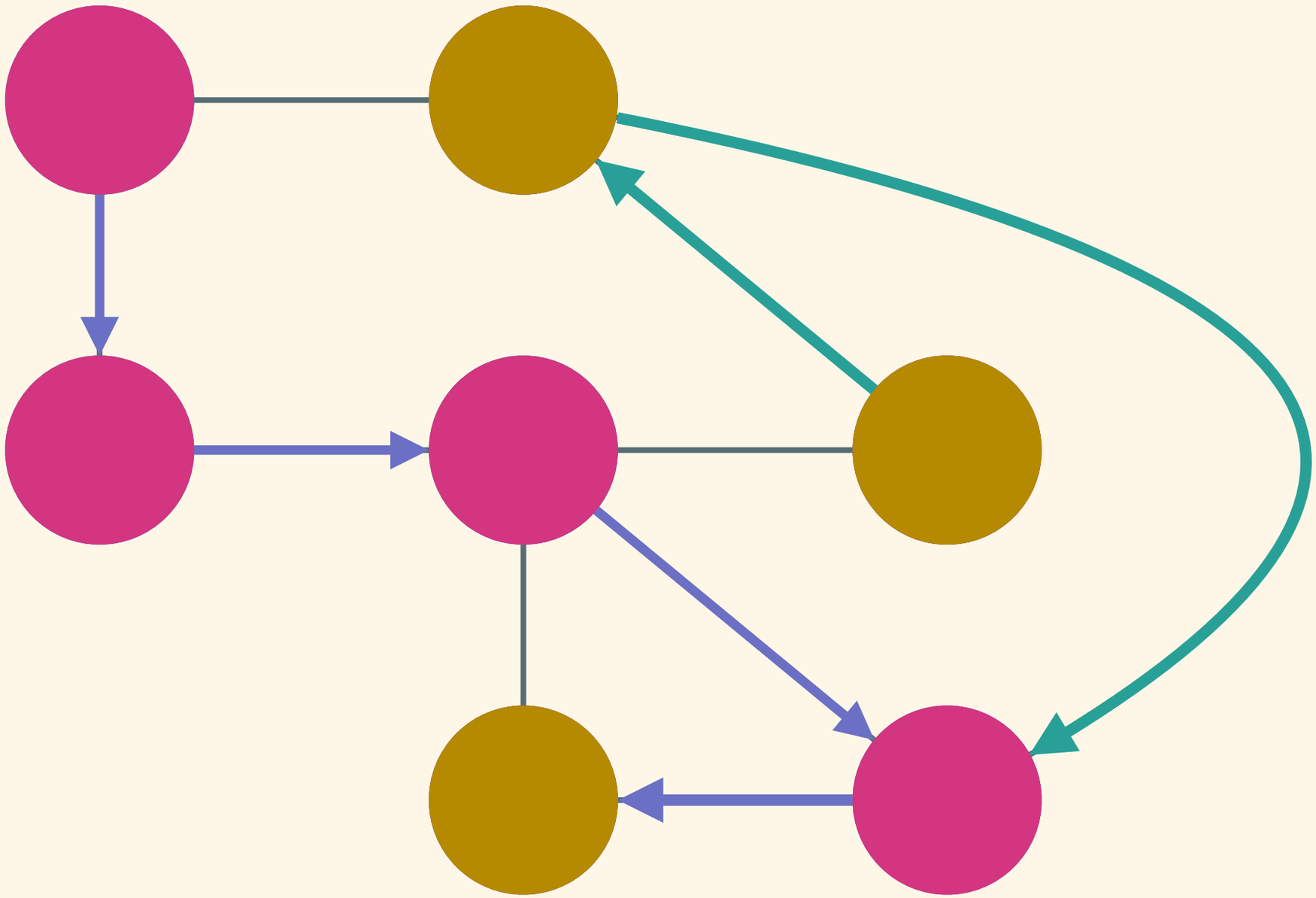
DFS (Depth-First Search)



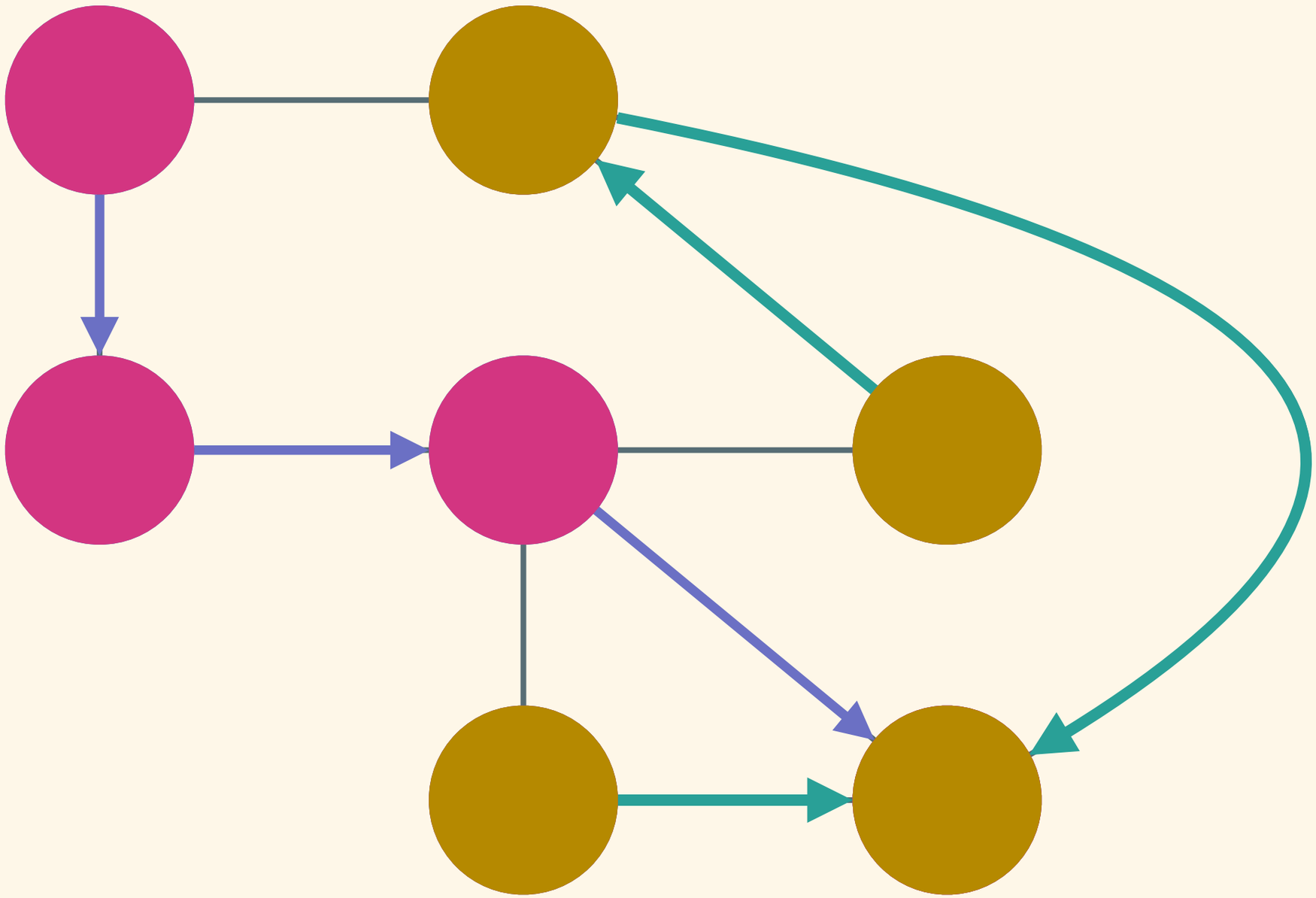
DFS (Depth-First Search)



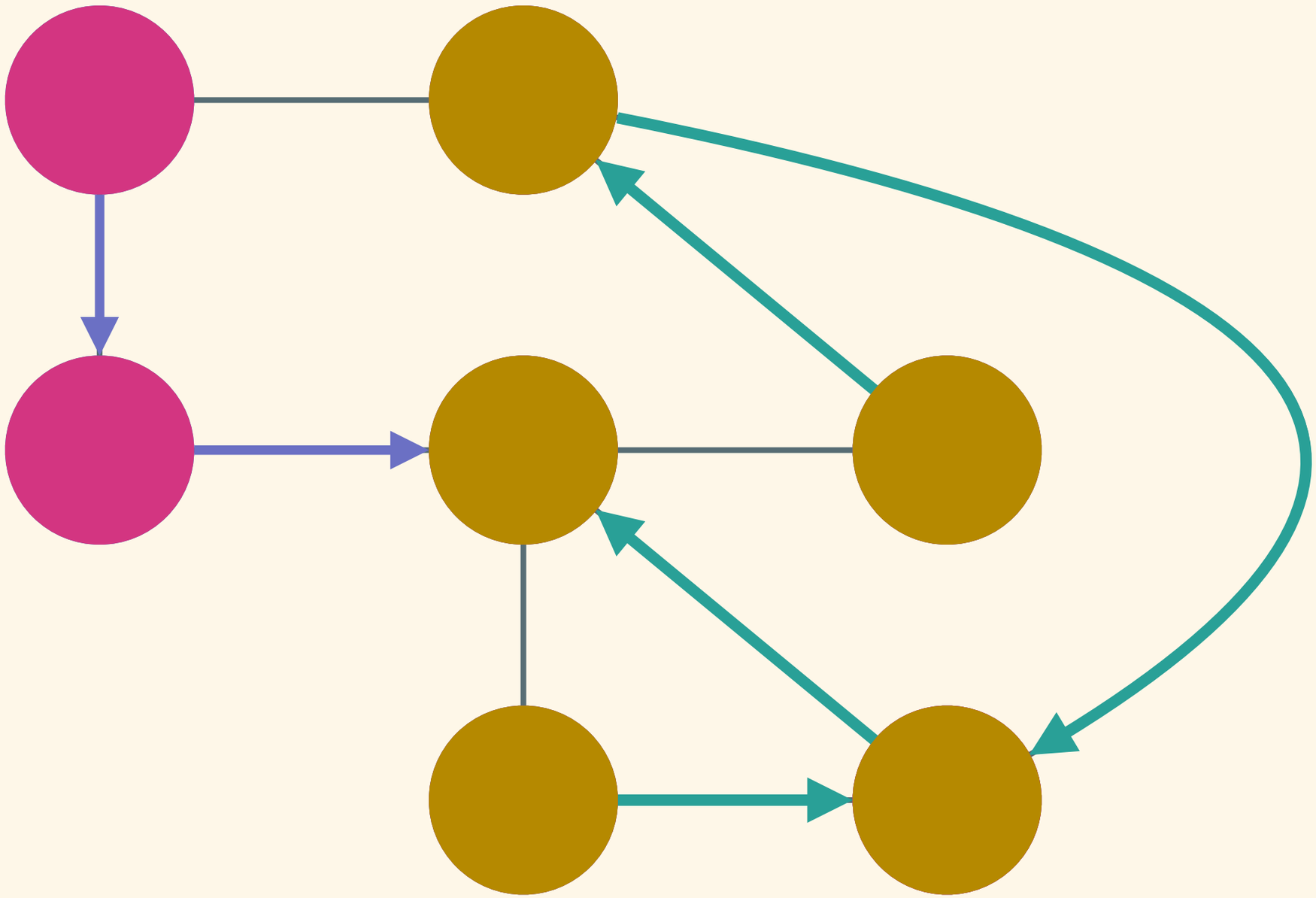
DFS (Depth-First Search)



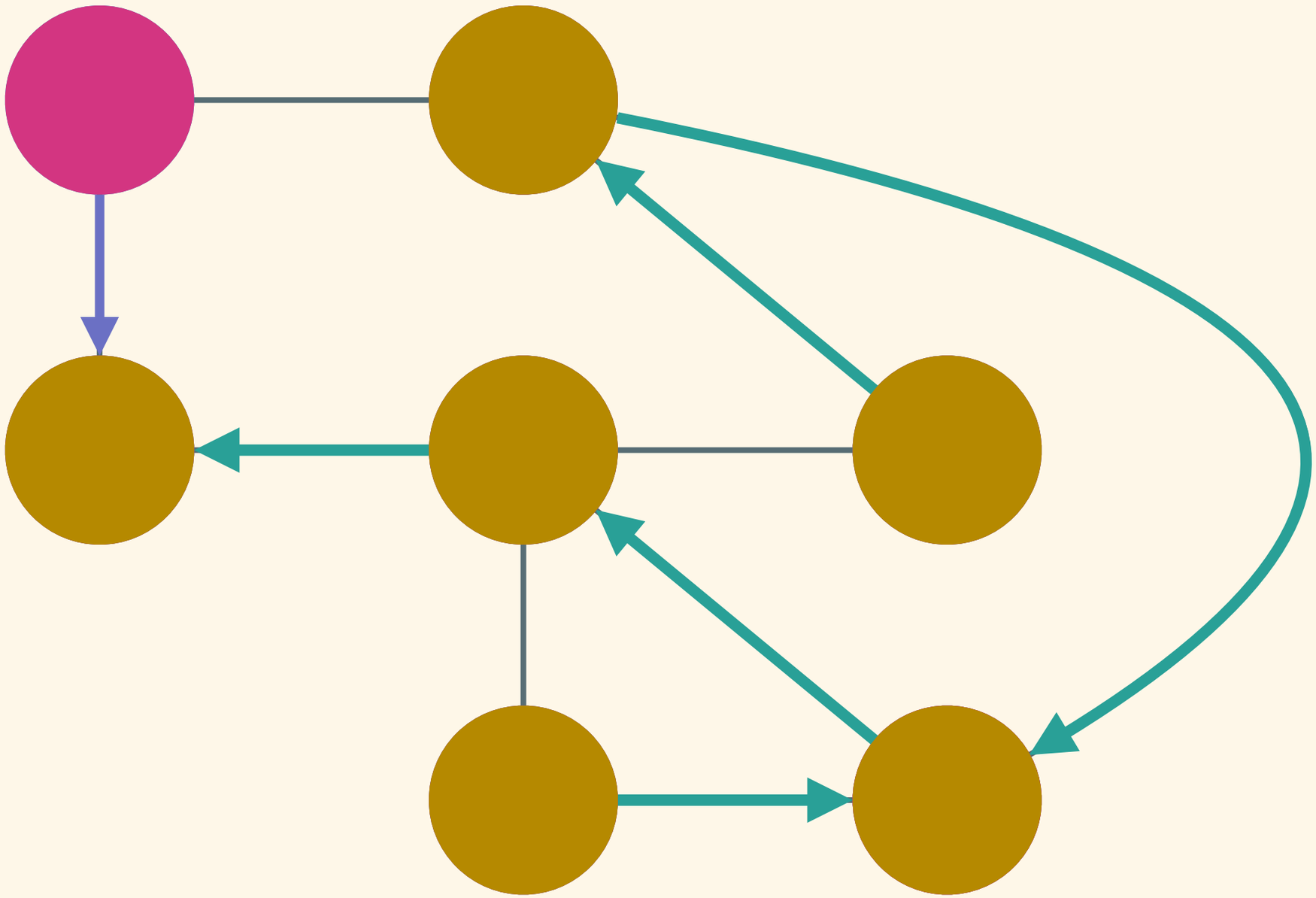
DFS (Depth-First Search)



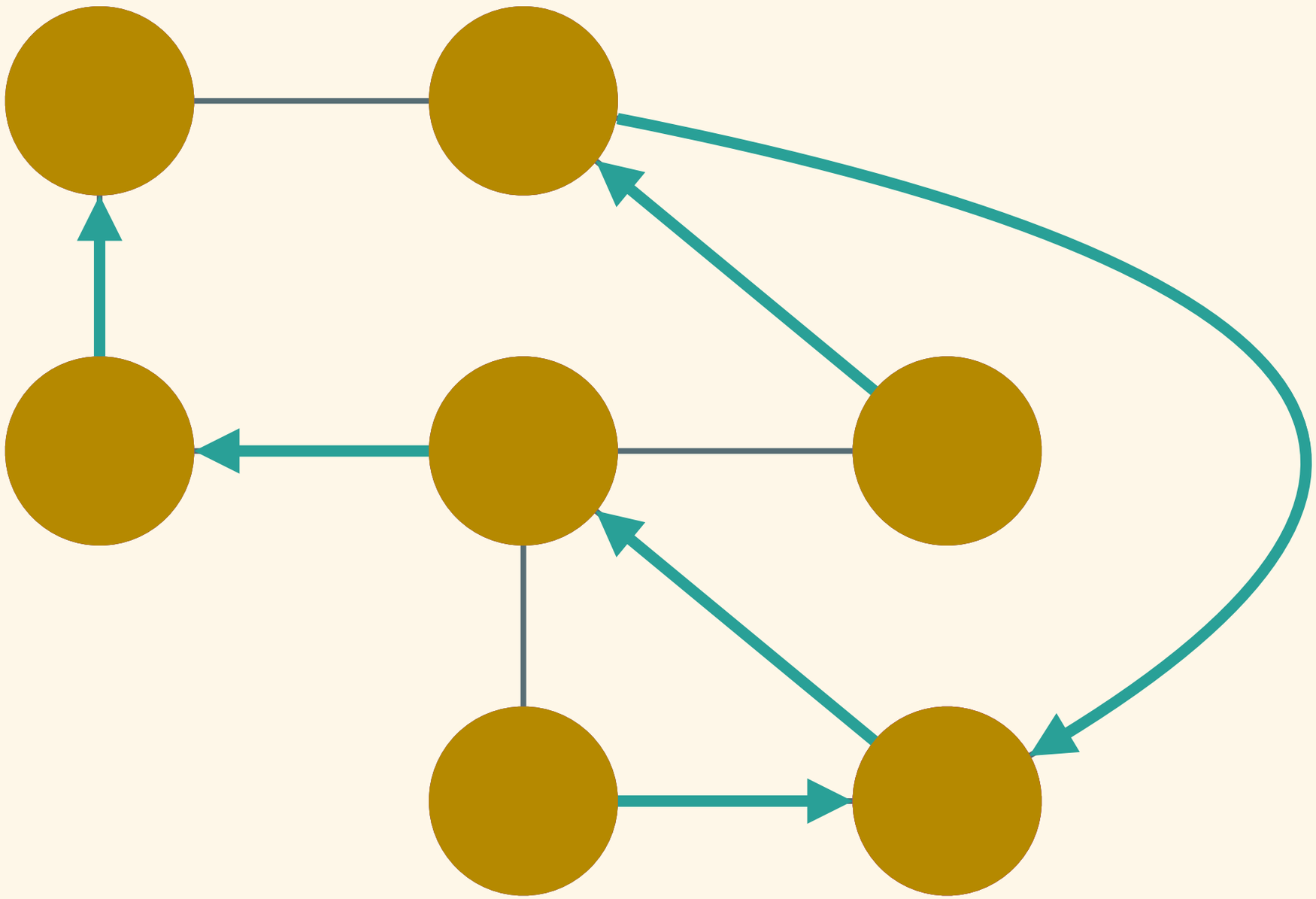
DFS (Depth-First Search)



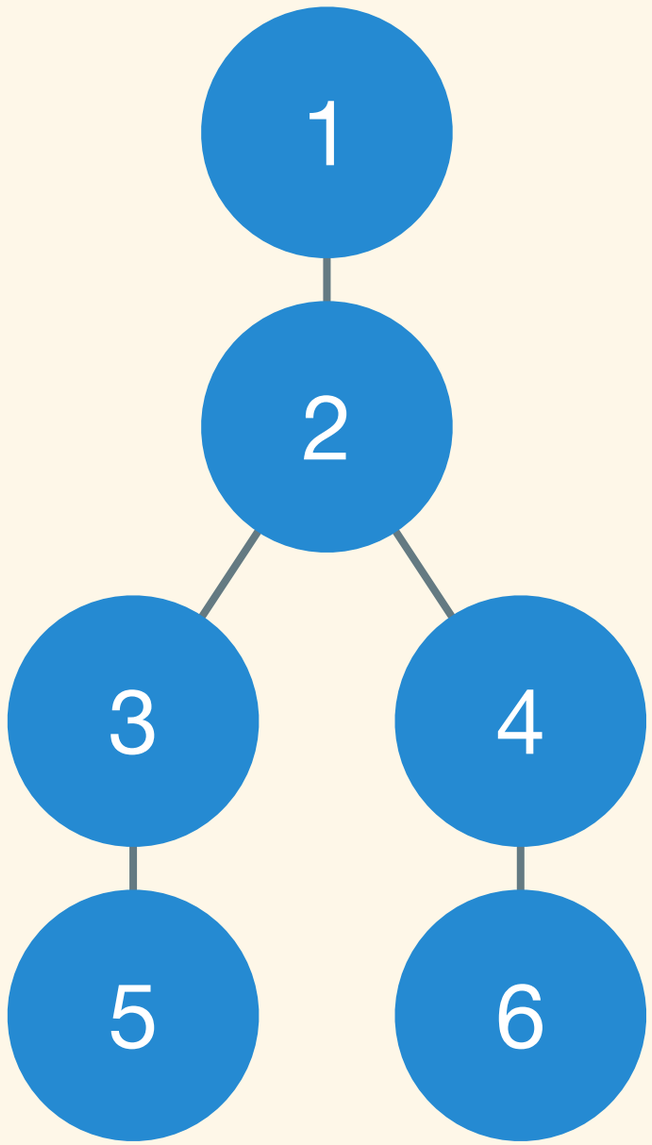
DFS (Depth-First Search)

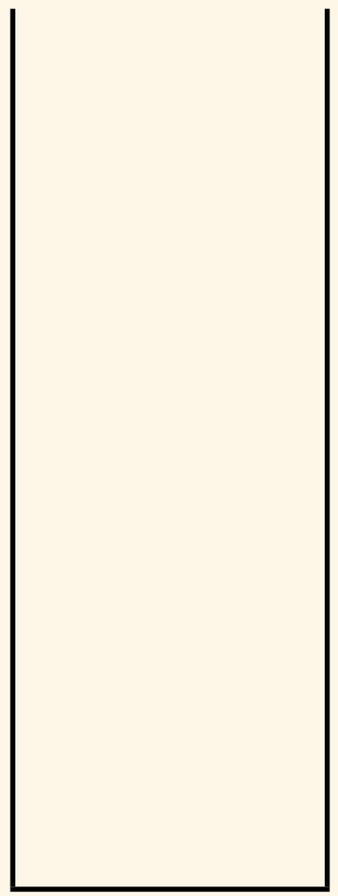


DFS (Depth-First Search)

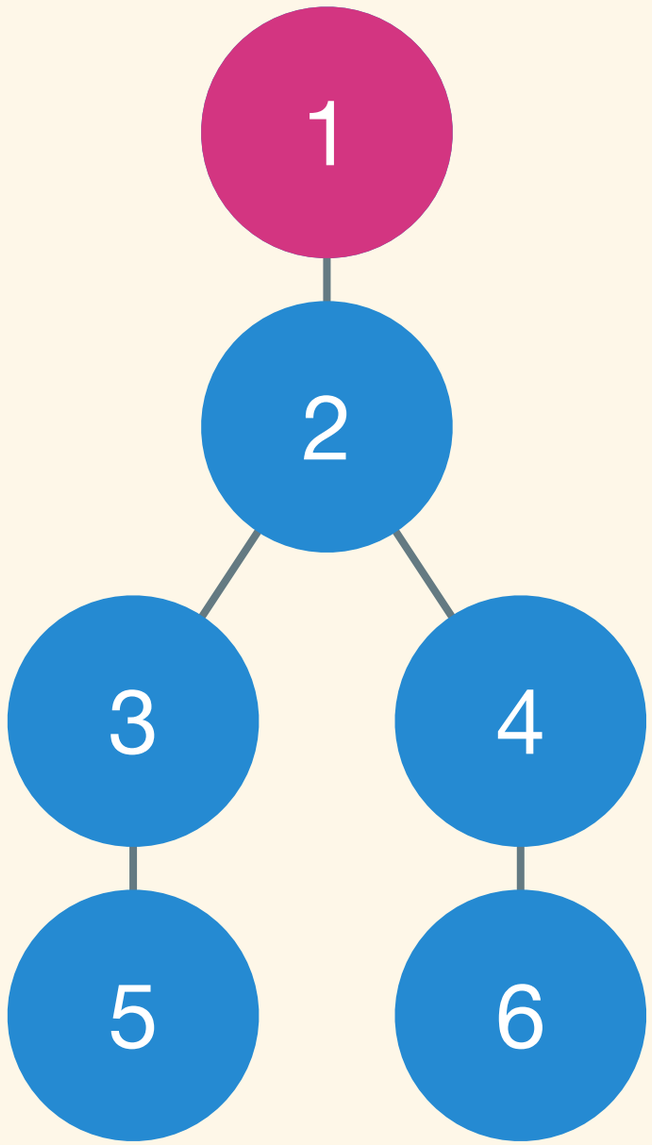


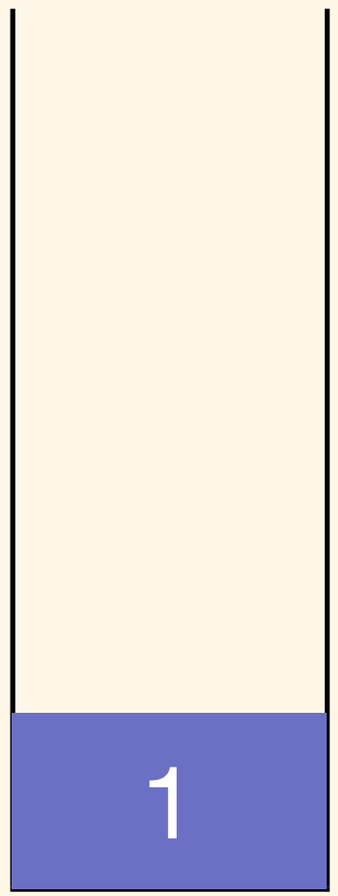
DFS & stack



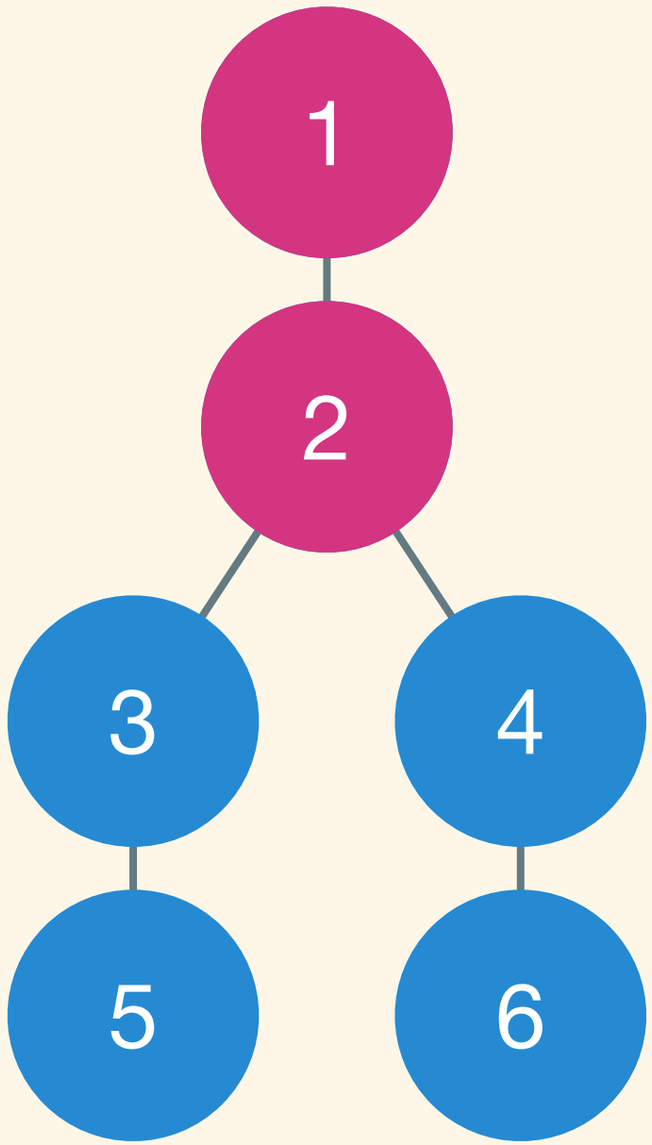


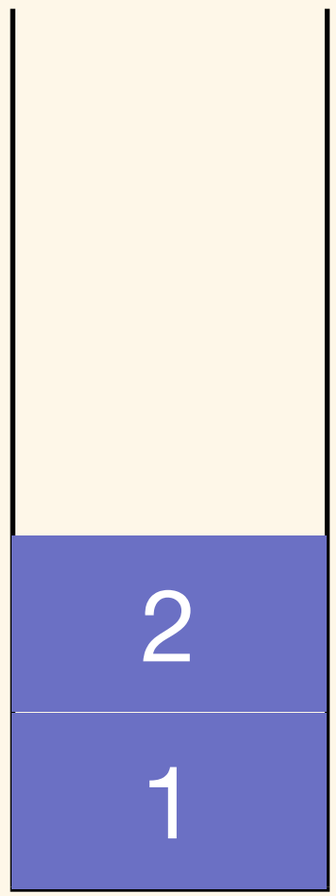
DFS & stack



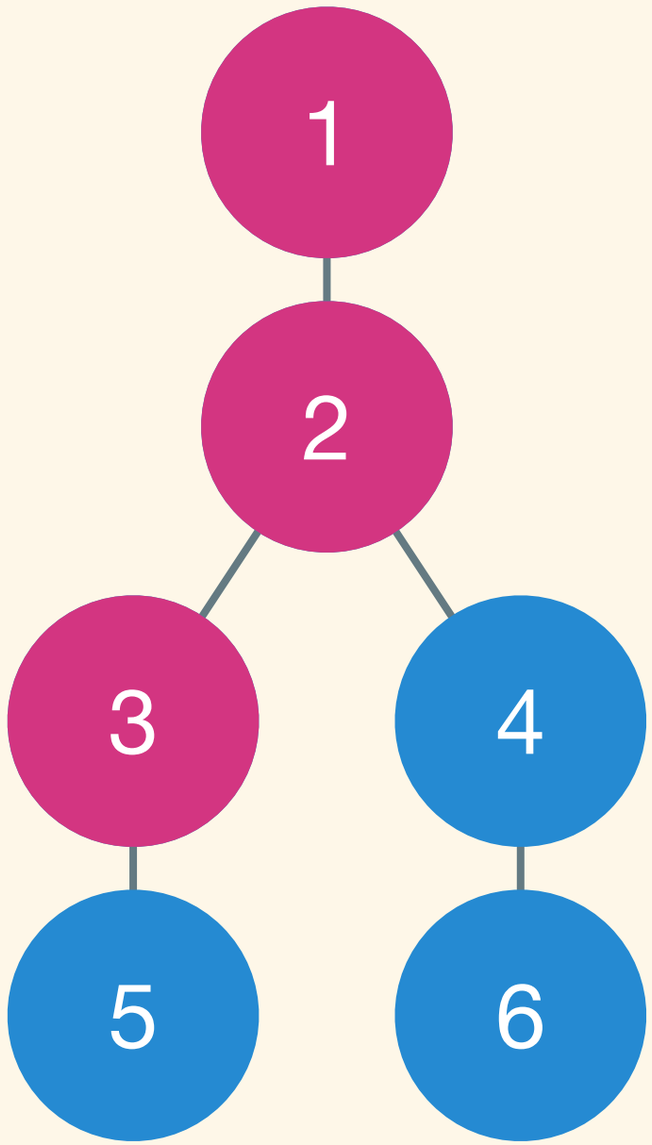


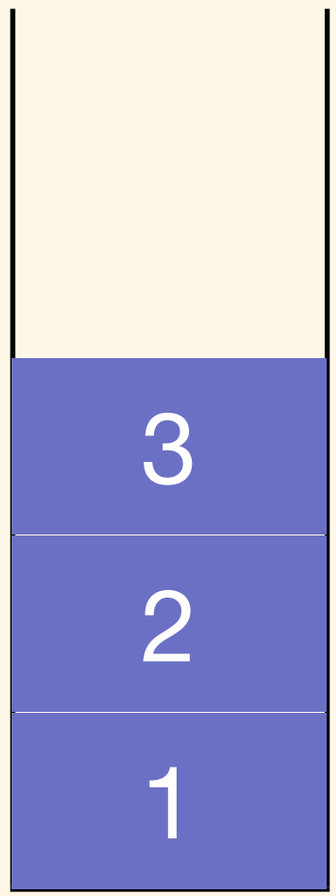
DFS & stack



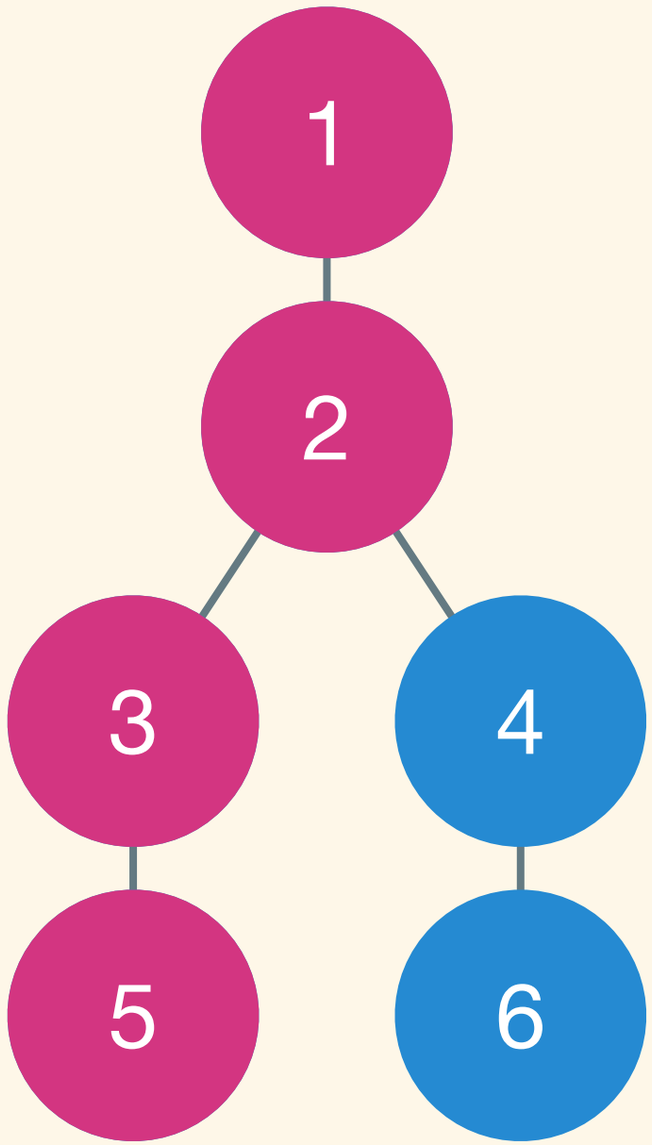


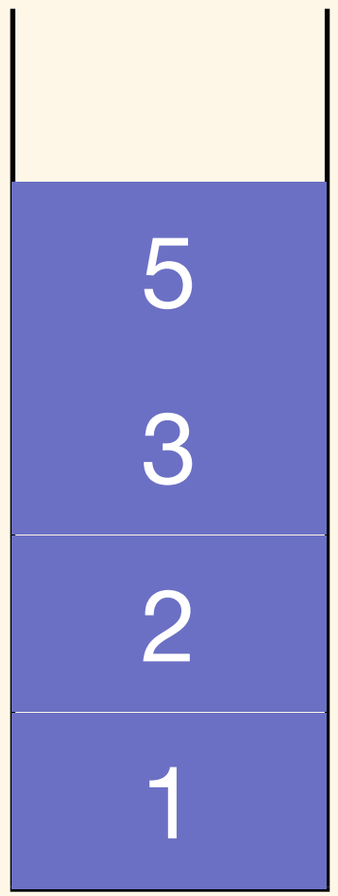
DFS & stack



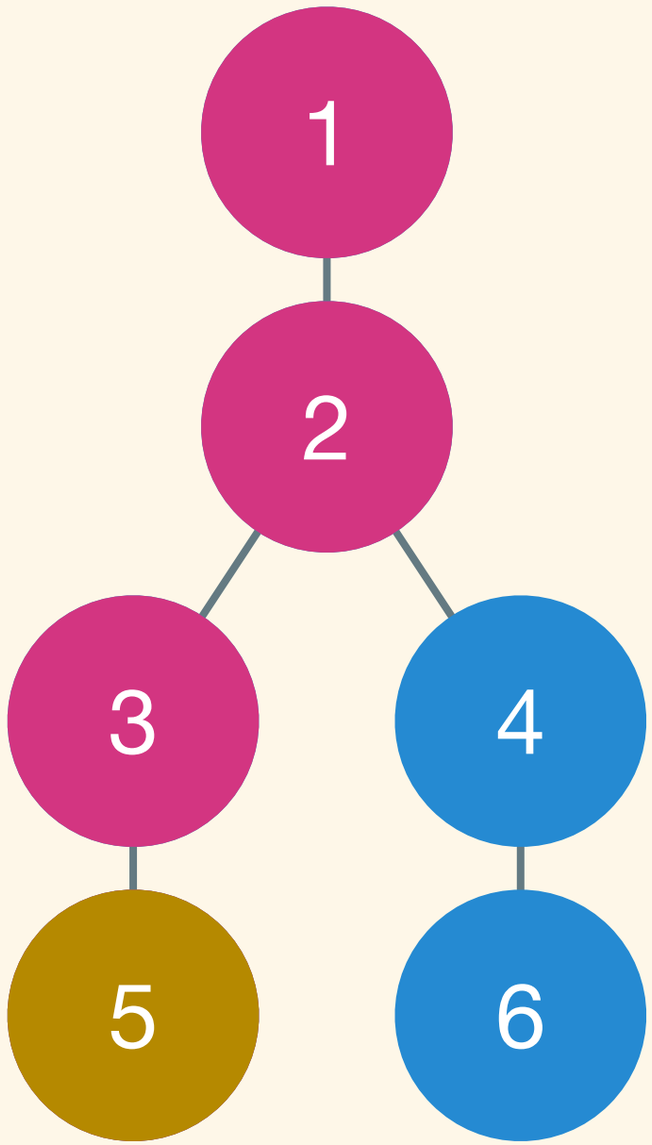


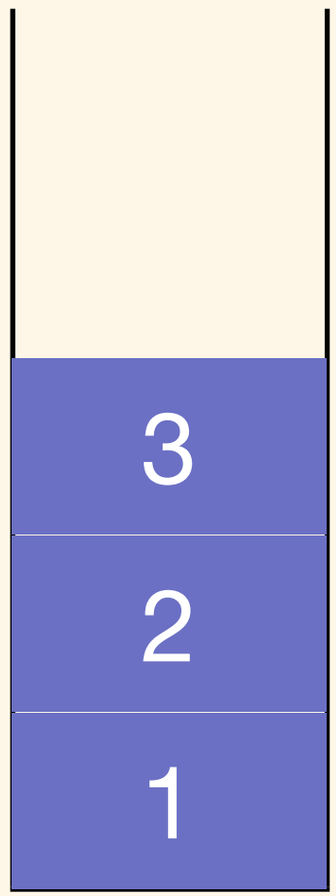
DFS & stack



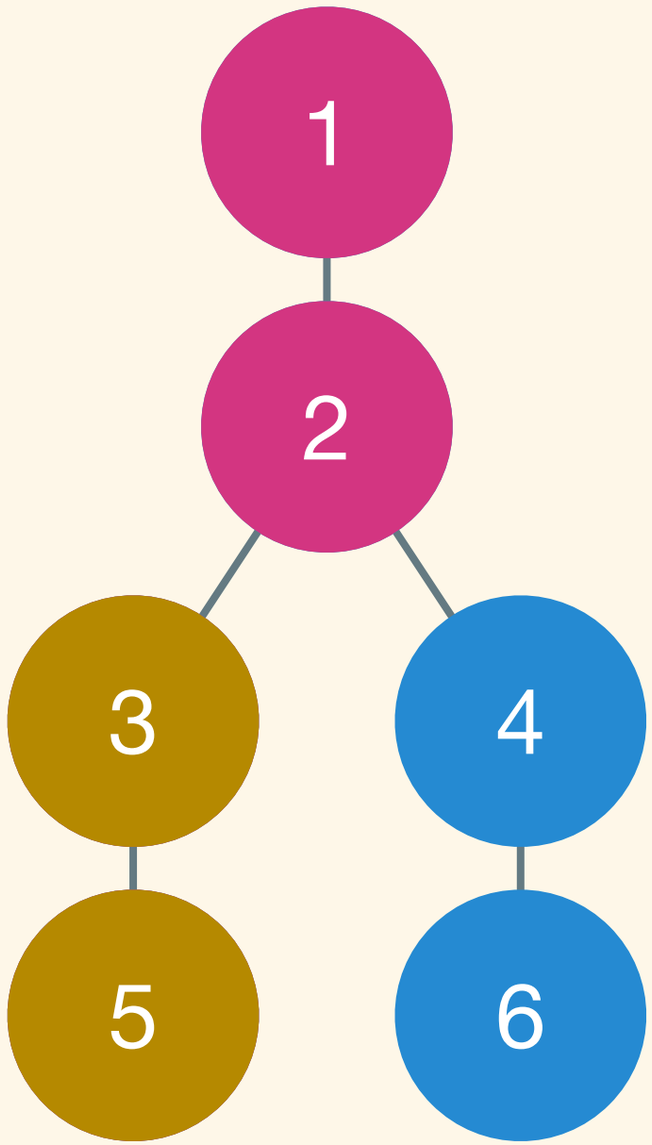


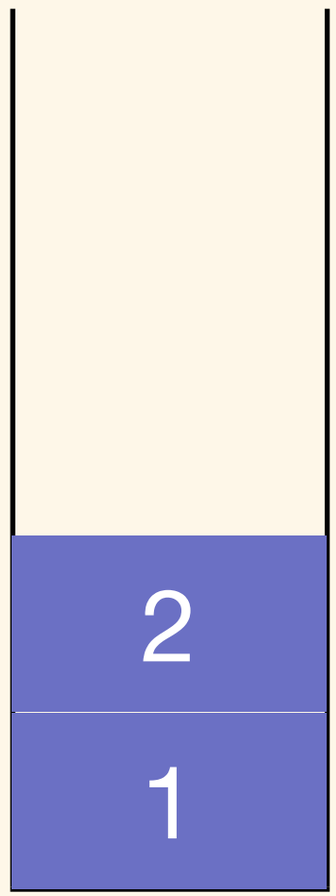
DFS & stack



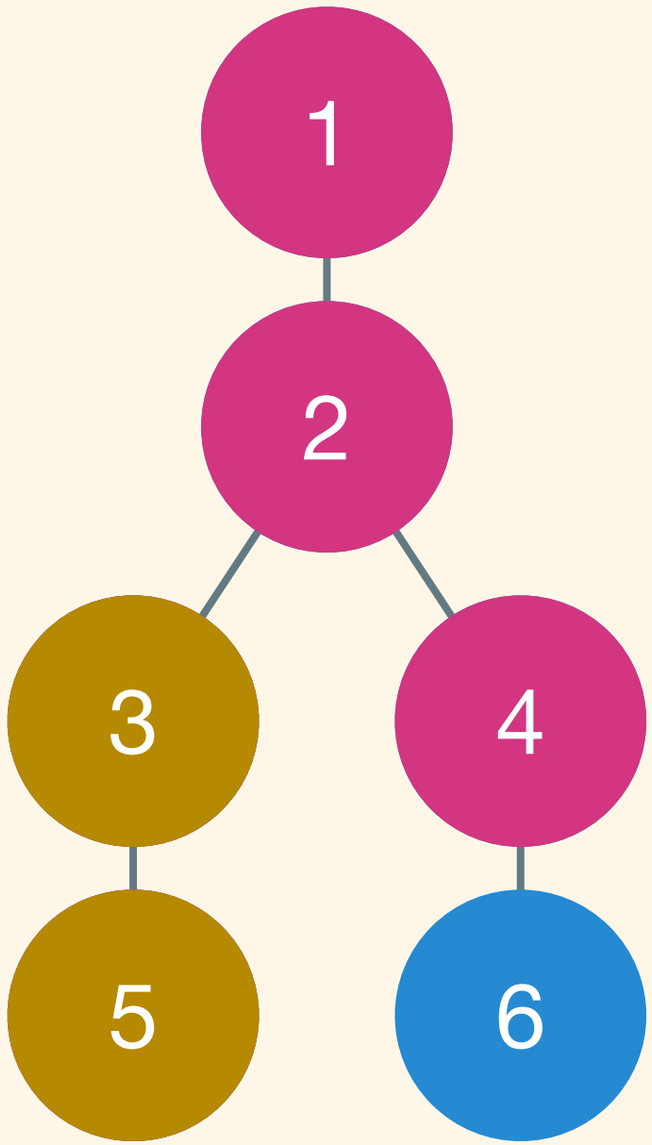


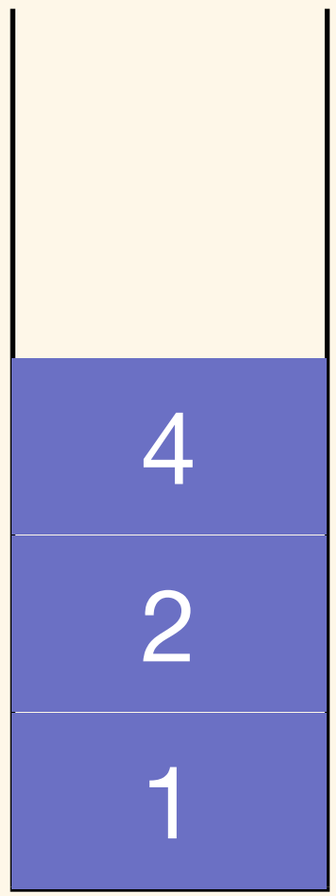
DFS & stack



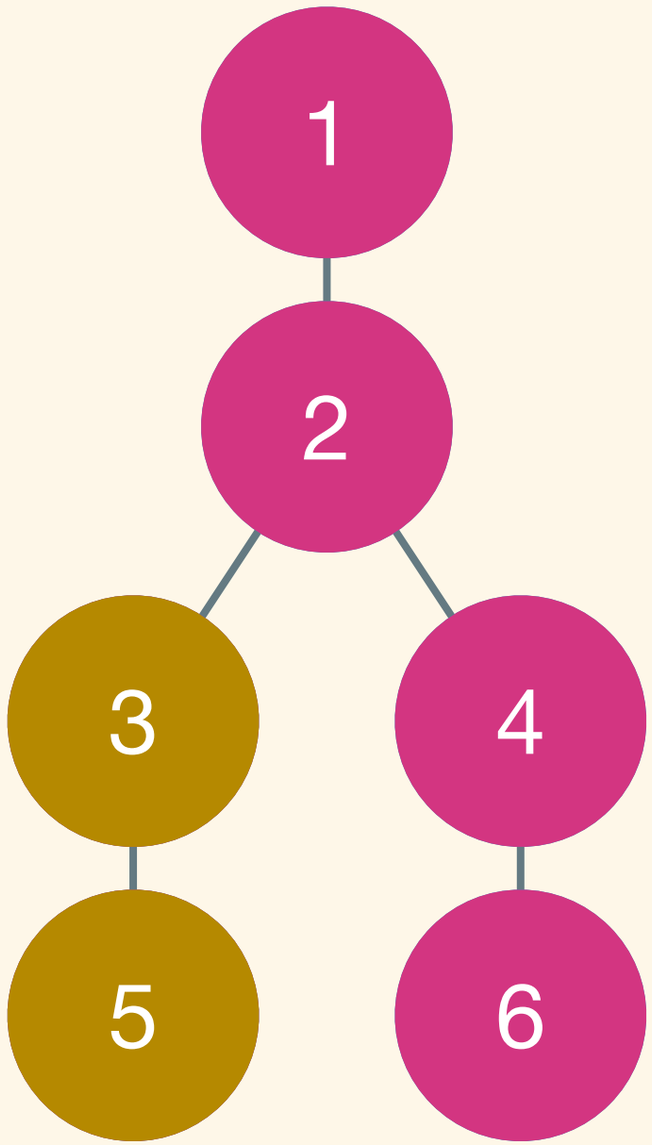


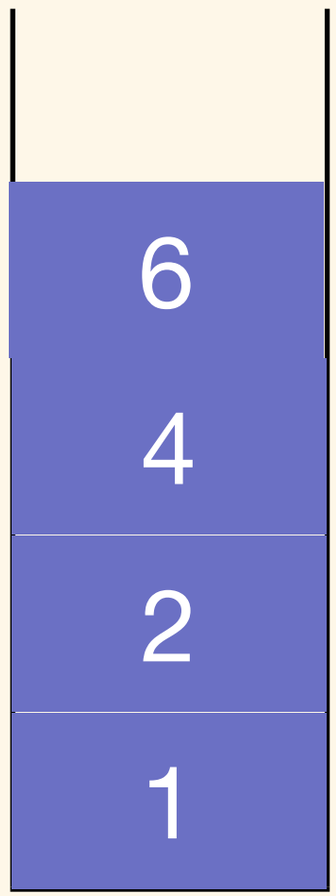
DFS & stack



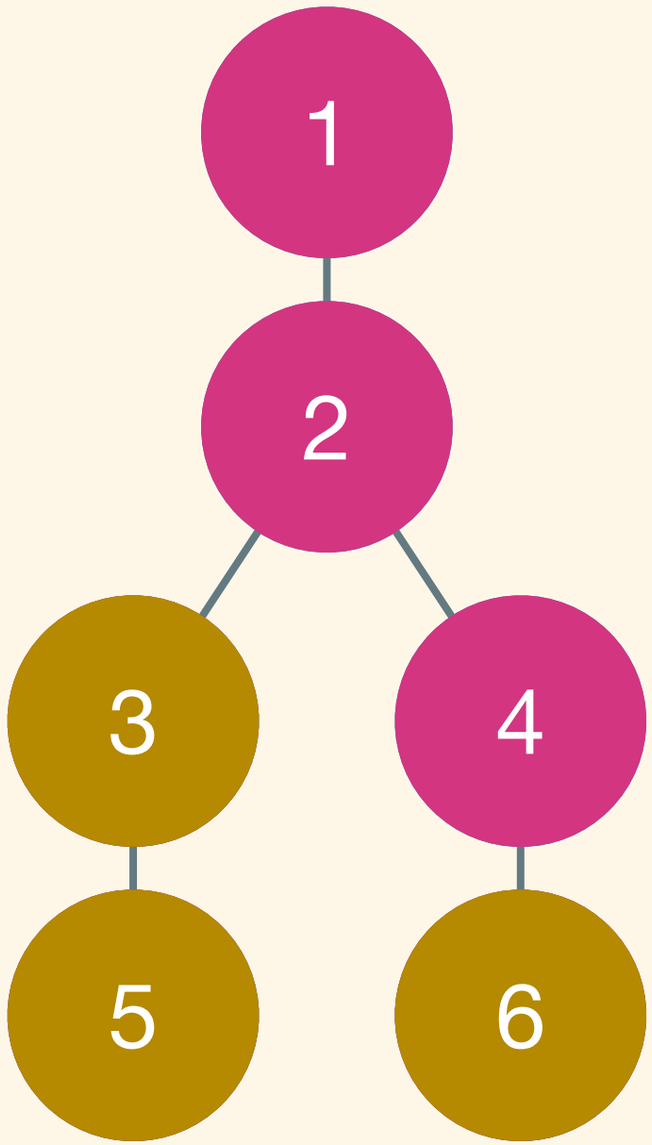


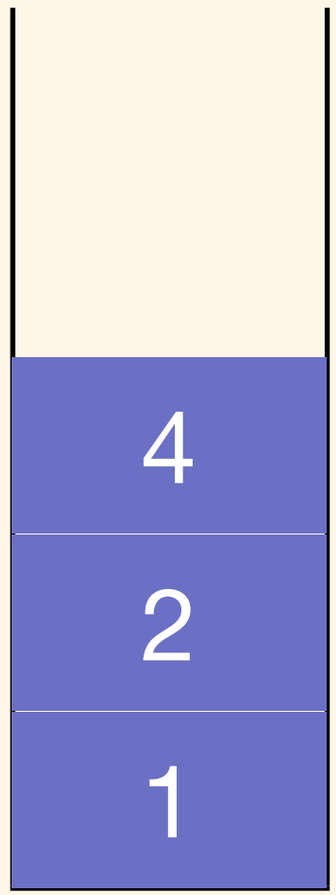
DFS & stack



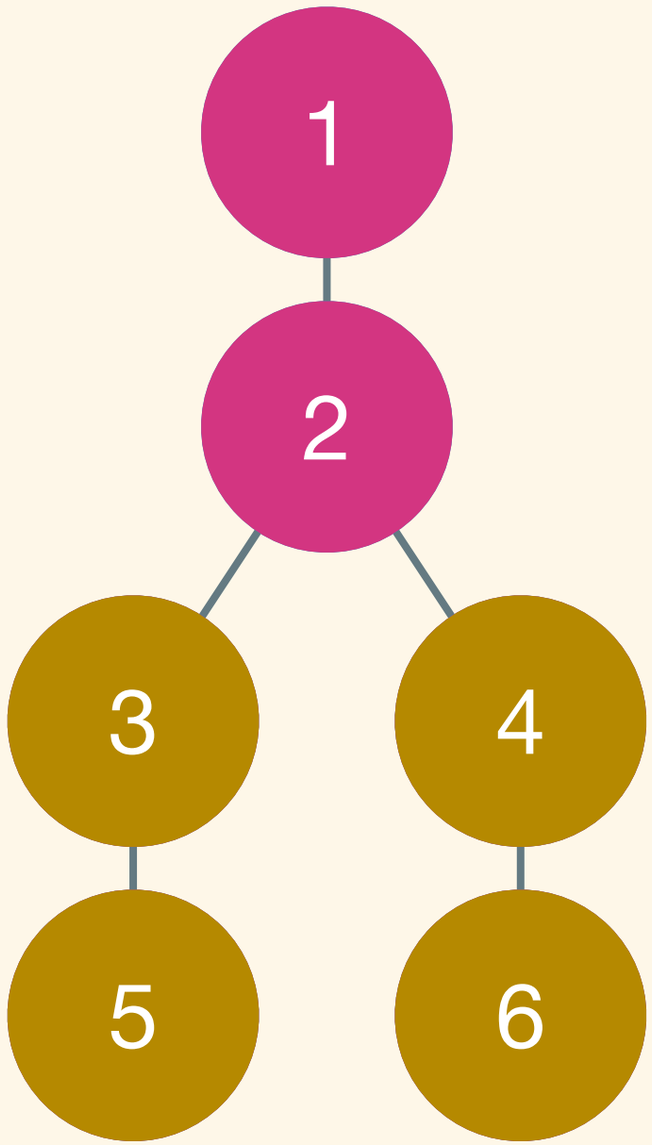


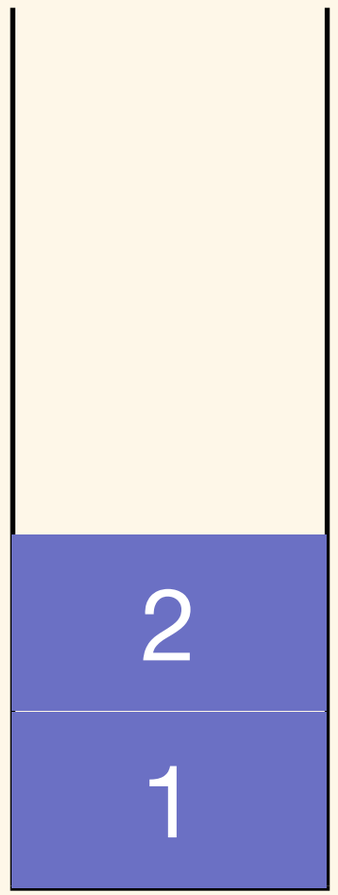
DFS & stack



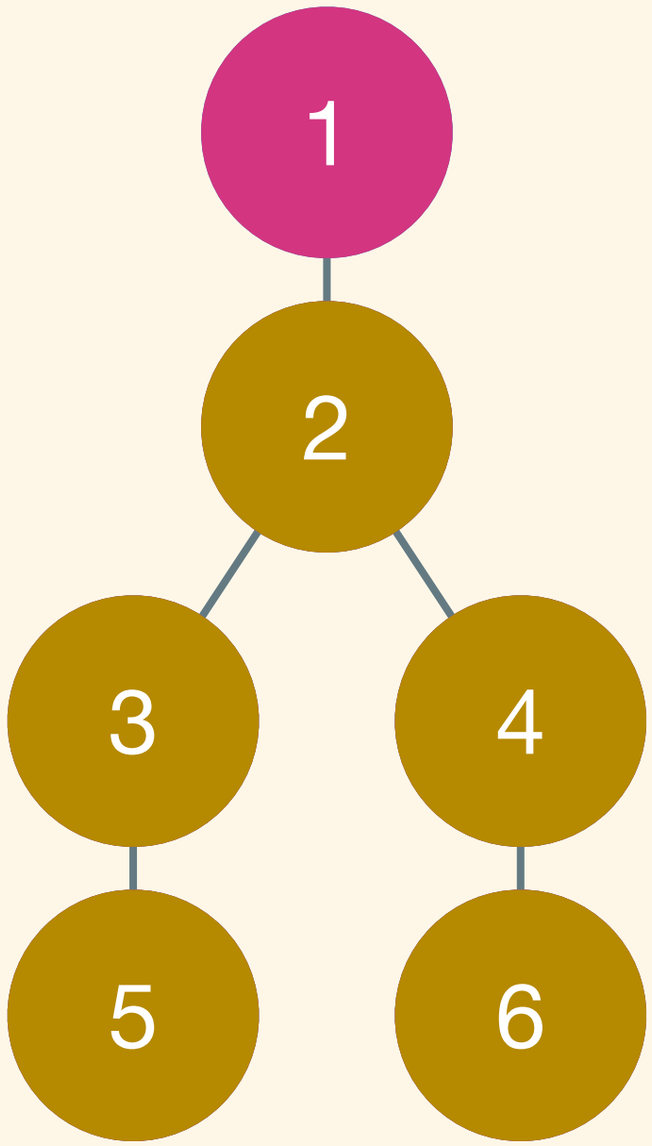


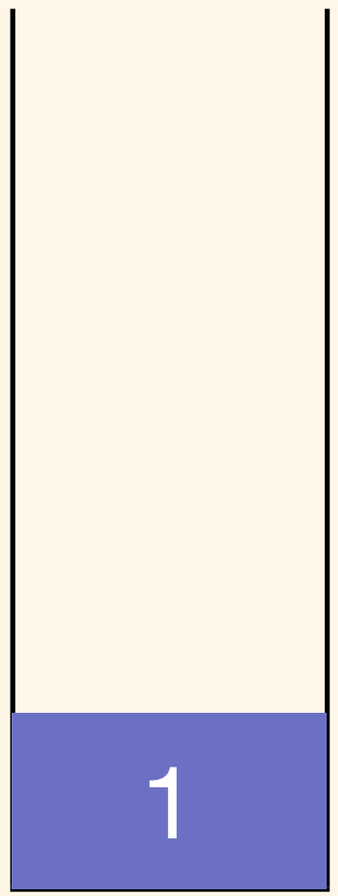
DFS & stack



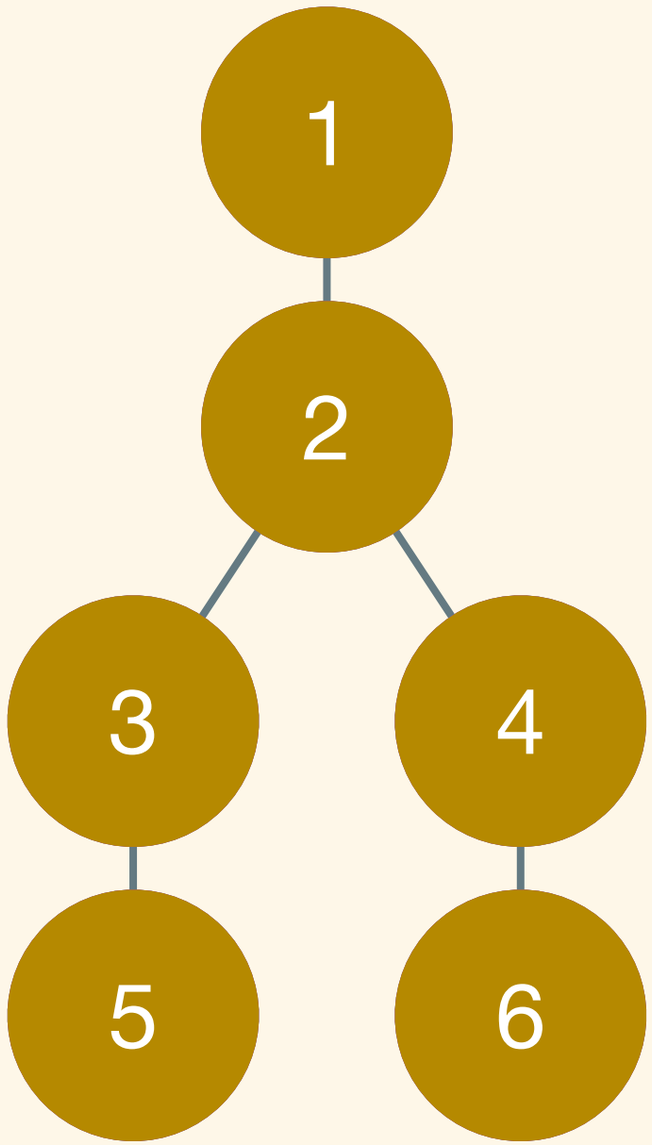


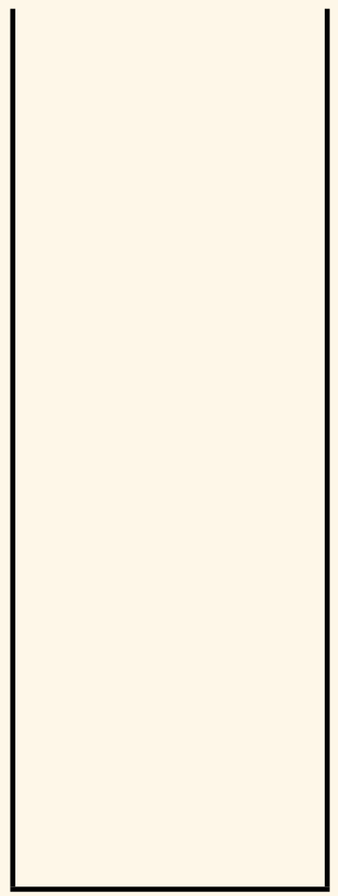
DFS & stack





DFS & stack





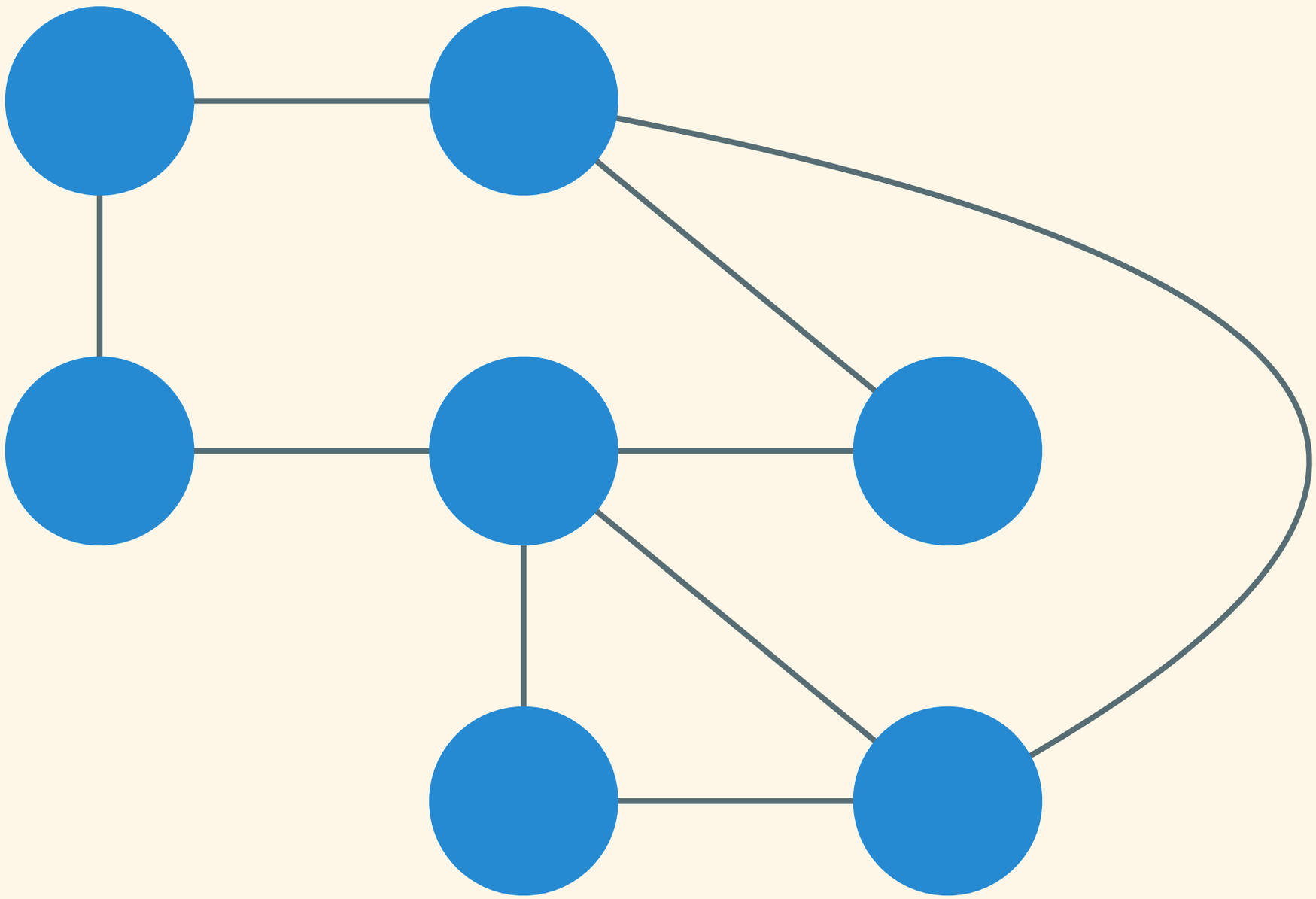
Source Code

```
// There are n nodes.
// g is an adjacent matrix.

void DFS( int now ) {
  if ( visited[ now ] == true )
    return;
  visited[ now ] = true;
  for ( int i = 0; i < n; ++i ) {
    if ( g[ now ][ i ] == true )
      DFS( i );
  }
  return;
}
```

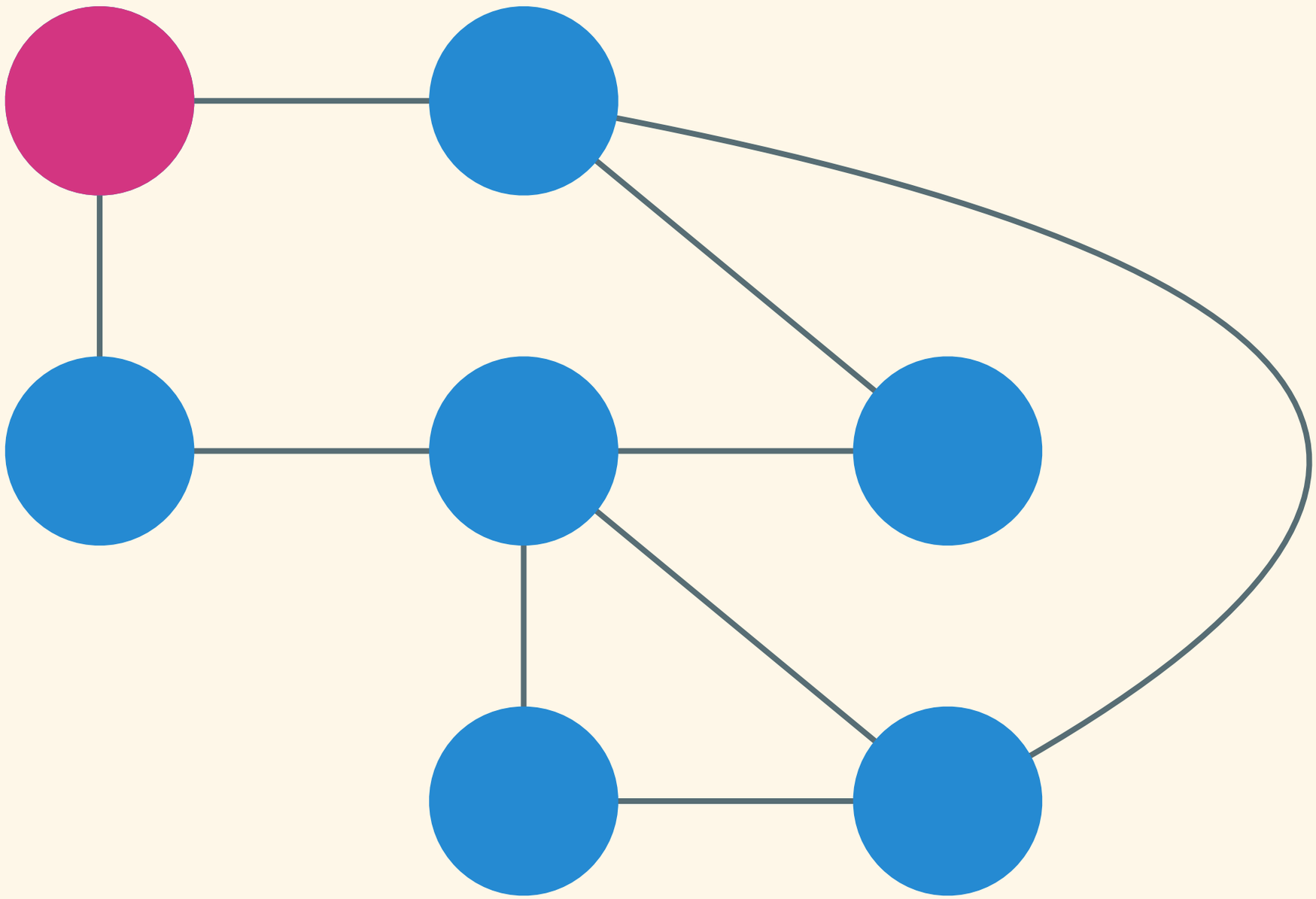
Source Code

```
// There are n nodes.
// g is an adjacent matrix.

void DFS() {
   stack< int > stk;
   stk.push( 0 );
   while ( stk.empty() != true ) {
      int now = stk.top();
      visited[ now ] = true;
      bool noleaf = true;
      for ( int i = 0; i < n; ++i )
         if ( g[ now ][ i ] == true && visited[ i ] != true ) {
            stk.push( i );
            noleaf = false;
            break;
         }
      if ( noleaf == true )
         stk.pop();
   }
}
```

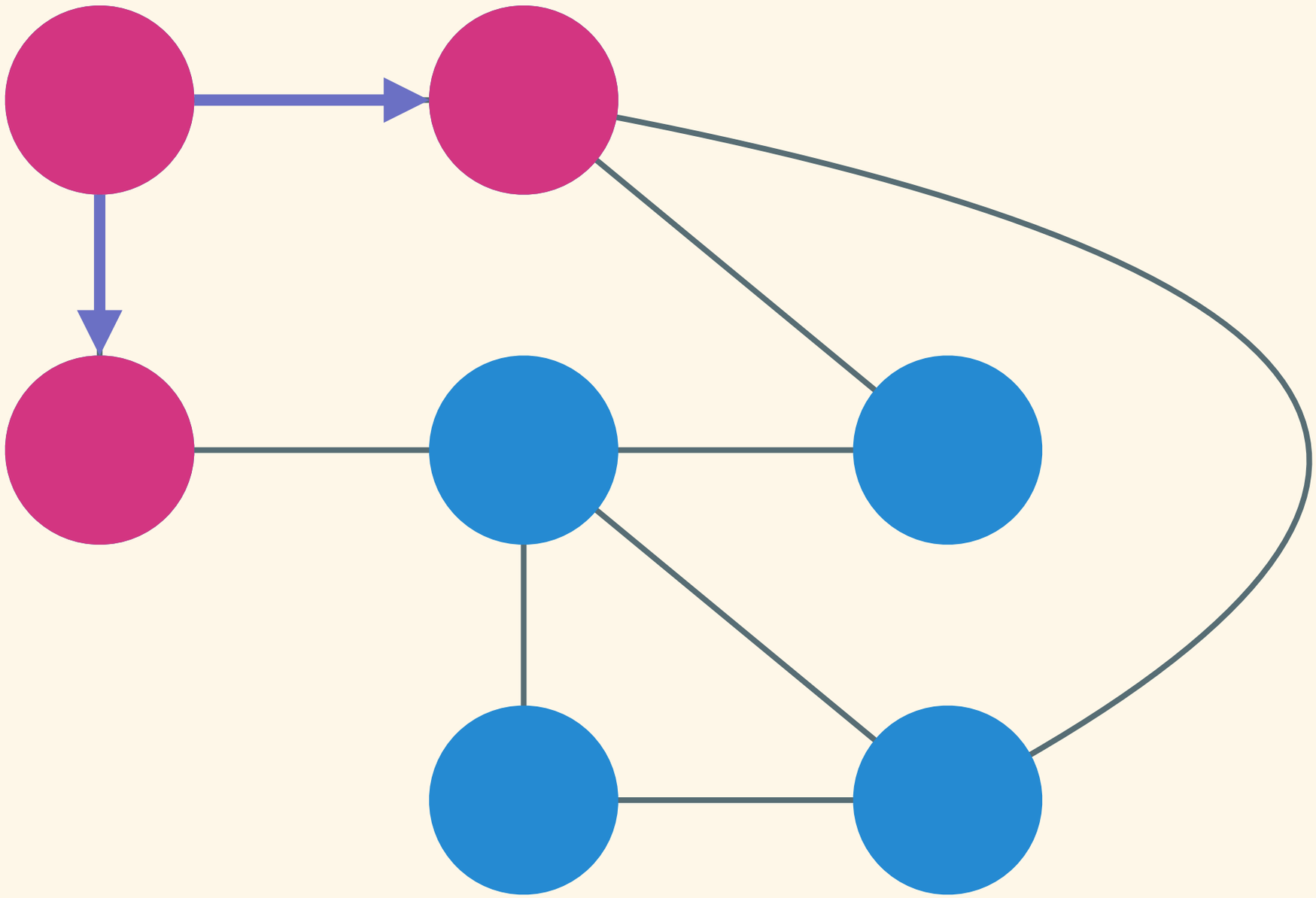
BFS (Breadth-First Search)



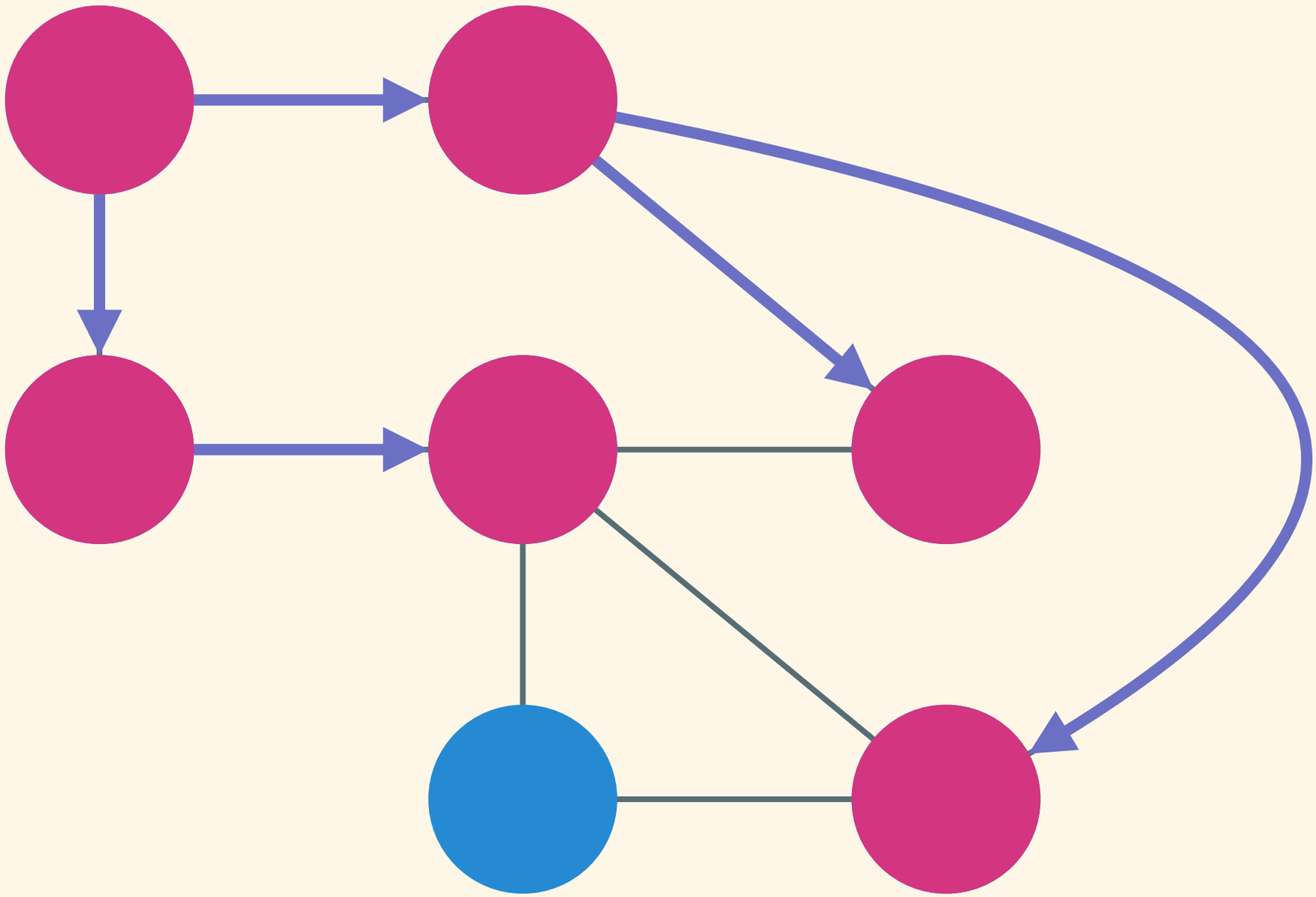
BFS (Breadth-First Search)



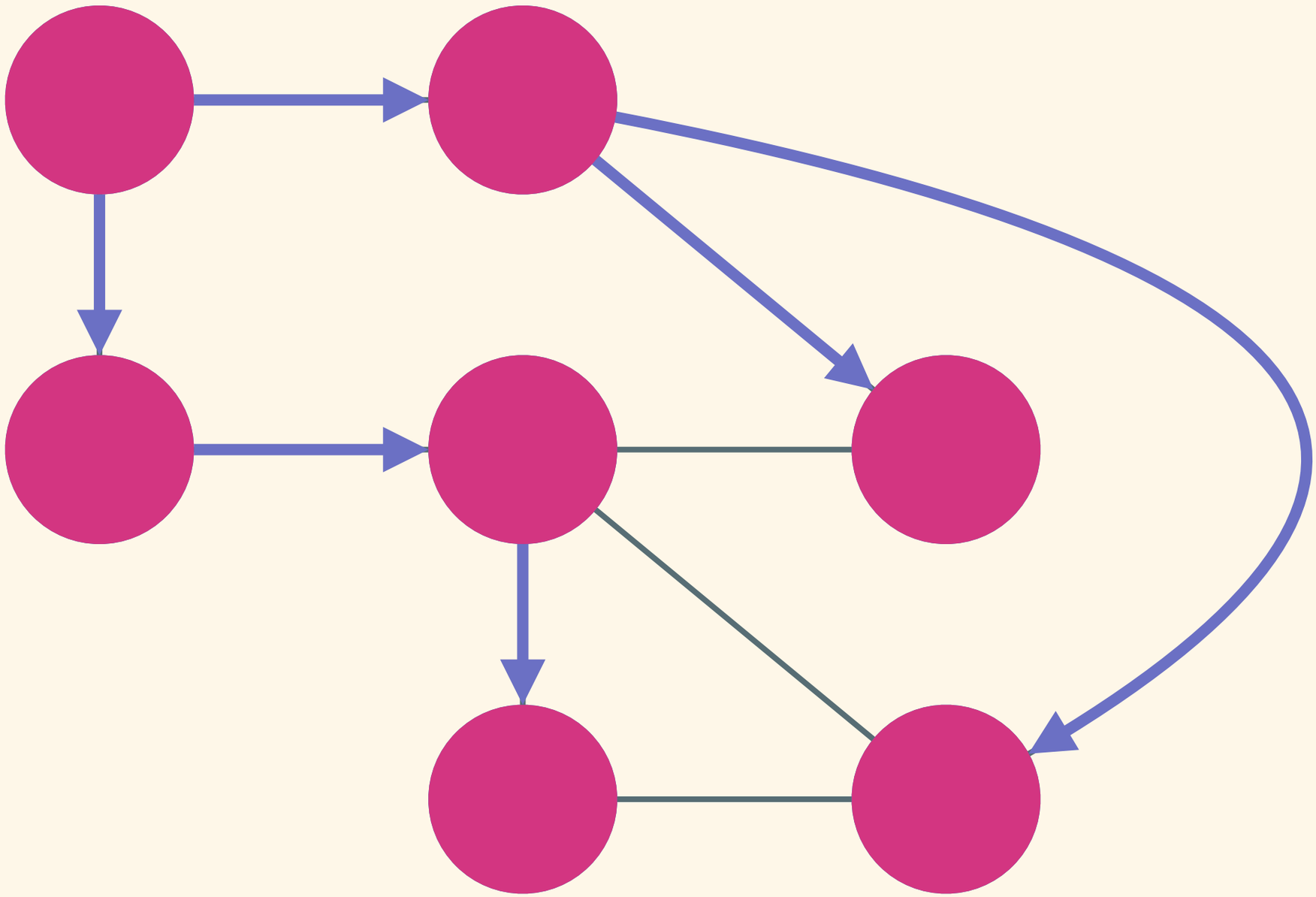
BFS (Breadth-First Search)



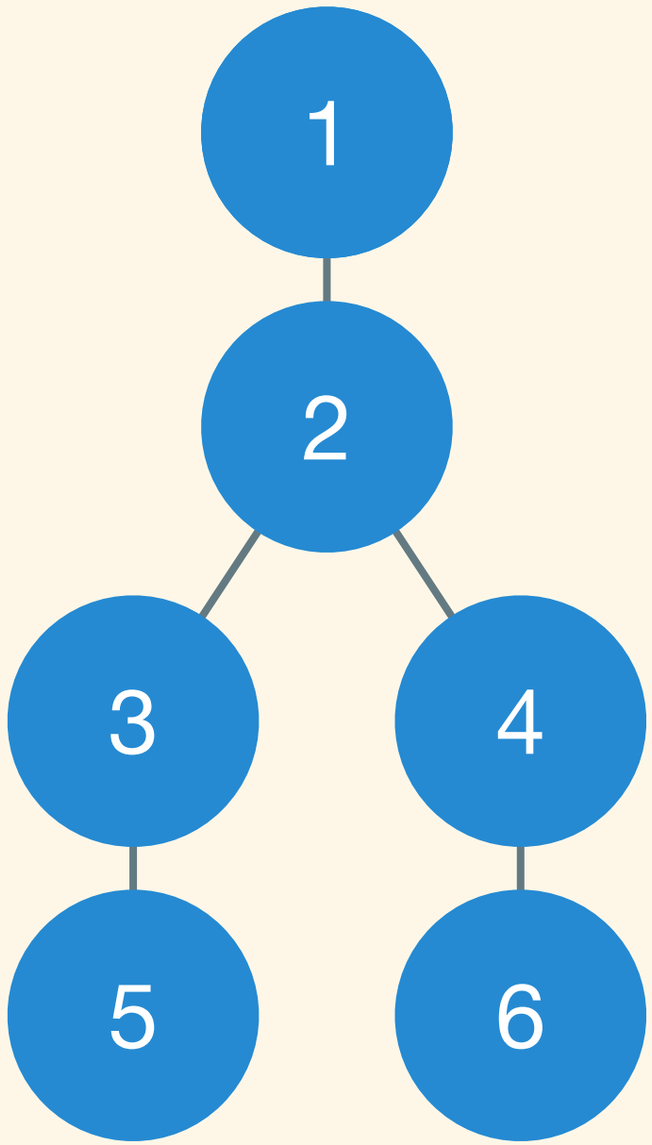
BFS (Breadth-First Search)



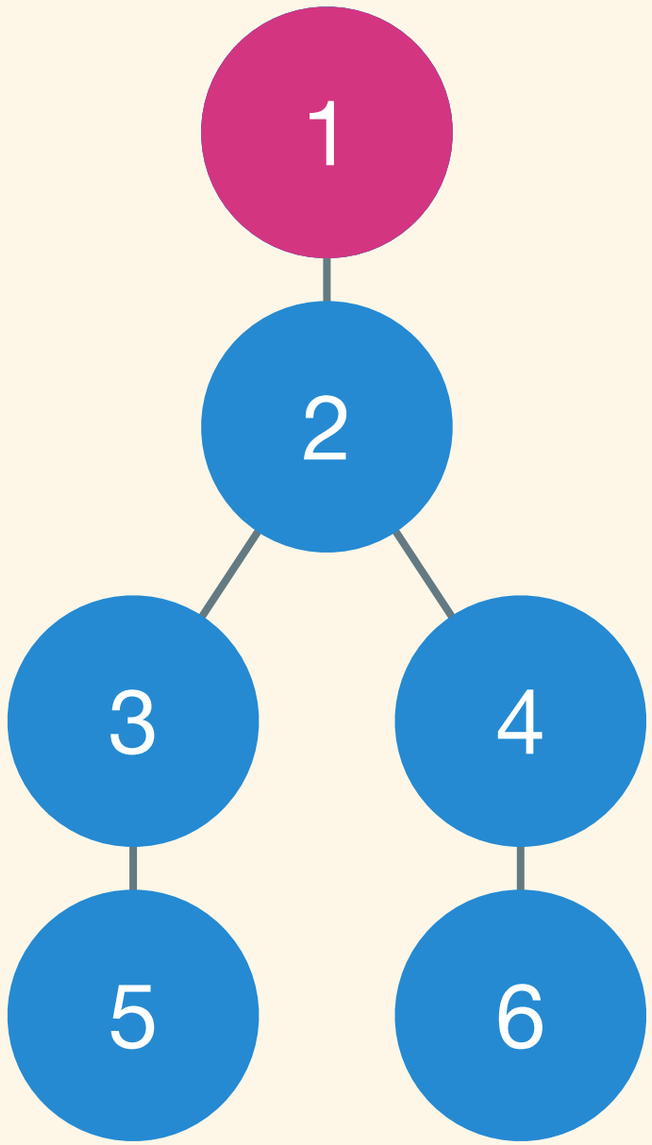
BFS (Breadth-First Search)

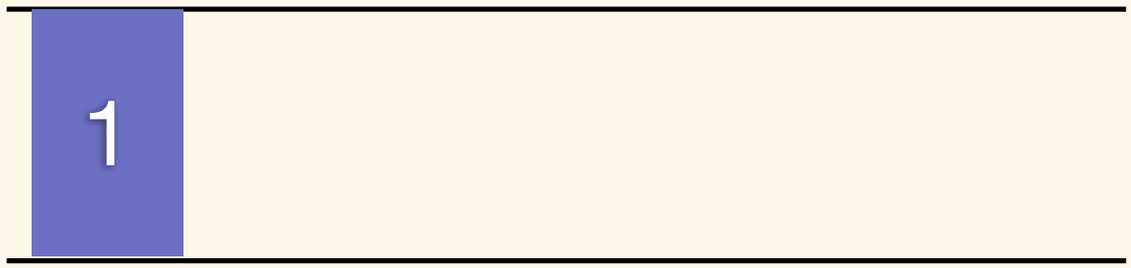


BFS & queue

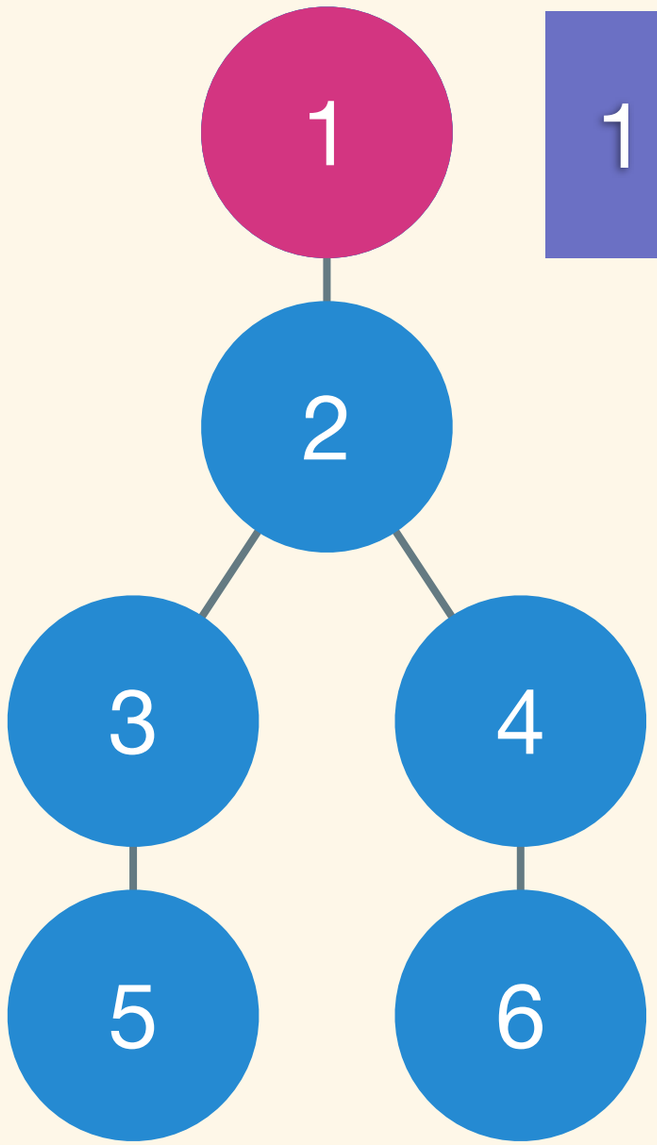


BFS & queue



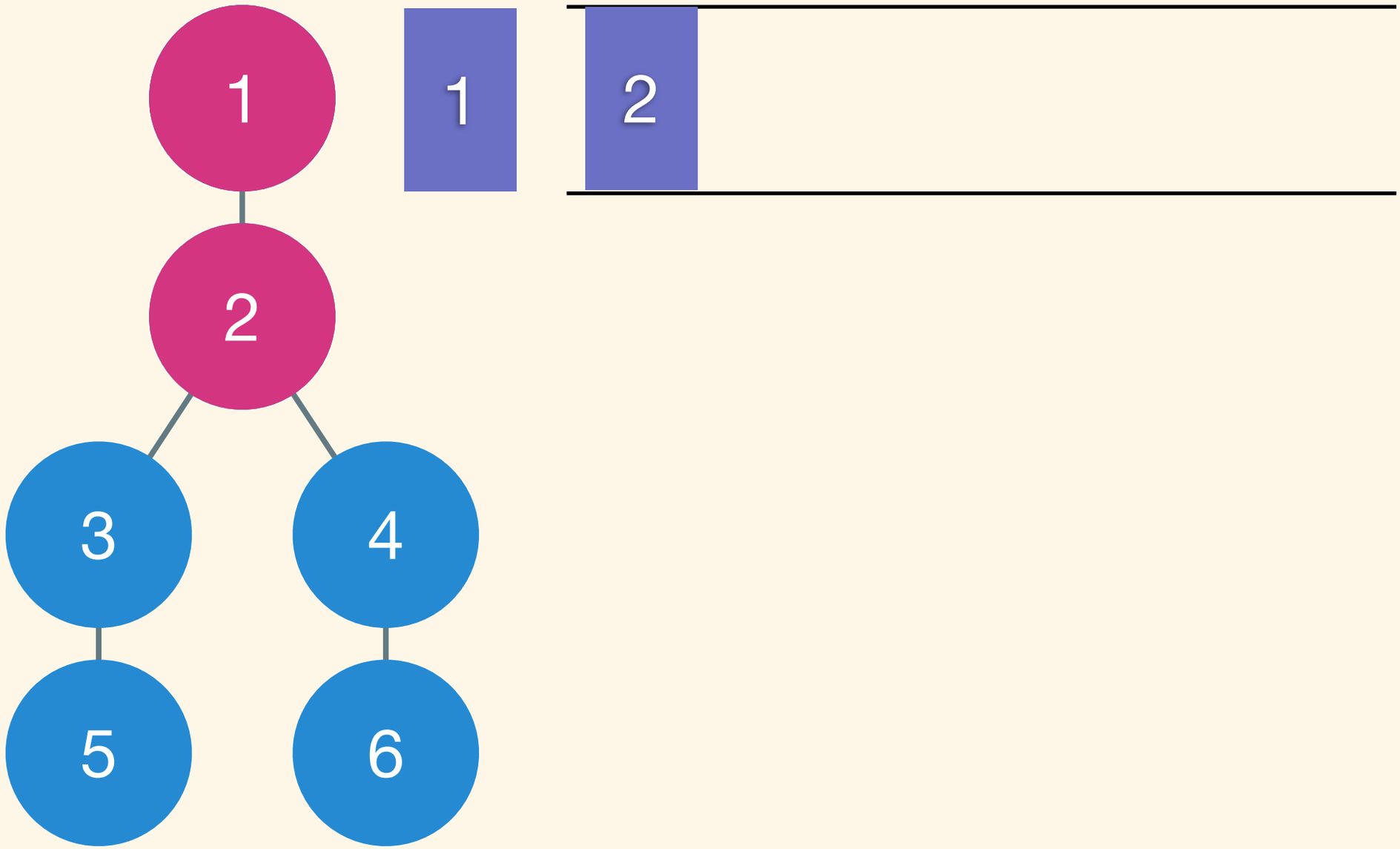


BFS & queue

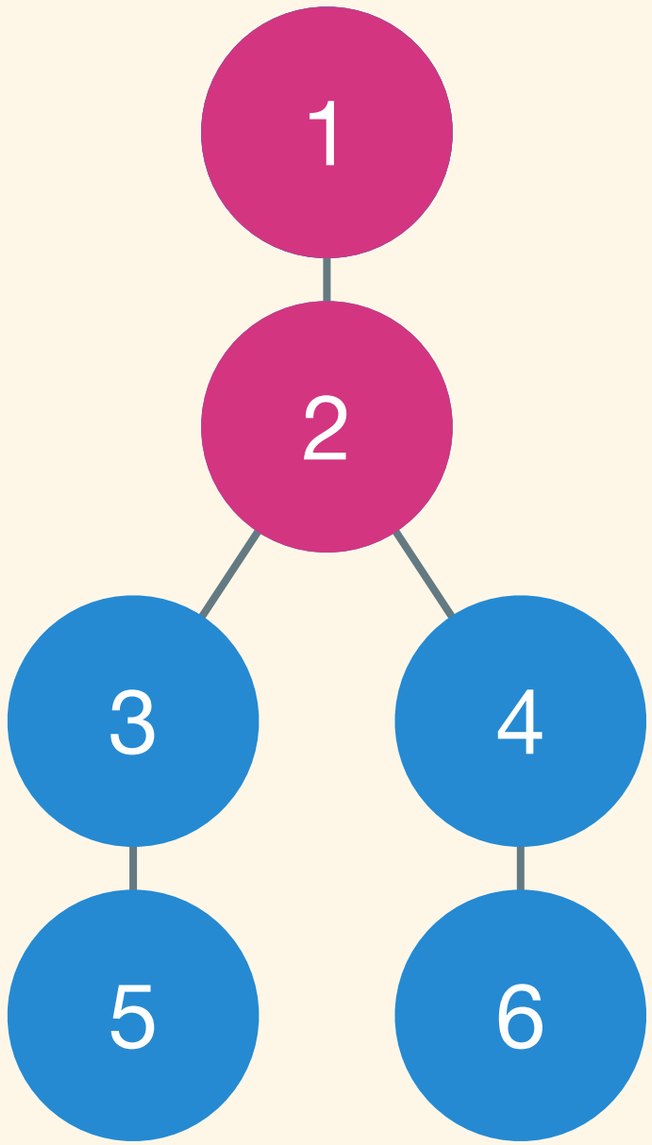


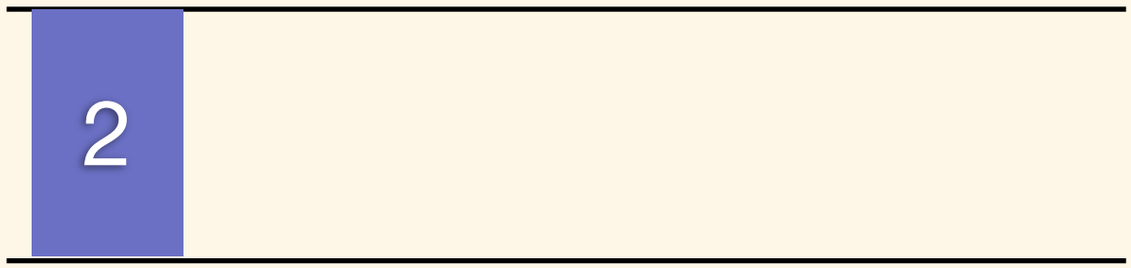
1

BFS & queue

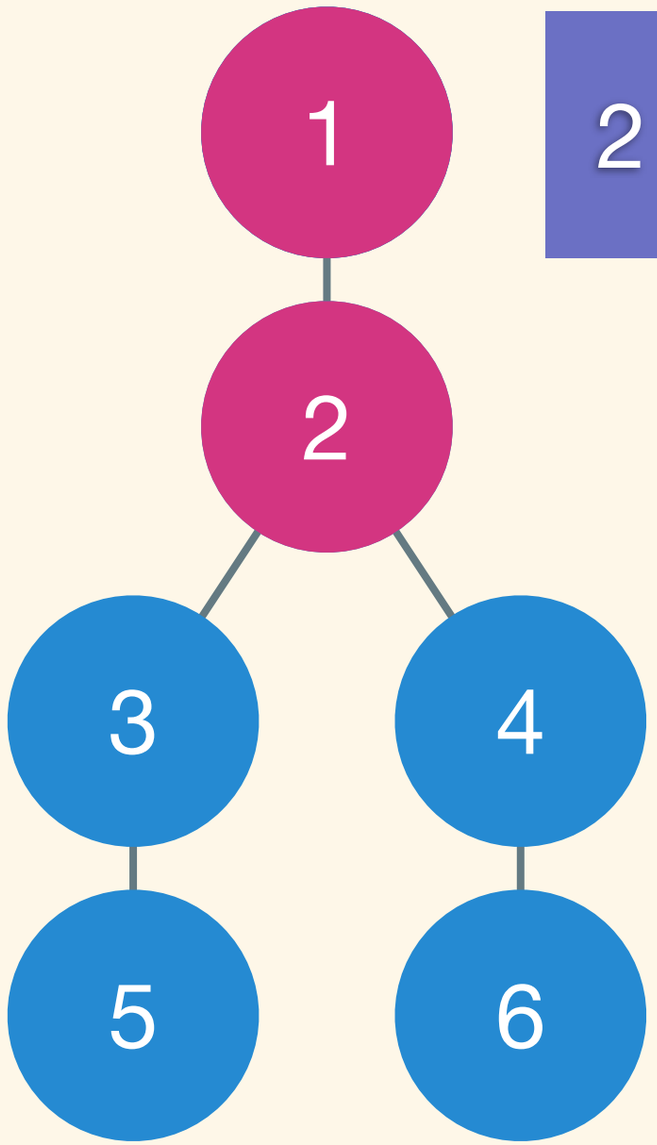


BFS & queue

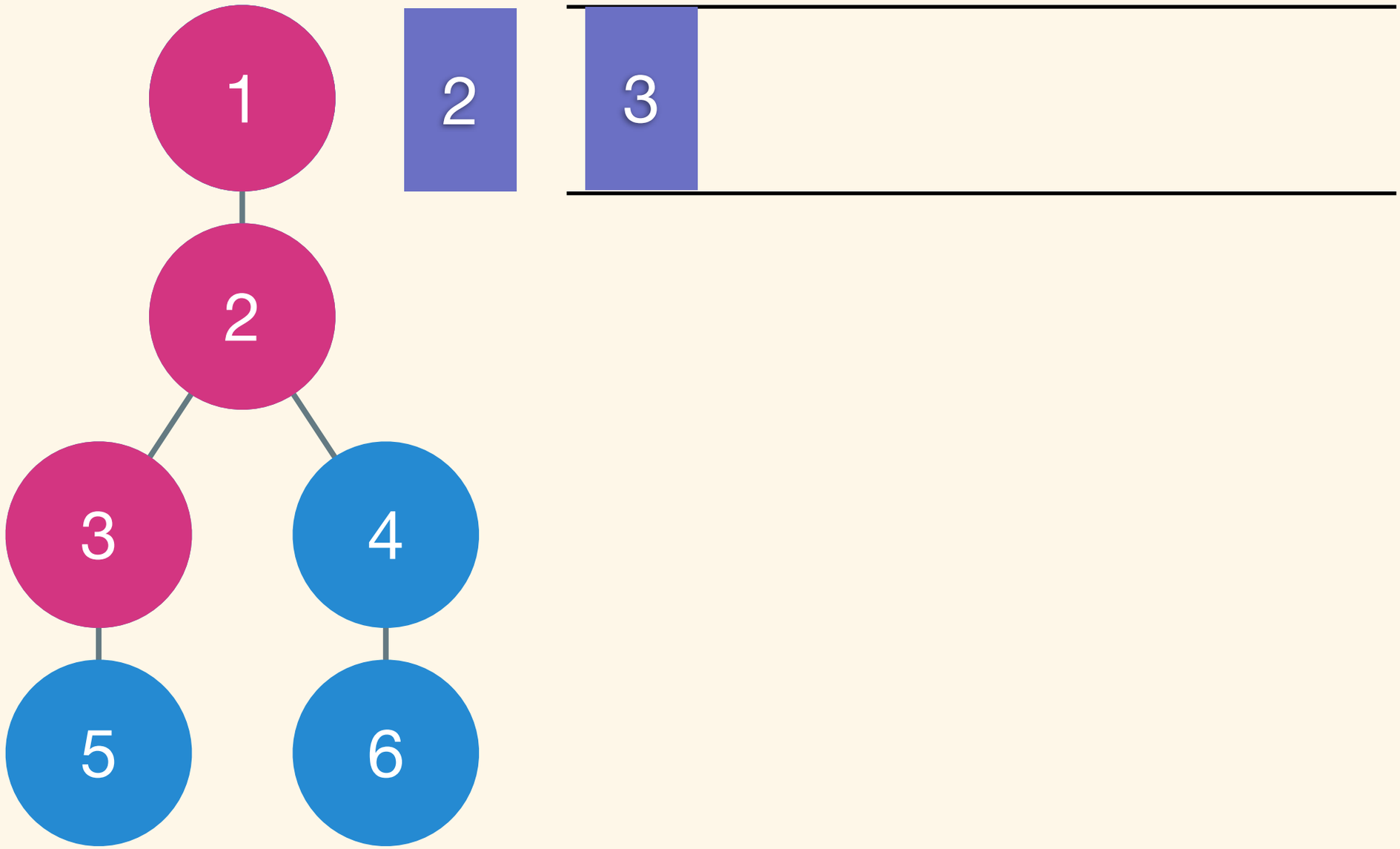




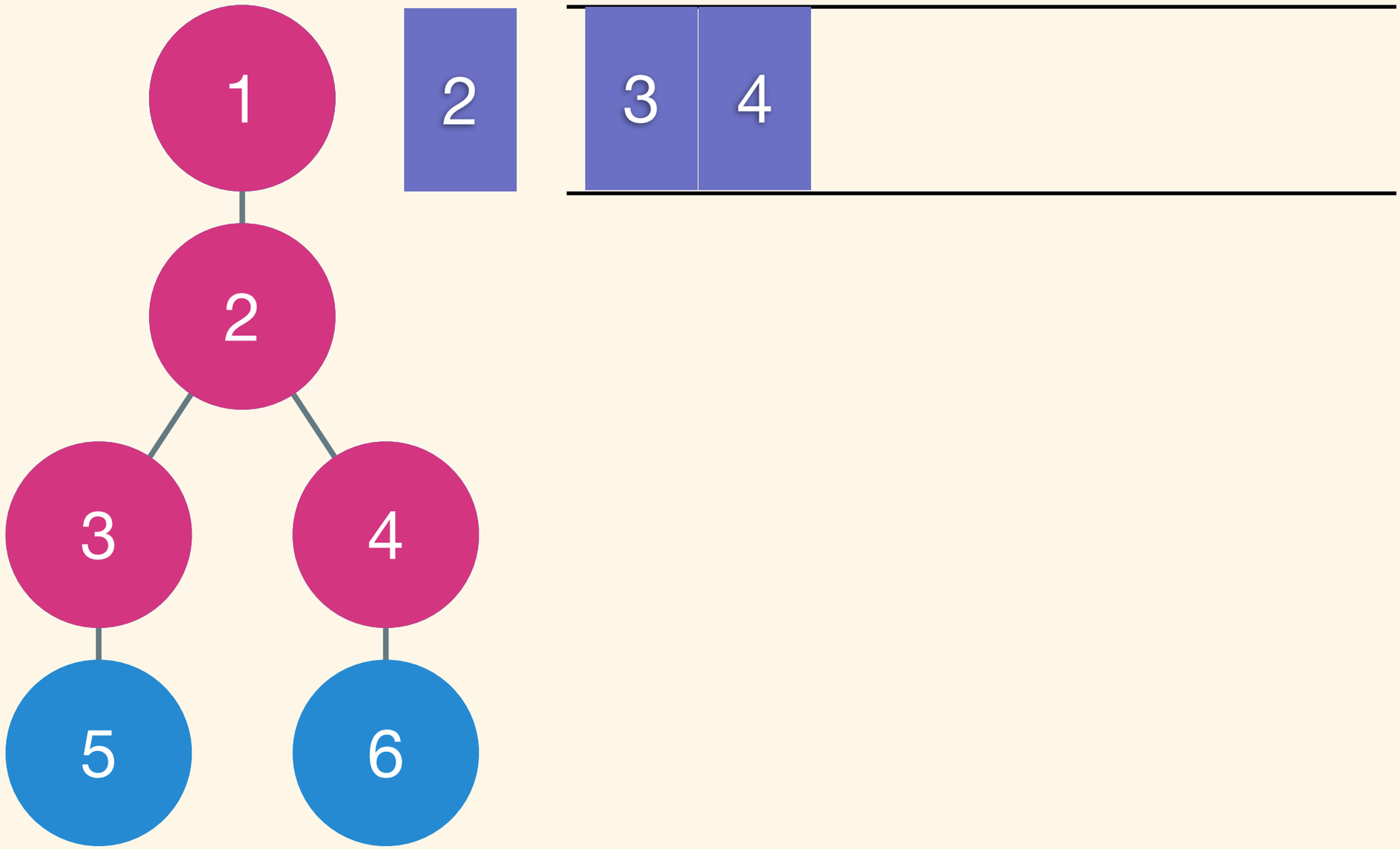
BFS & queue



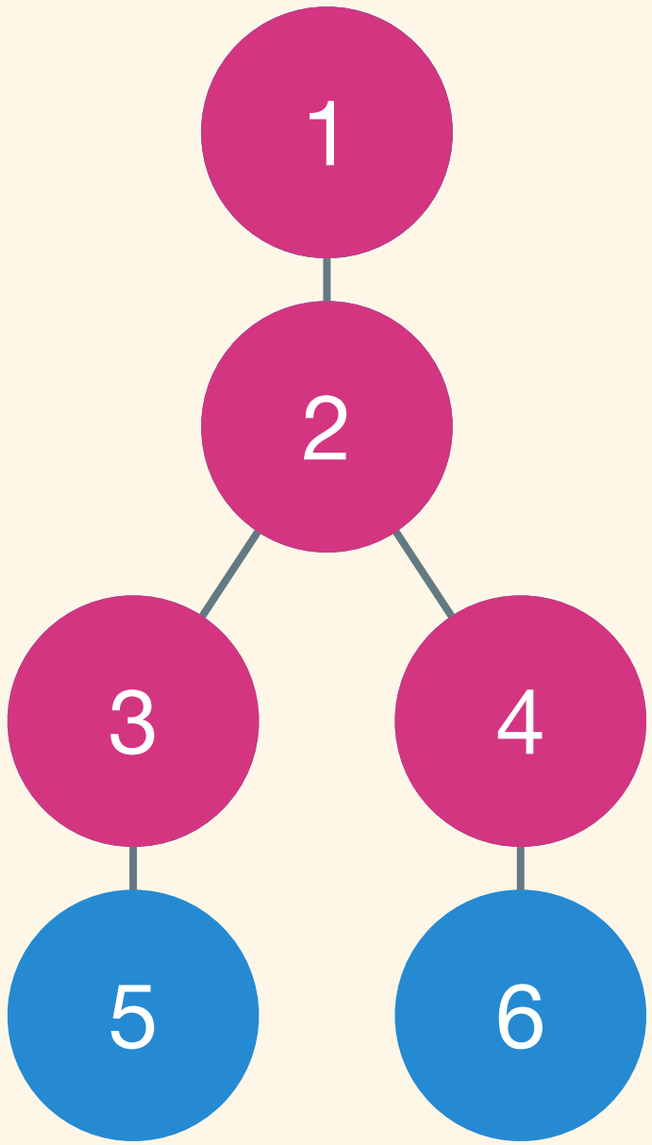
BFS & queue

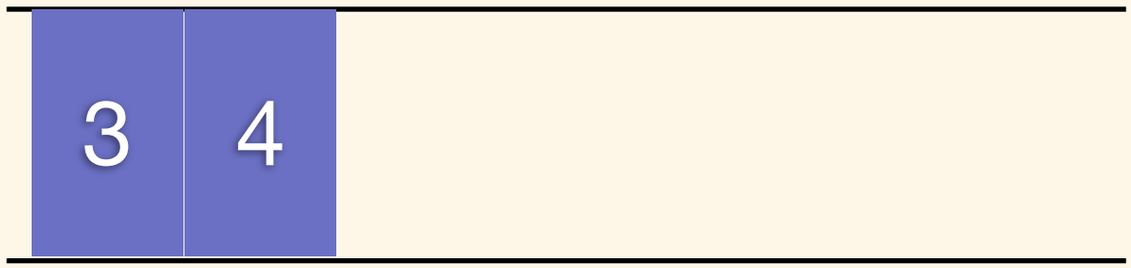


BFS & queue

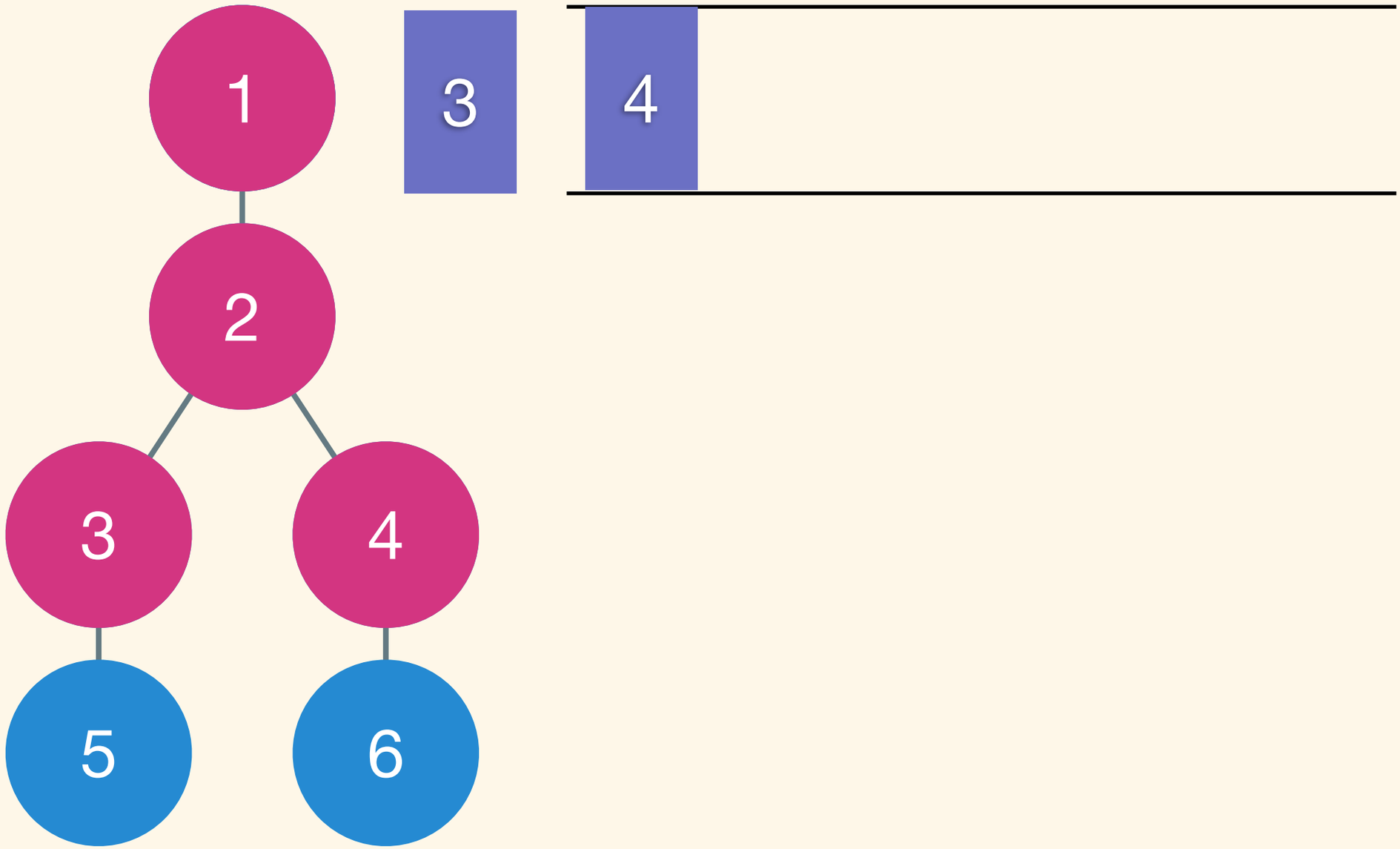


BFS & queue

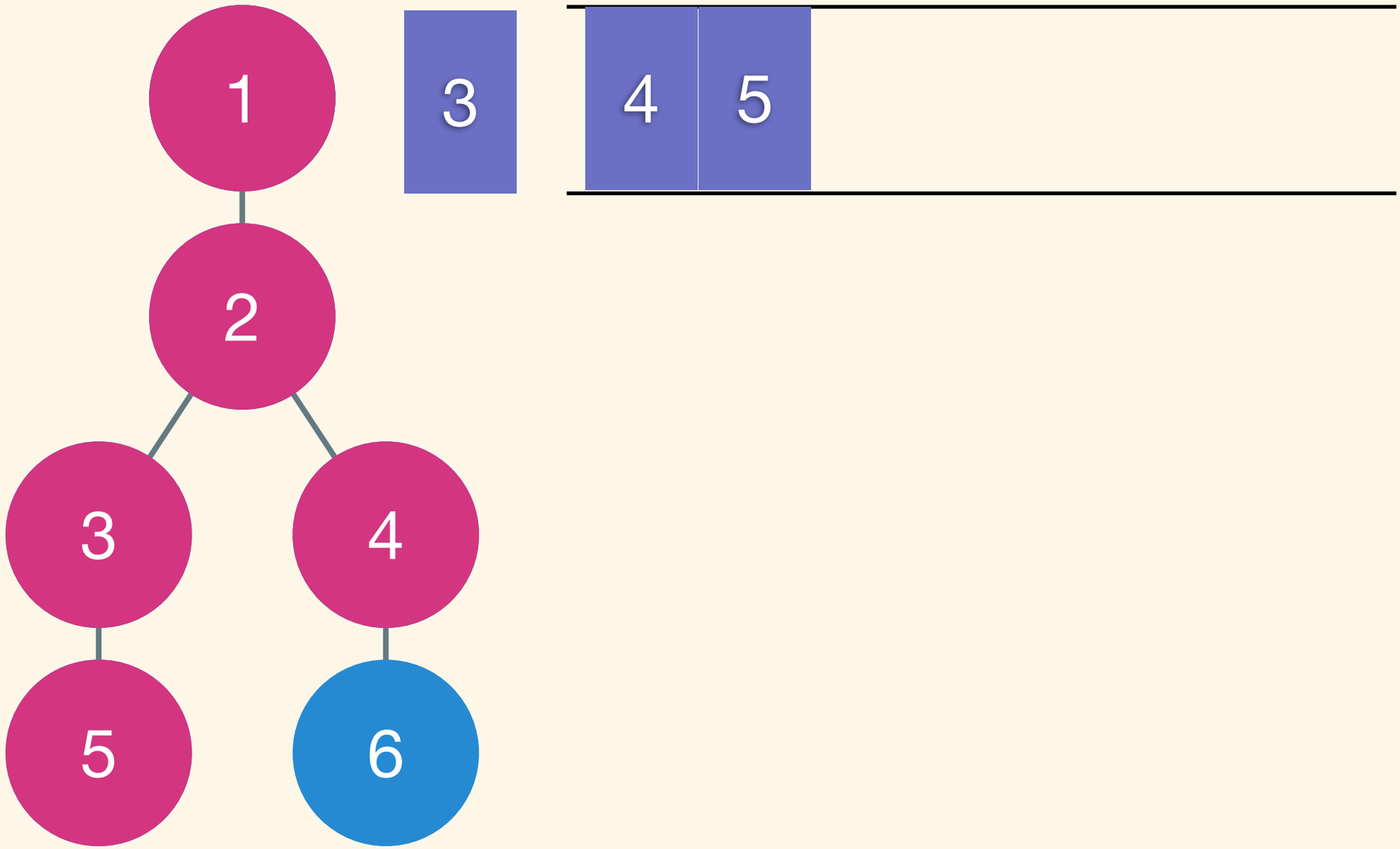




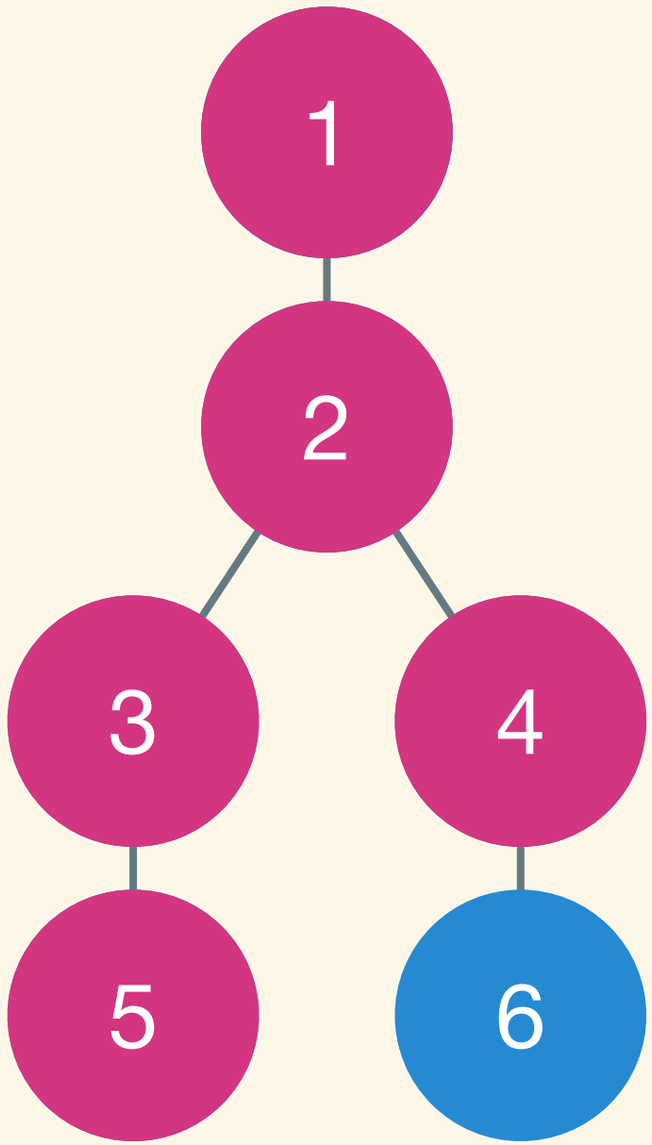
BFS & queue

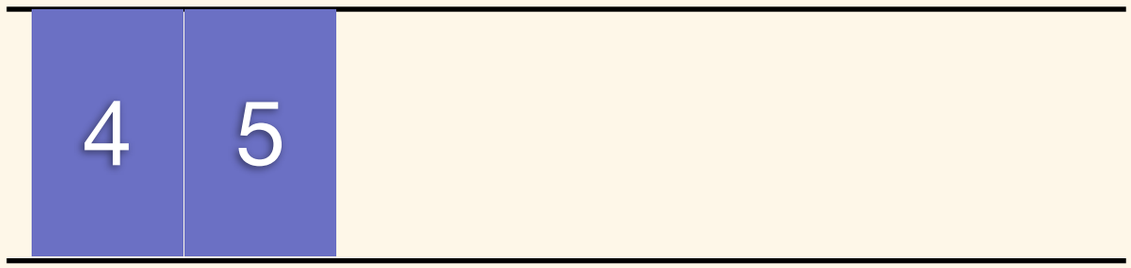


BFS & queue

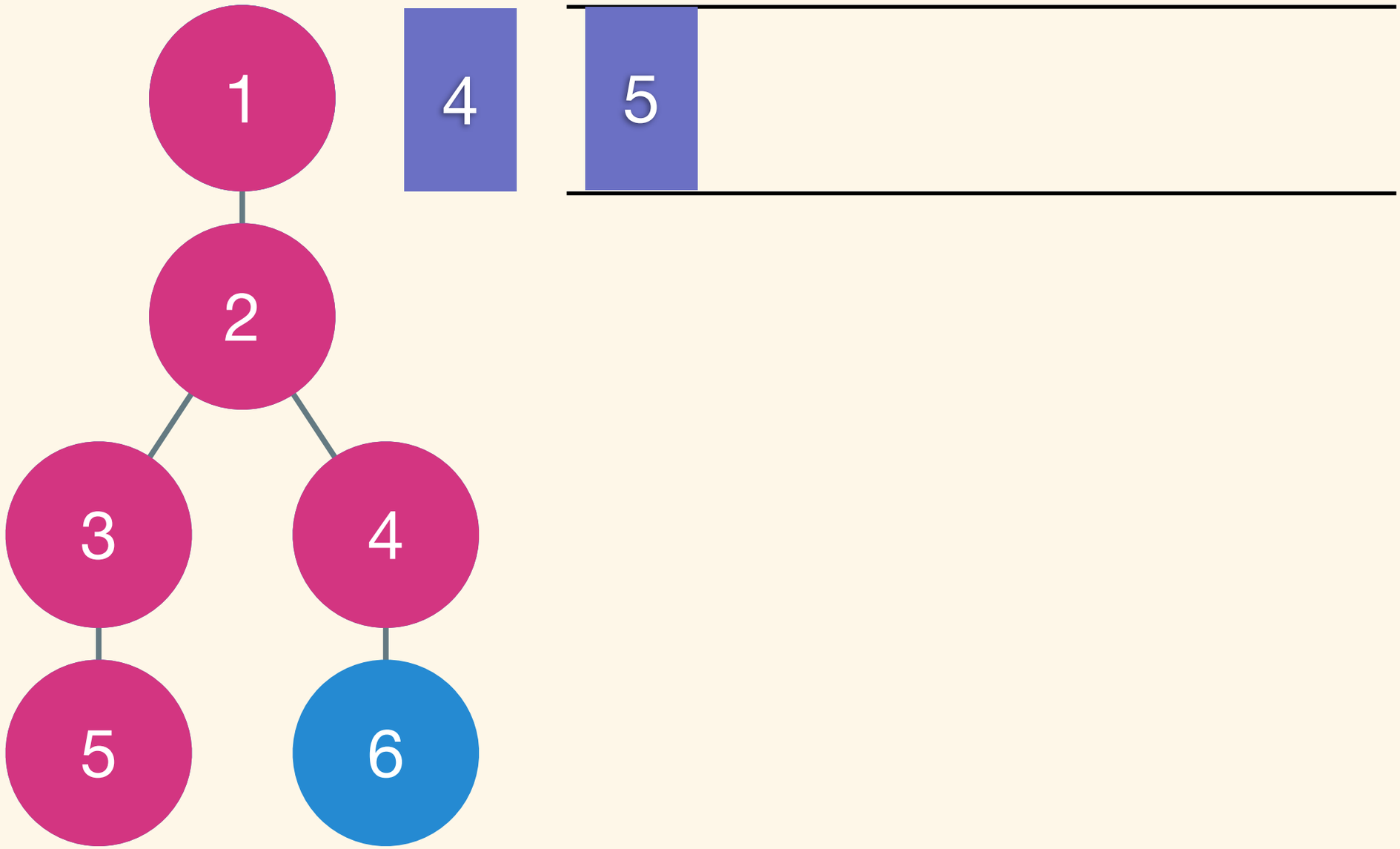


BFS & queue

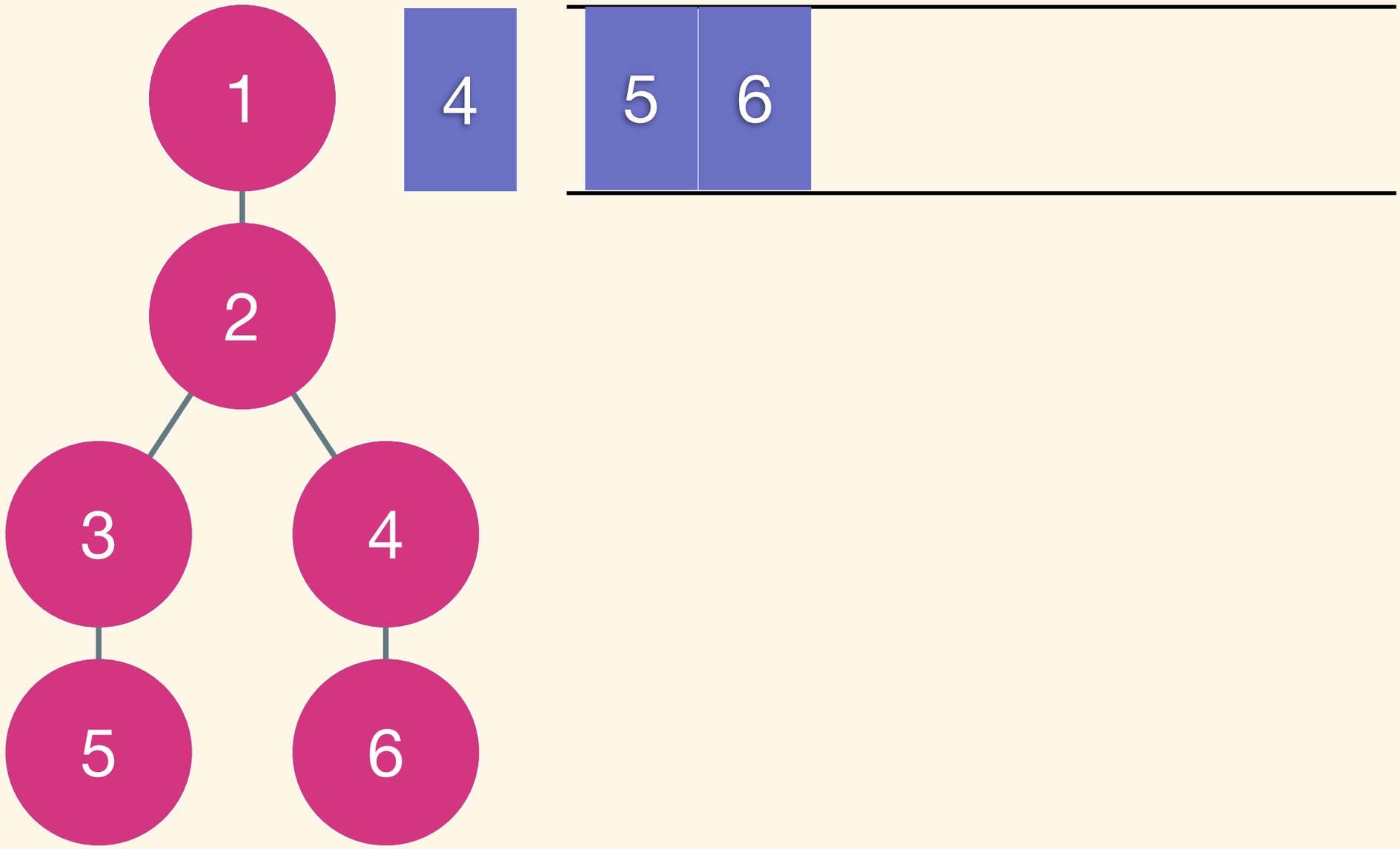




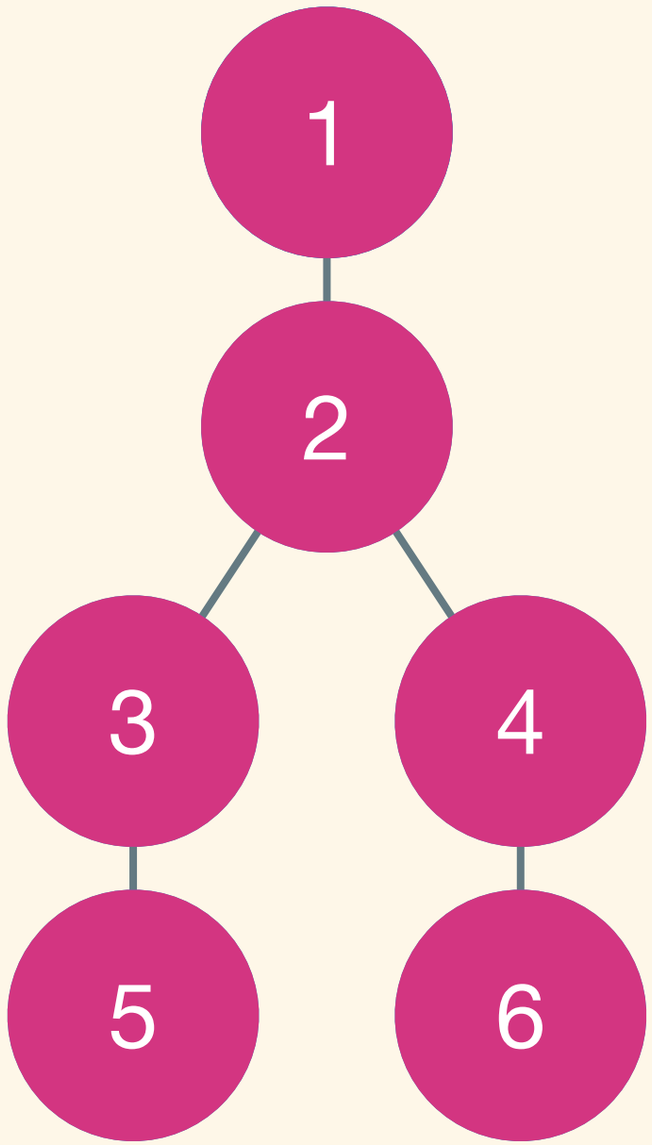
BFS & queue

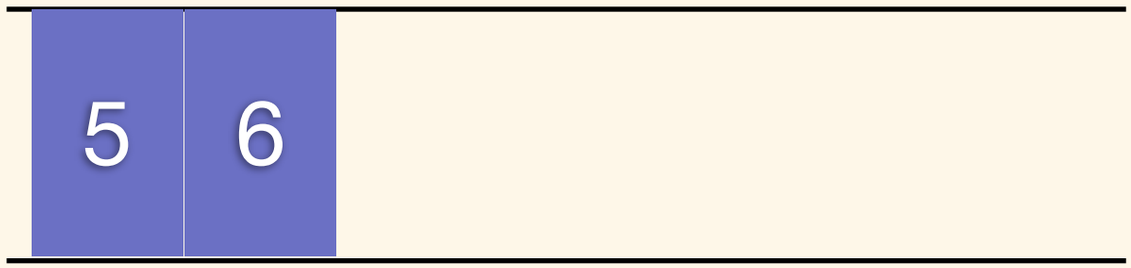


BFS & queue

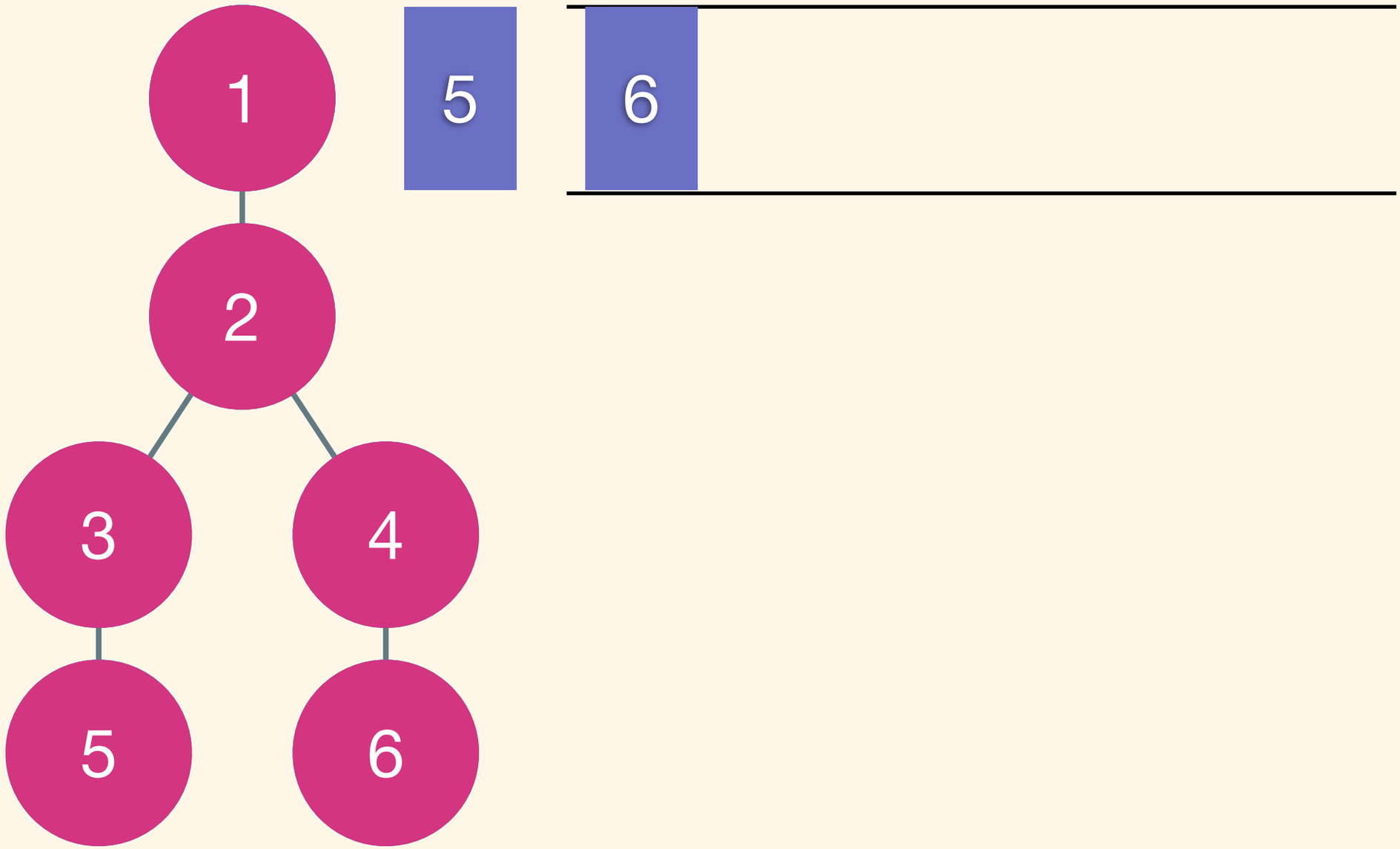


BFS & queue

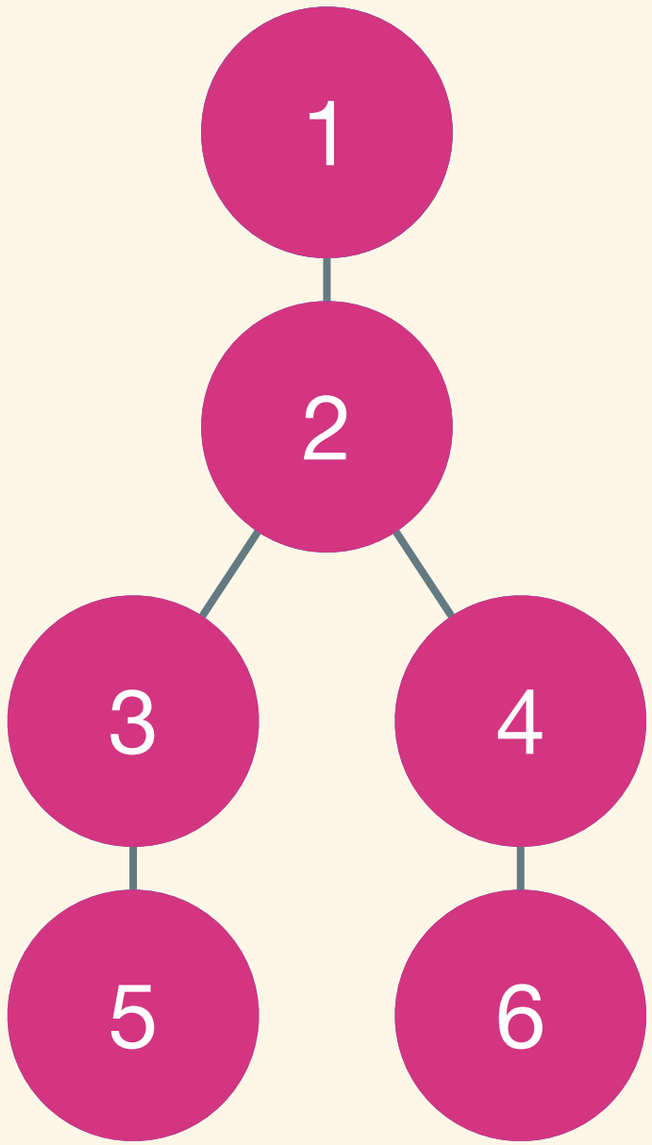


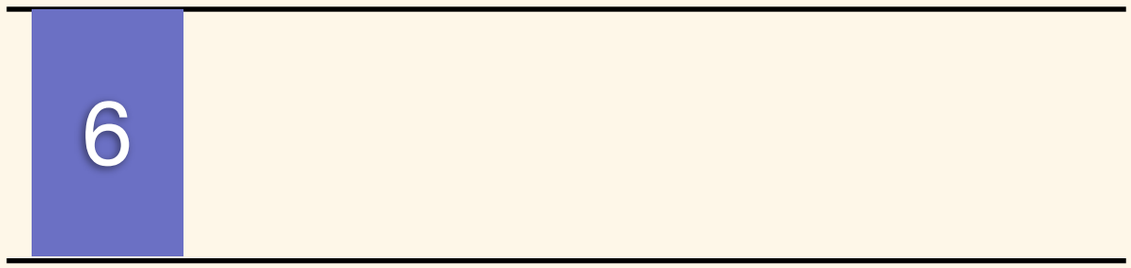


BFS & queue

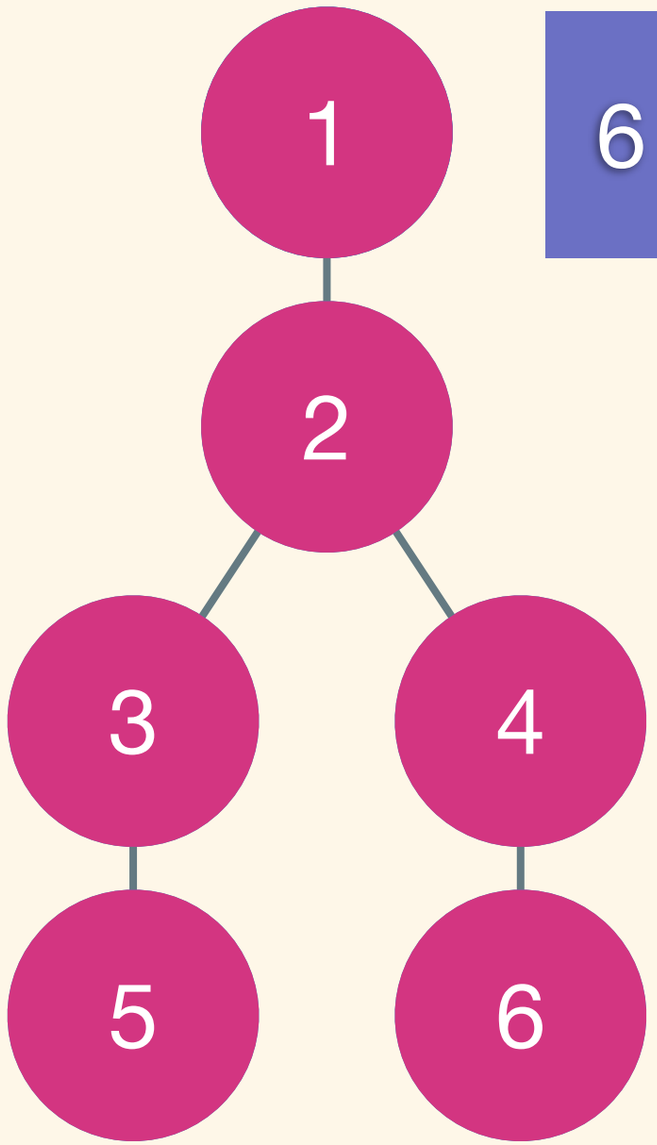


BFS & queue



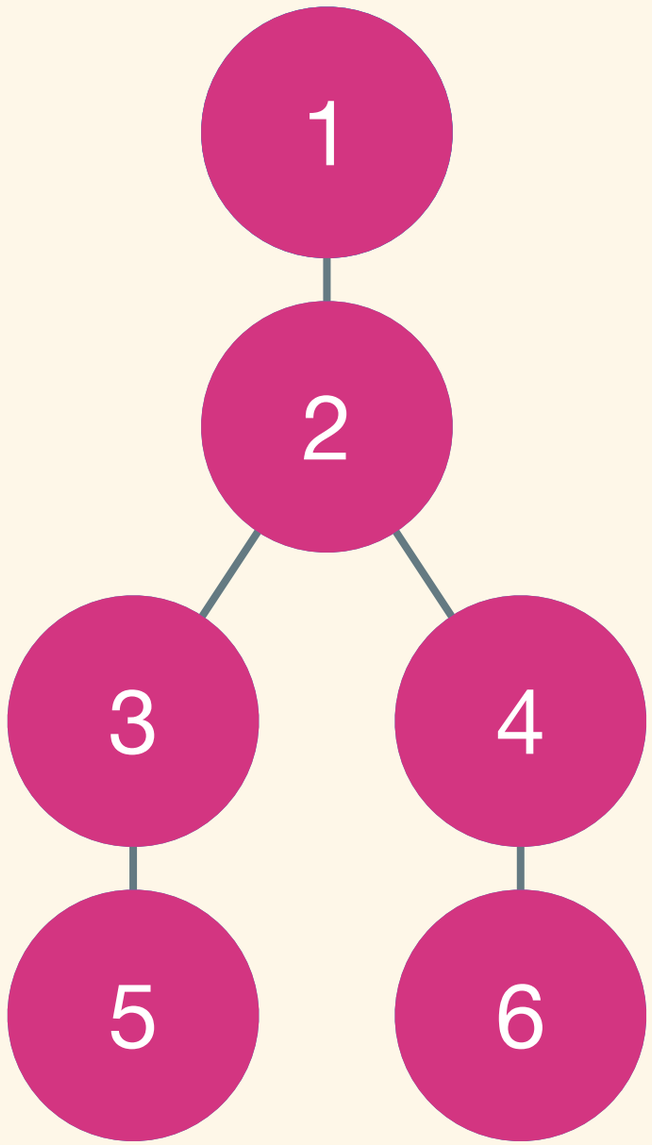


BFS & queue



6

BFS & queue



Source Code

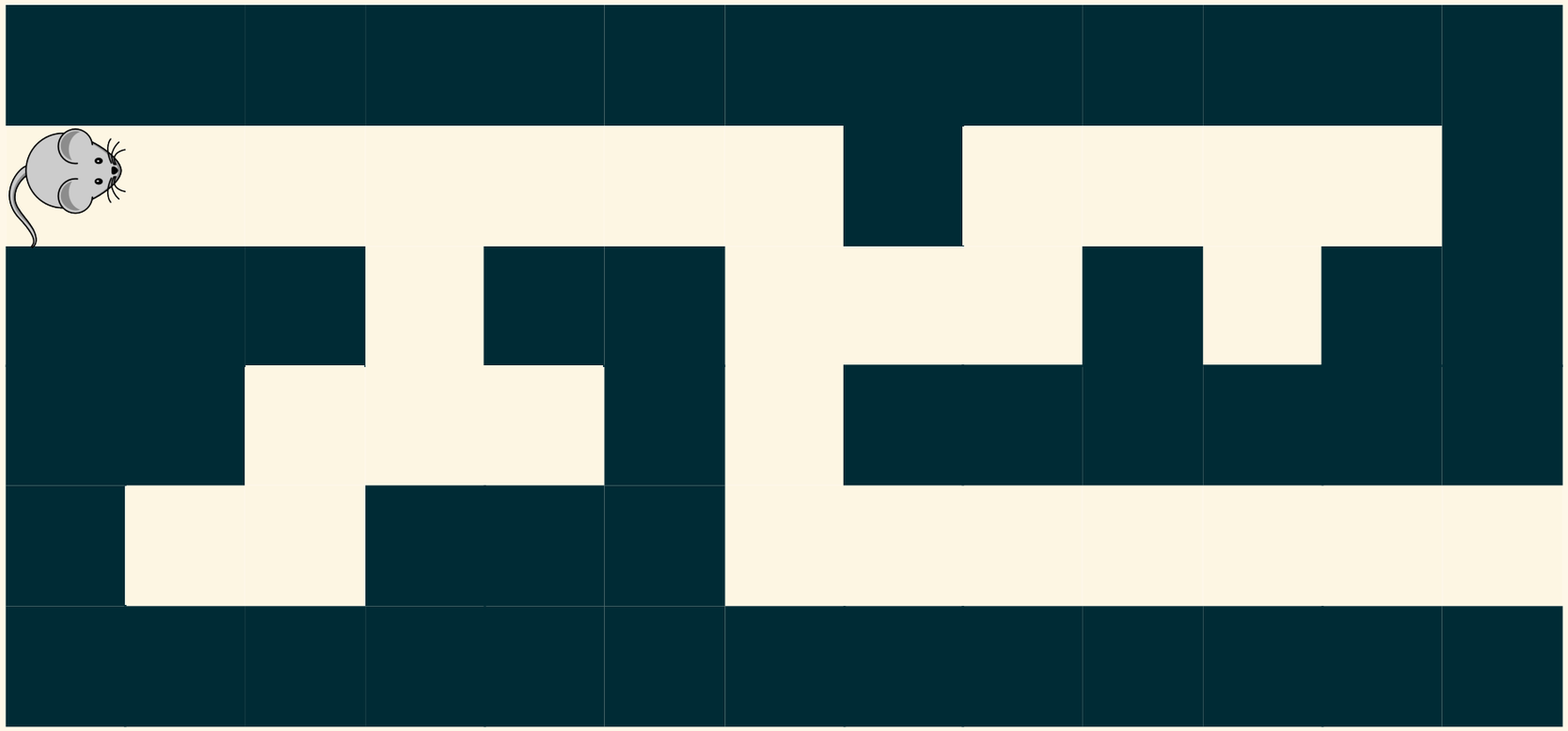
```
// There are n nodes.
// g is an adjacent matrix.

void BFS() {
  queue< int > que;
  que.push( 0 );
  while ( que.empty() != true ) {
    int now = que.front();
    que.pop();
    visited[ now ] = true;
    for ( int i = 0; i < n; ++i )
      if ( g[ now ][ i ] == true )
        que.push( i );
  }
}
```

Time Complexity

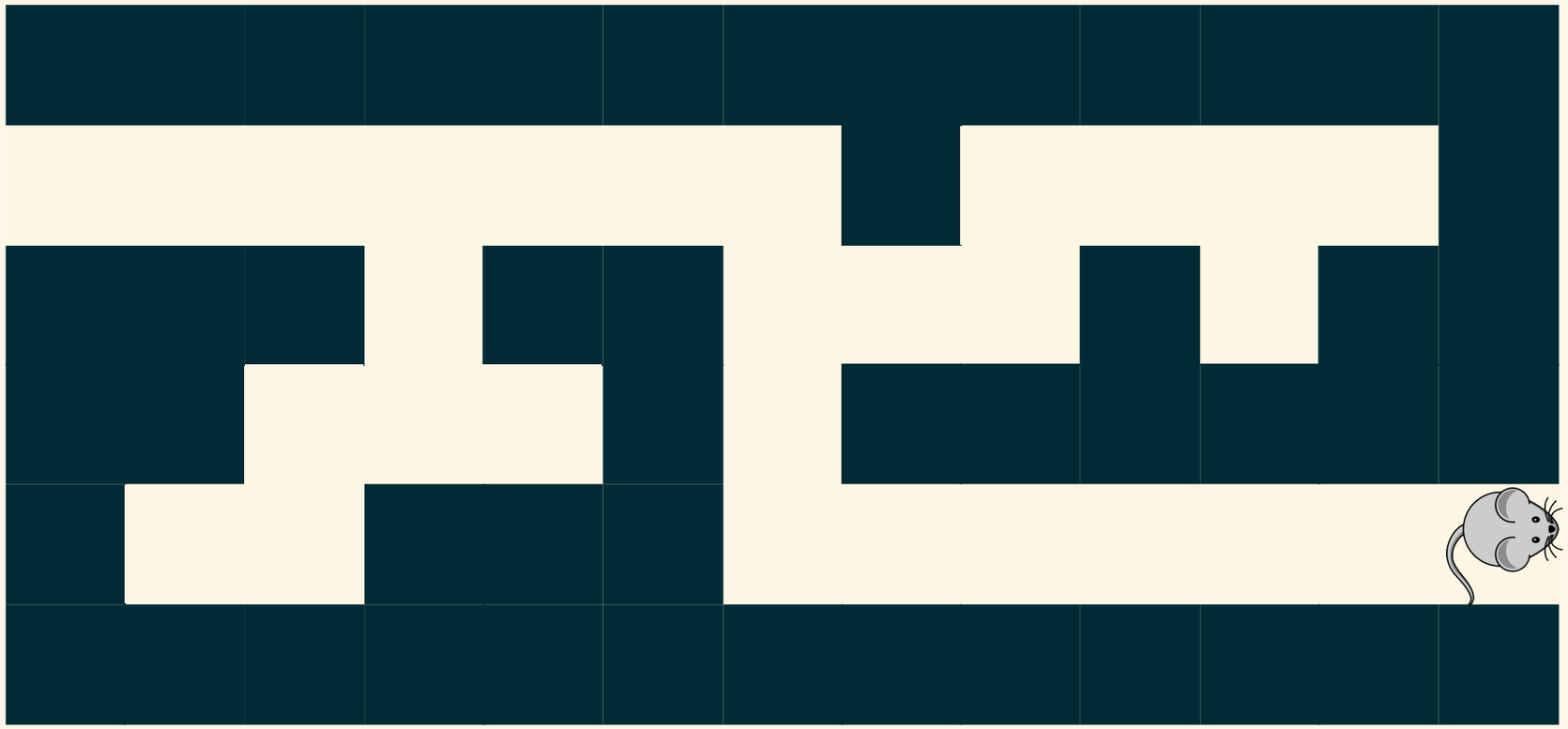
DFS	**O(V+E)**
BFS	**O(V+E)**

V: the number of nodes
E: the number of edges

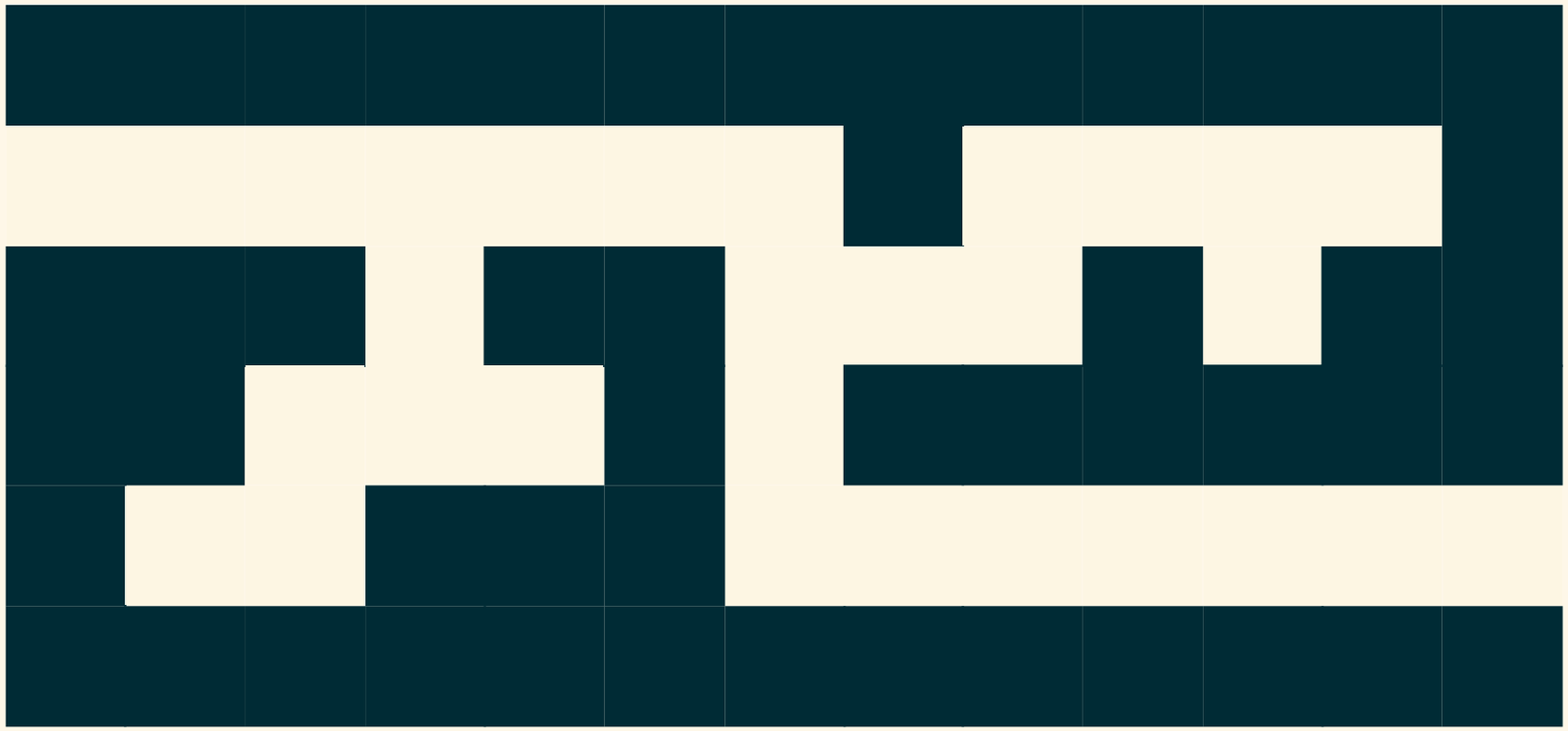
老鼠走迷宮 (DFS)



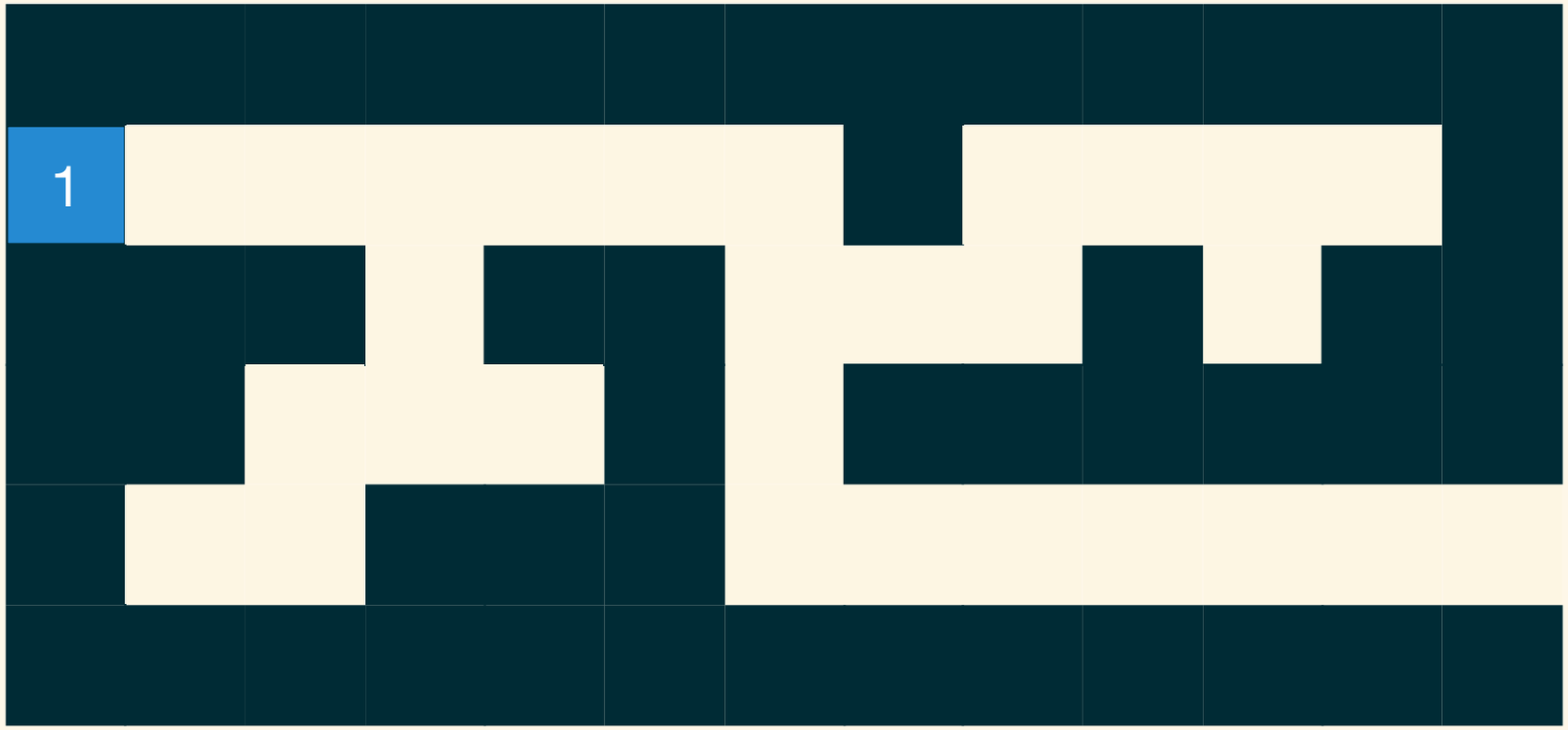
老鼠走迷宫 (DFS)



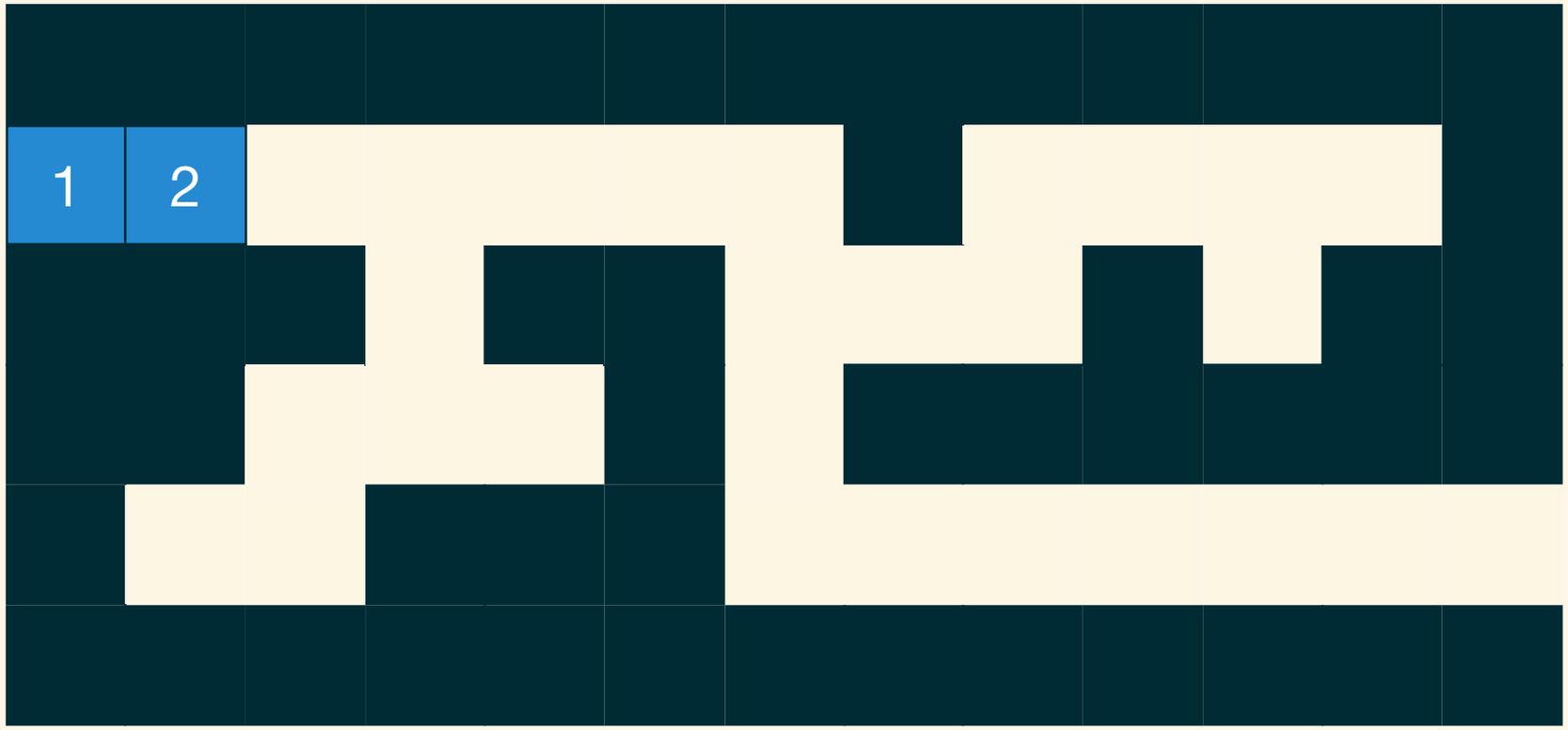
老鼠走迷宮 (BFS)



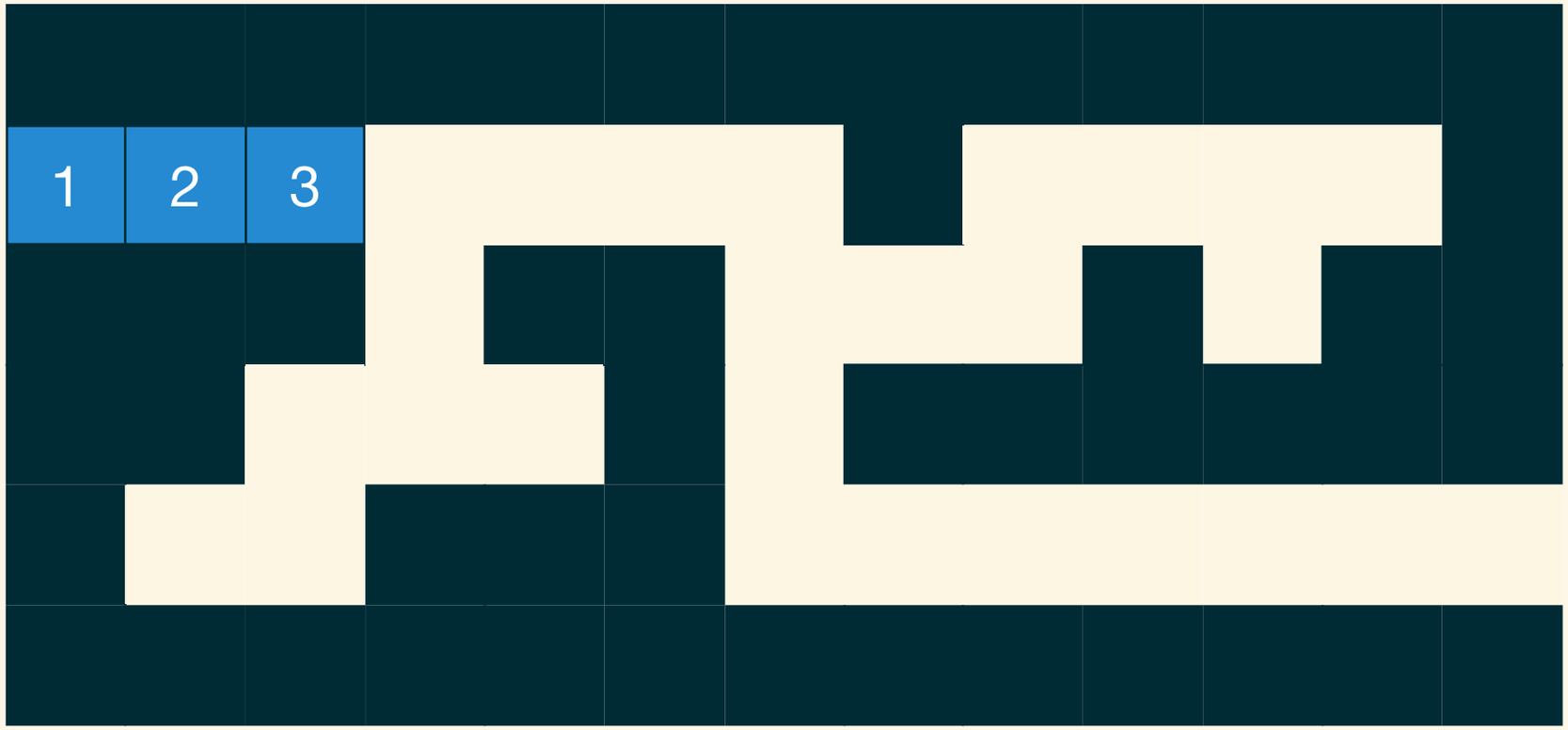
老鼠走迷宮 (BFS)



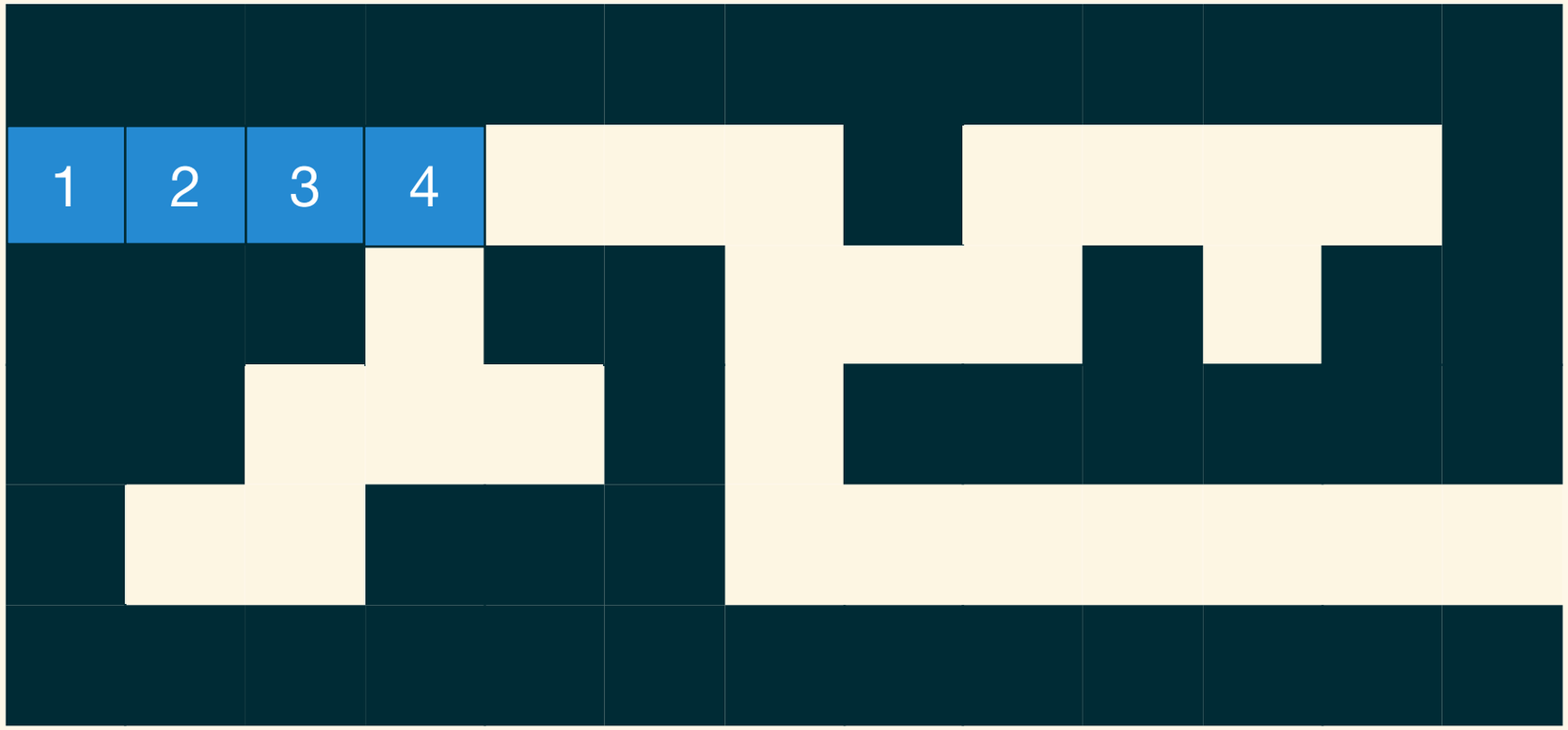
老鼠走迷宫 (BFS)



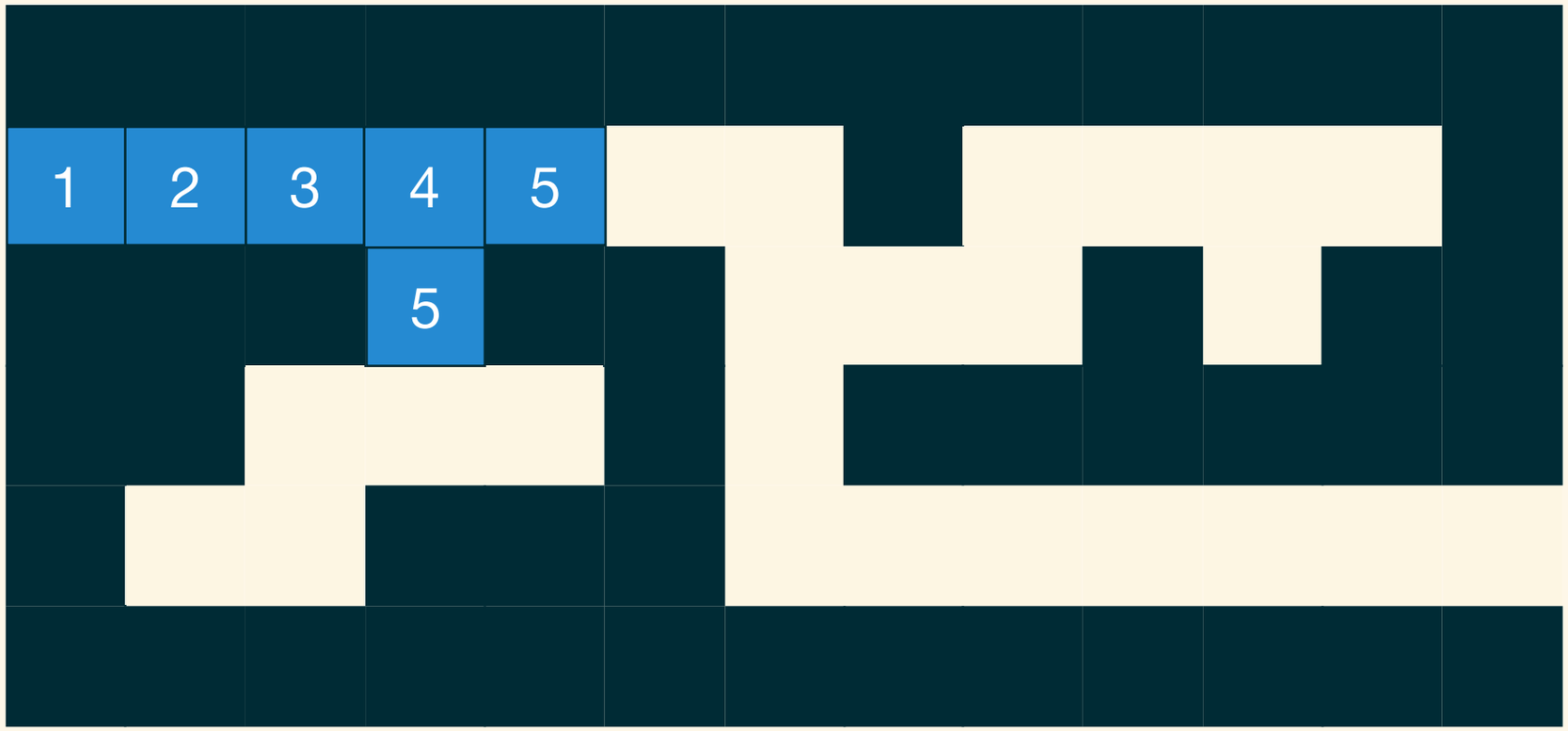
老鼠走迷宮 (BFS)



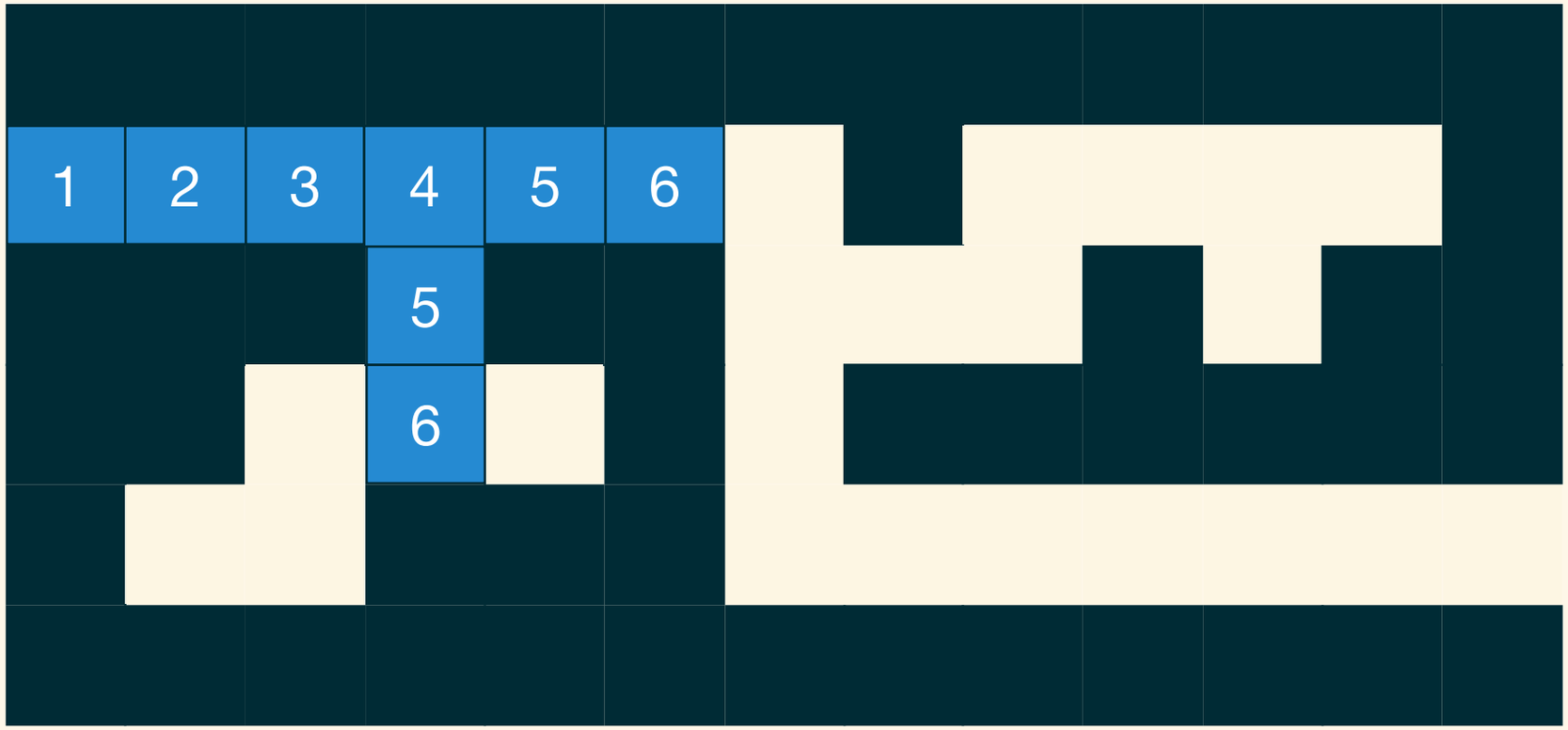
老鼠走迷宫 (BFS)



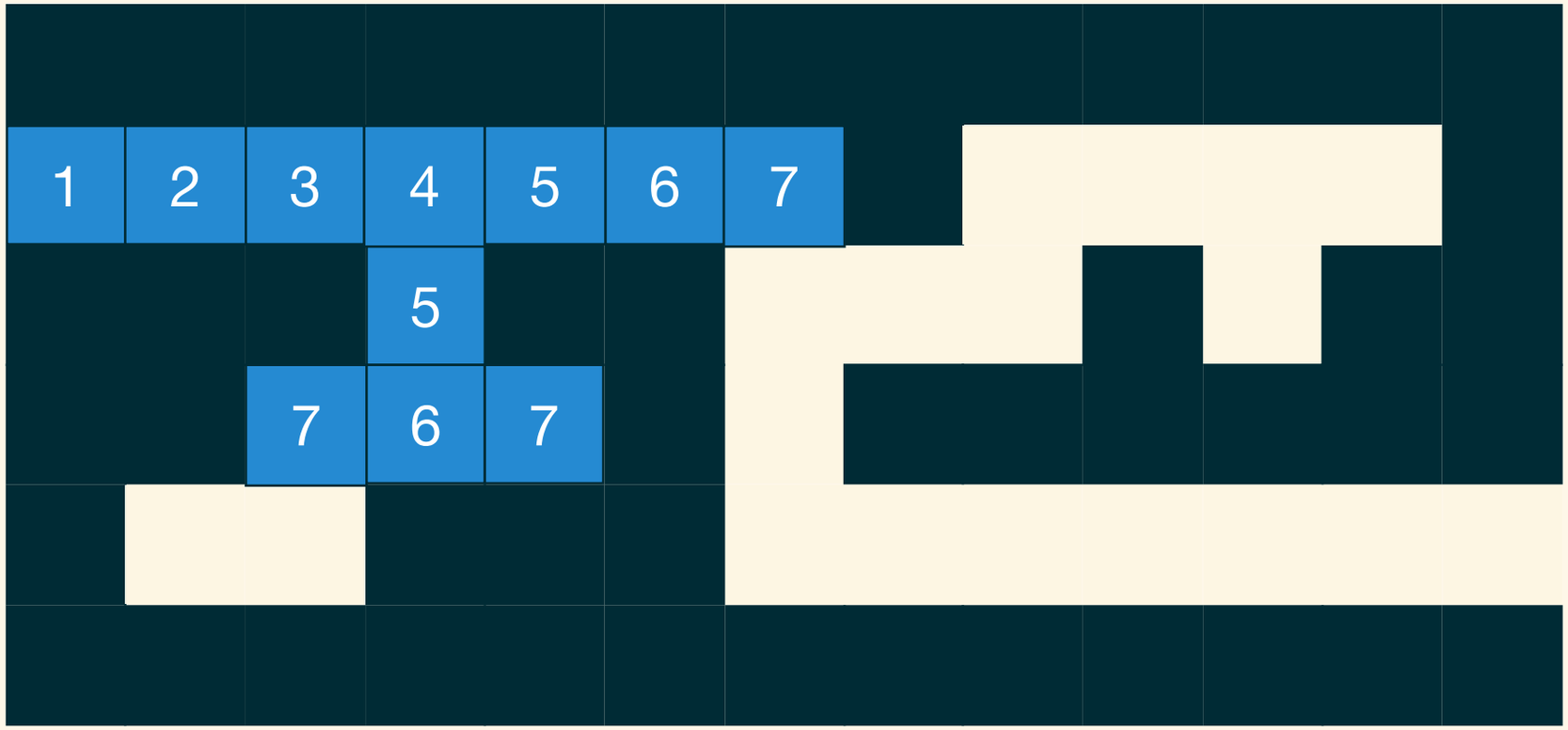
老鼠走迷宫 (BFS)



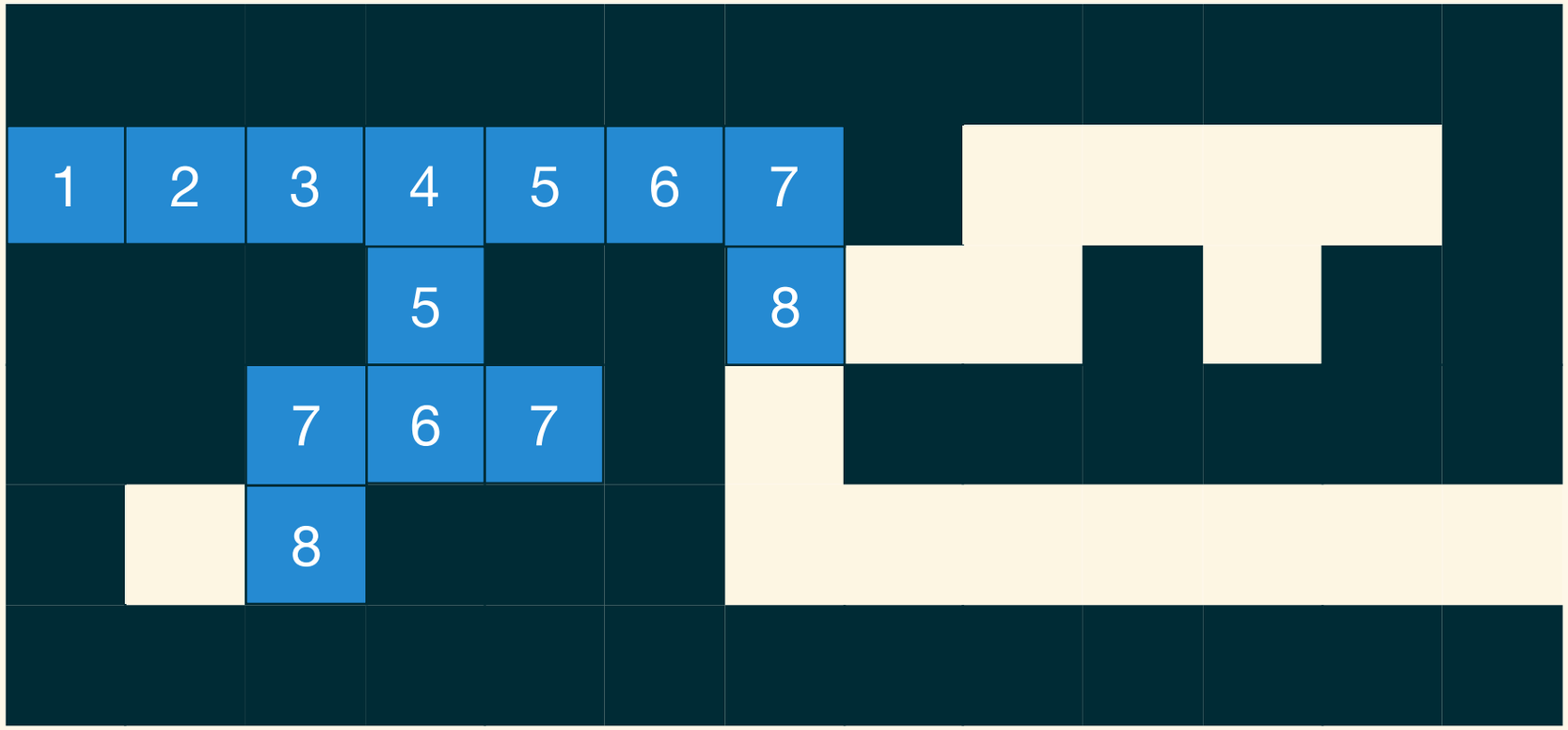
老鼠走迷宮 (BFS)



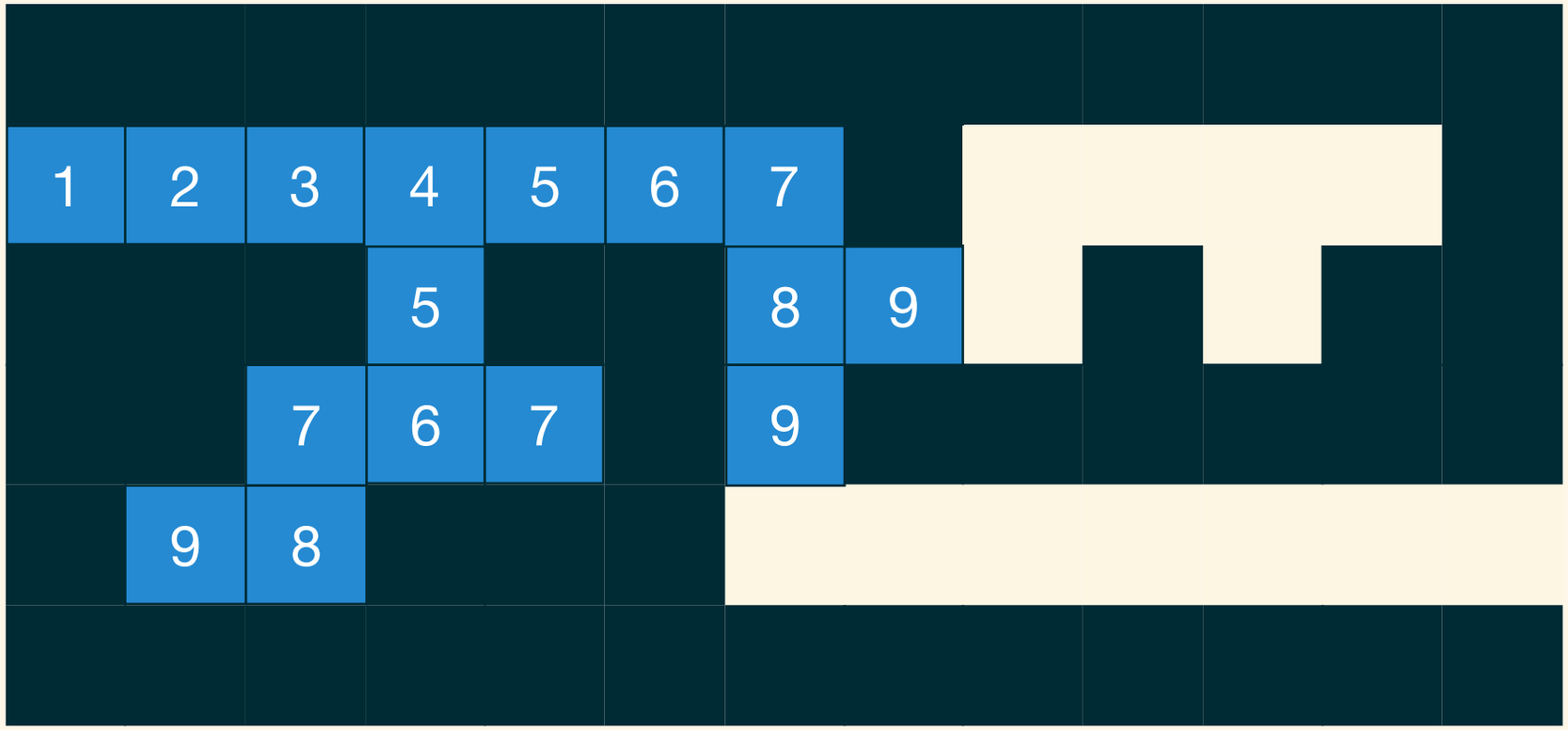
老鼠走迷宮 (BFS)



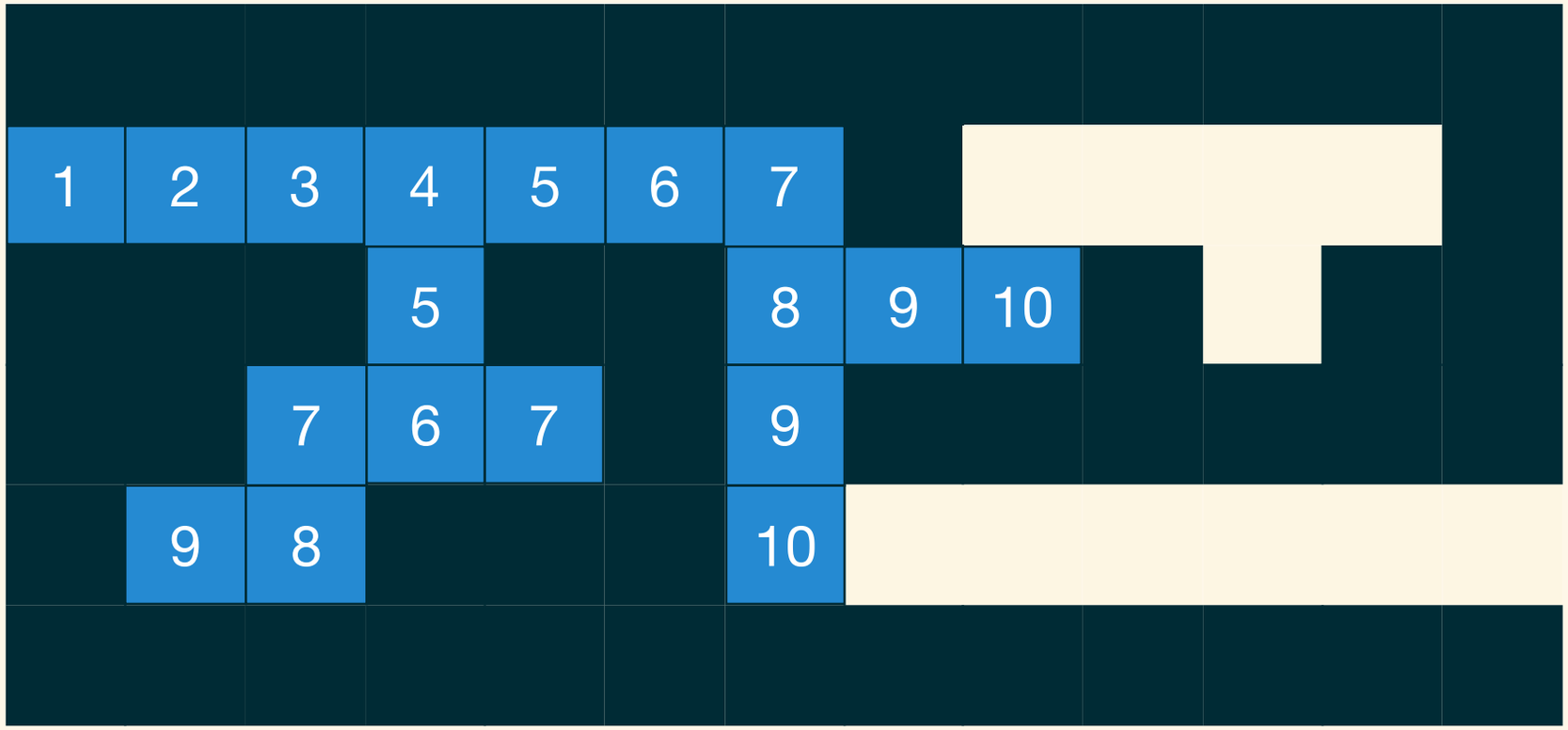
老鼠走迷宮 (BFS)



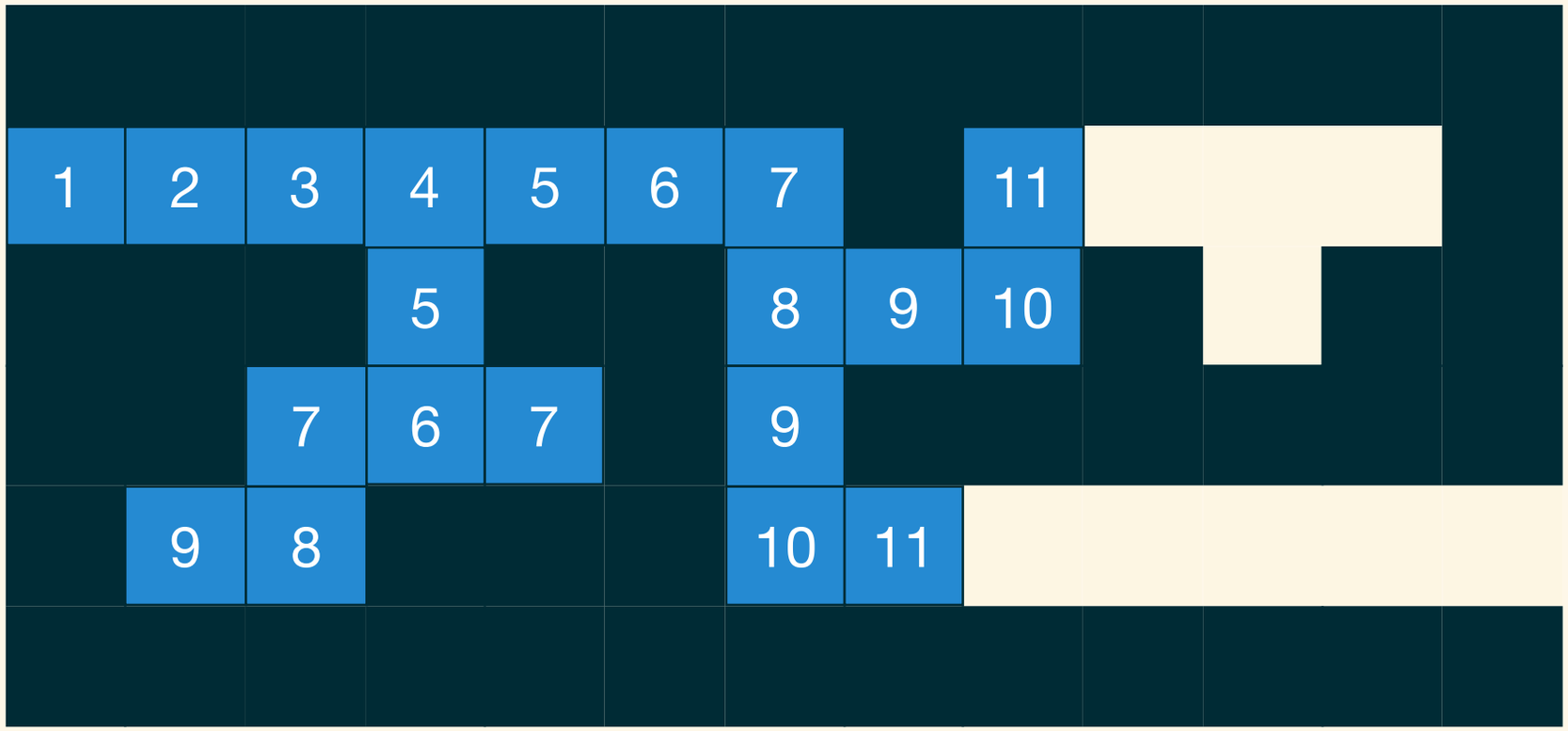
老鼠走迷宮 (BFS)



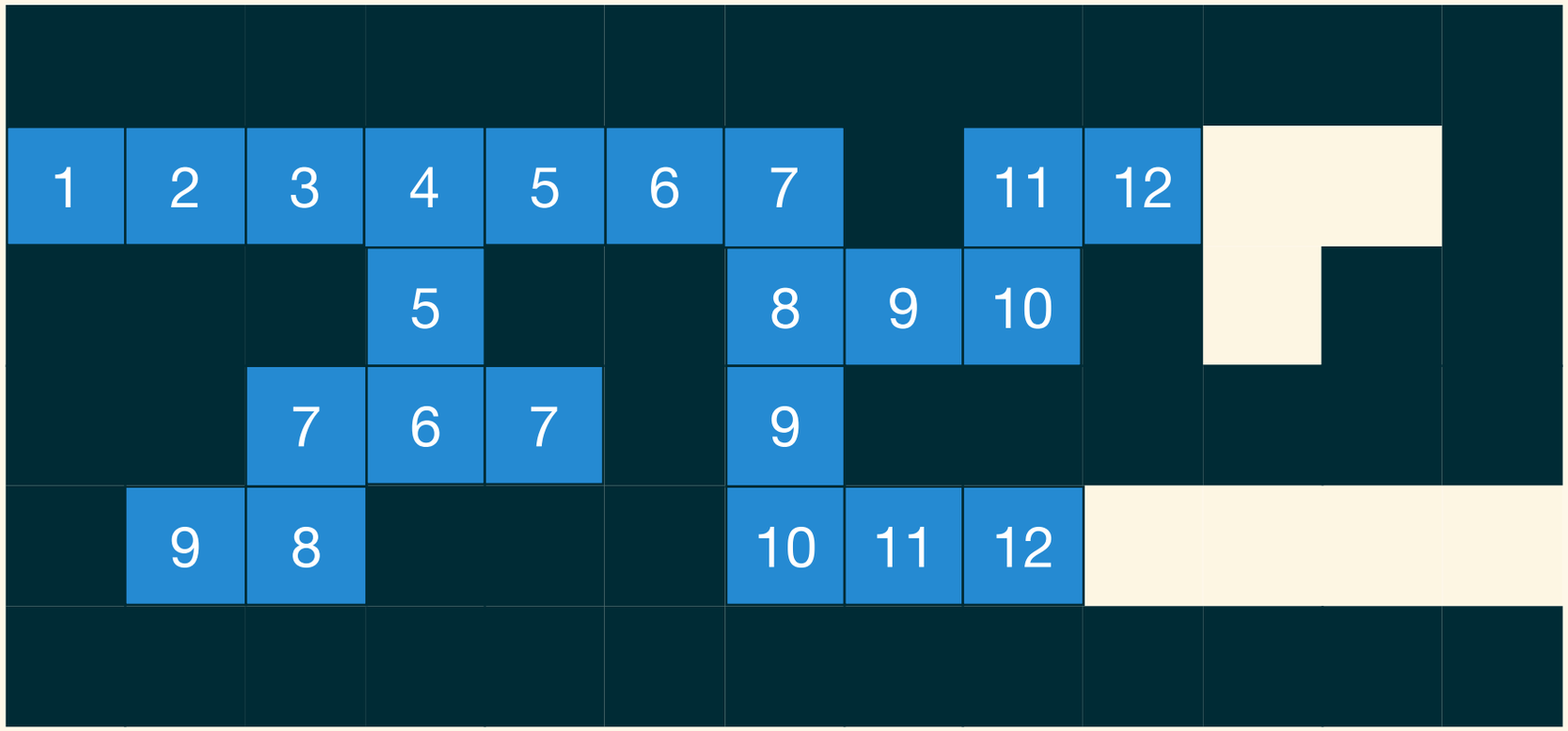
老鼠走迷宫 (BFS)



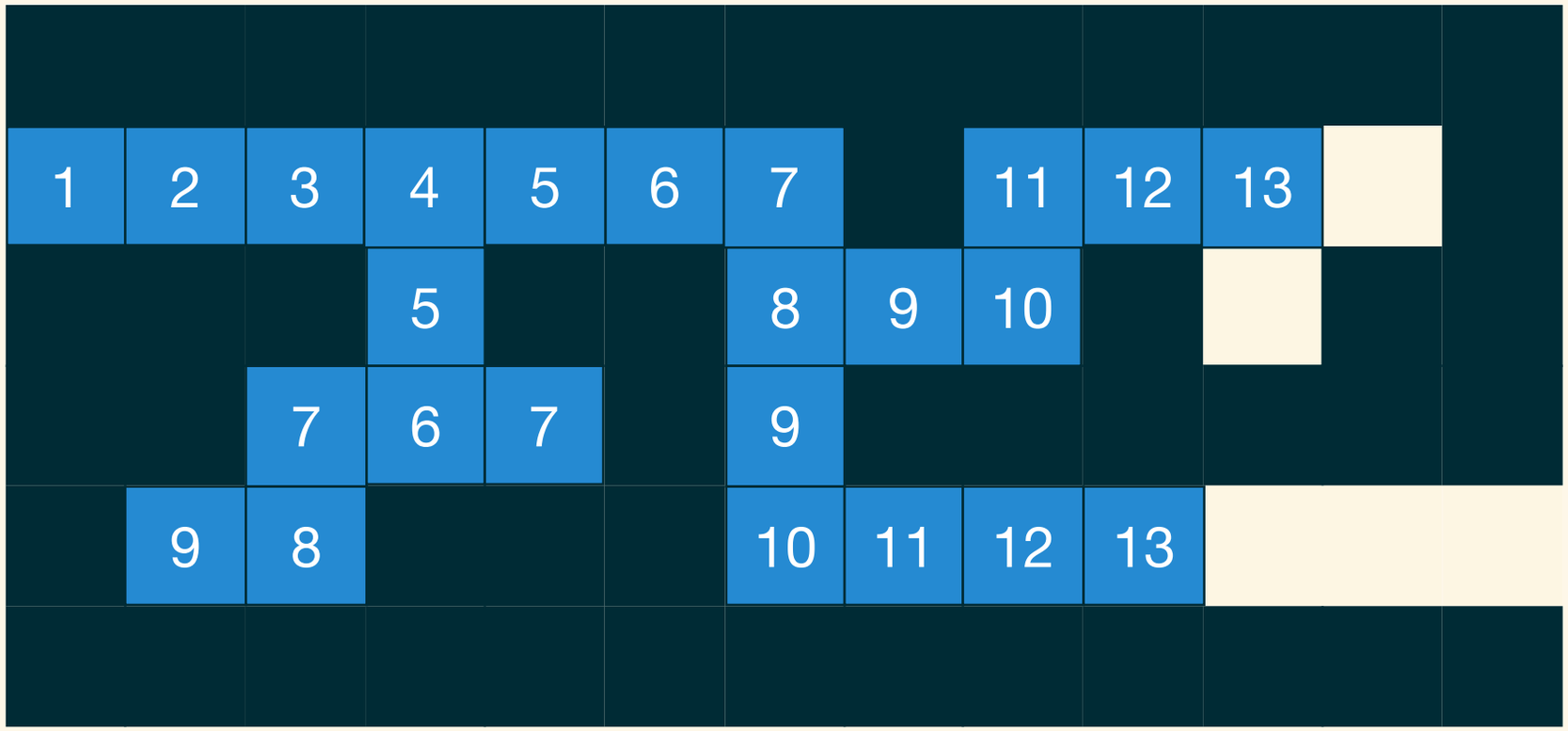
老鼠走迷宫 (BFS)



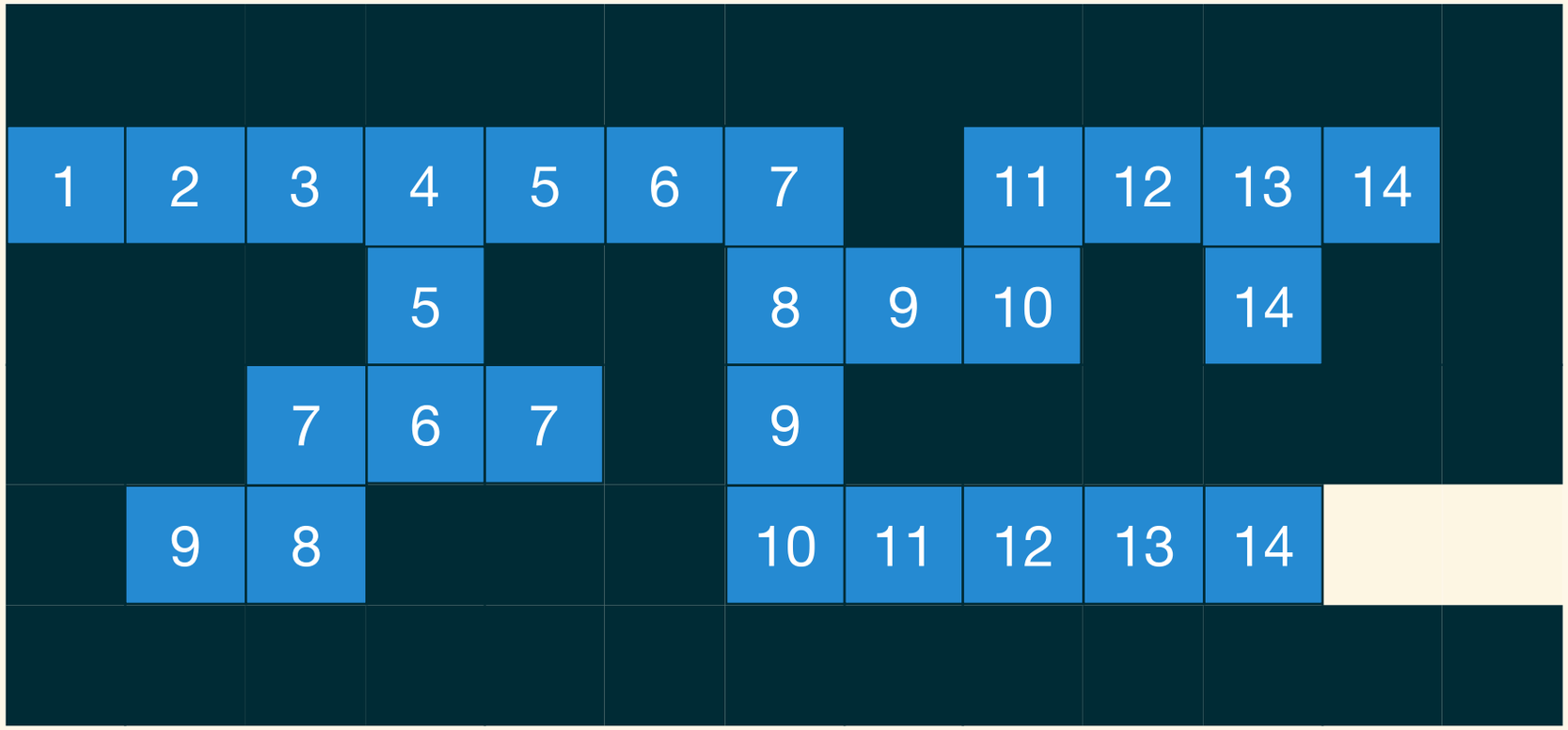
老鼠走迷宫 (BFS)



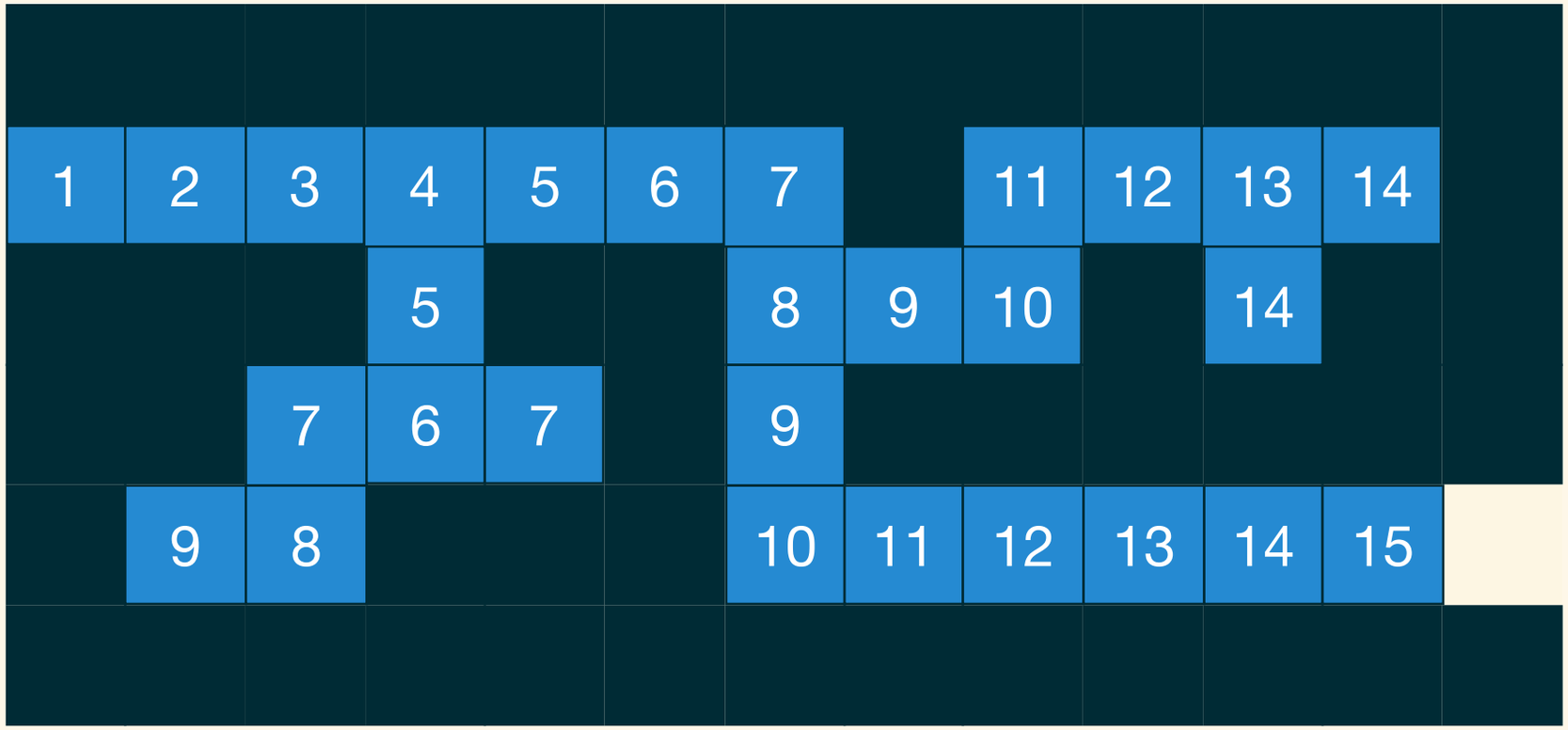
老鼠走迷宮 (BFS)



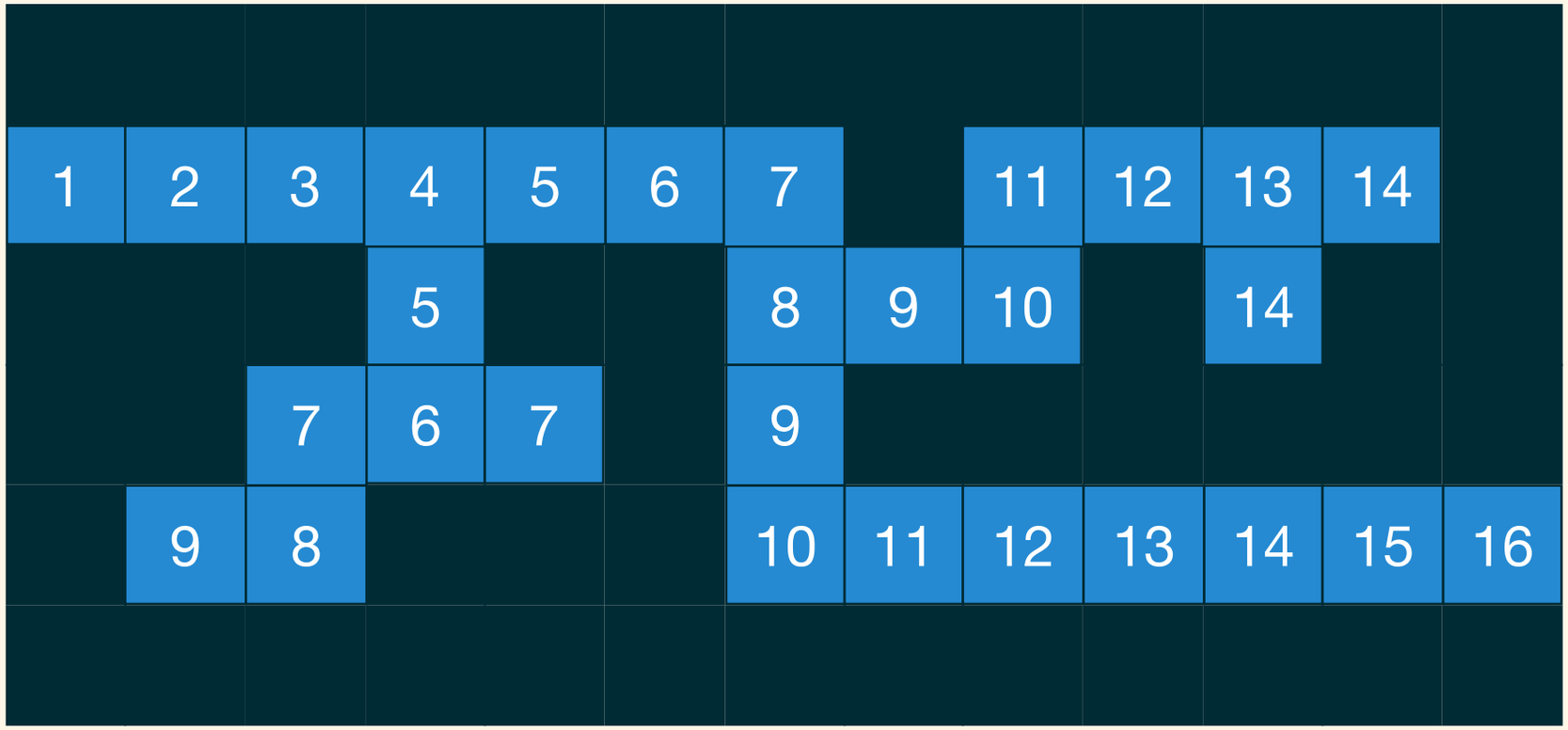
老鼠走迷宫 (BFS)



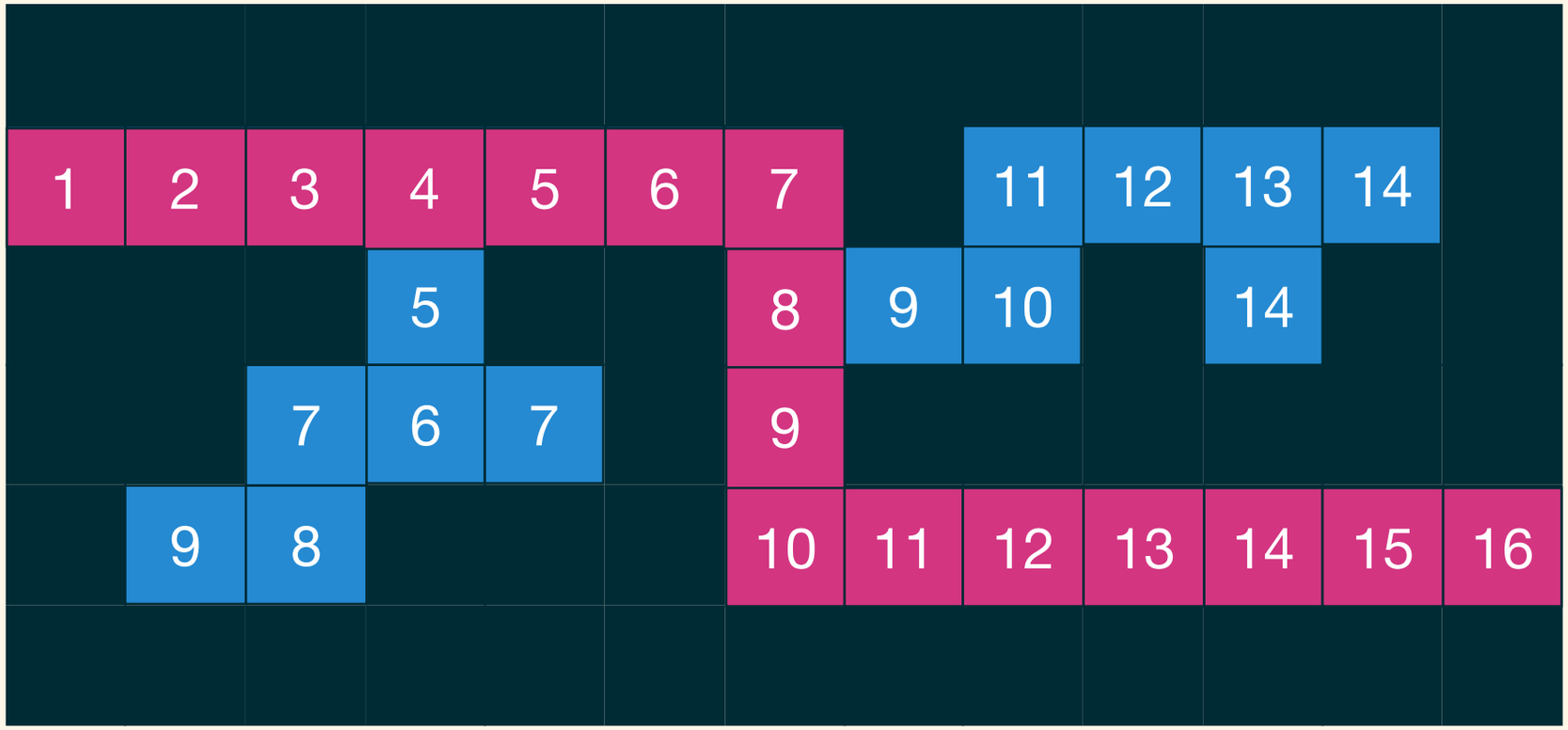
老鼠走迷宫 (BFS)



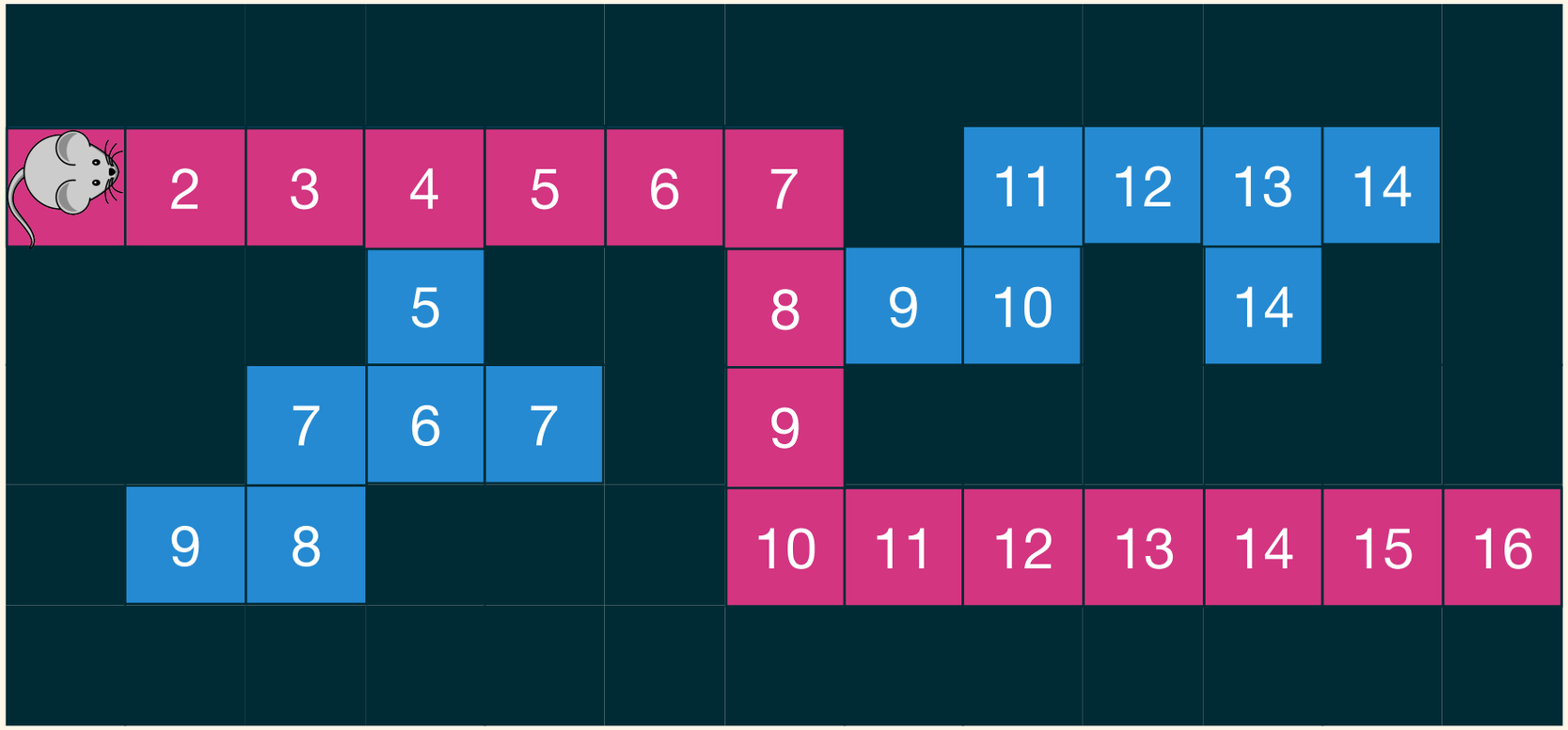
老鼠走迷宫 (BFS)



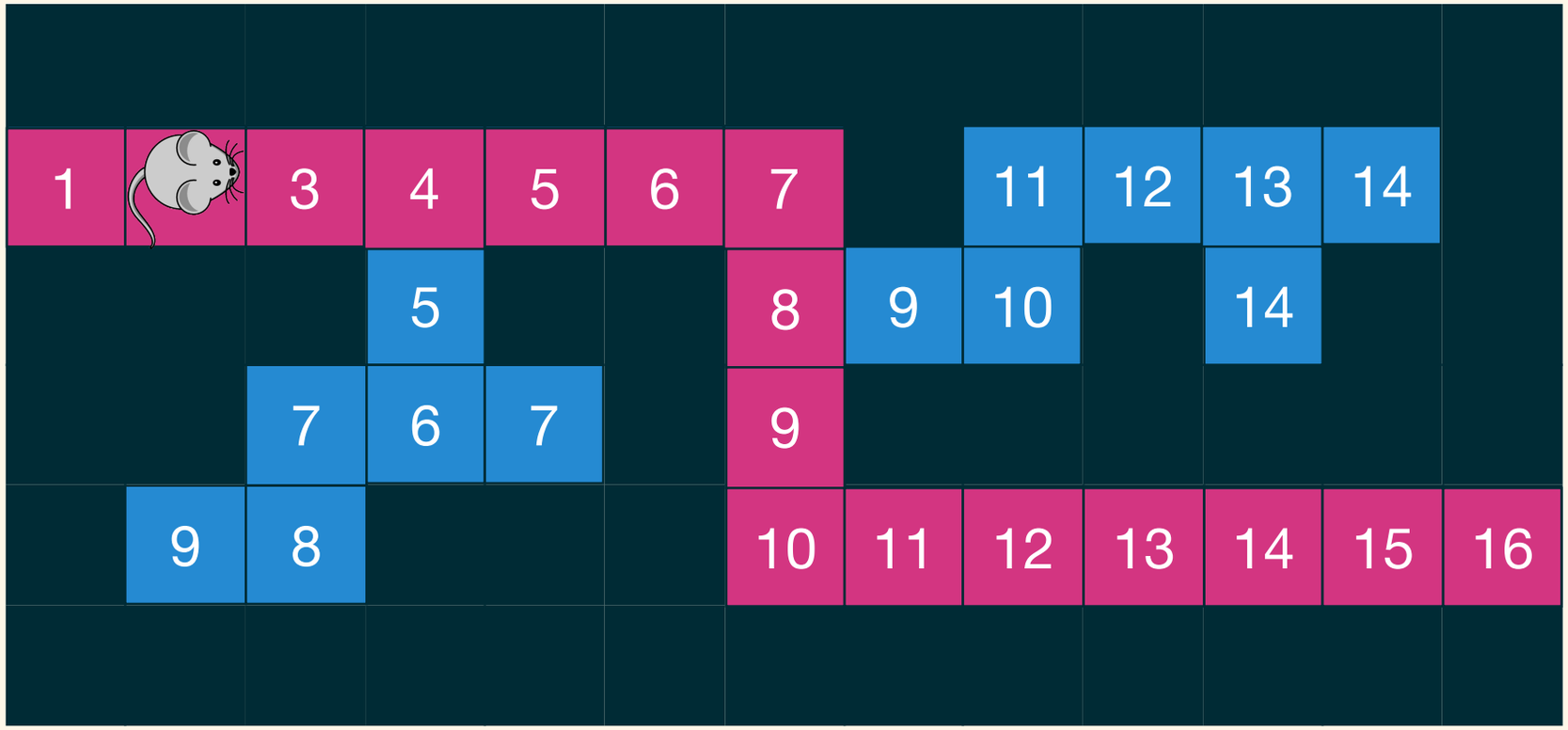
老鼠走迷宫 (BFS)



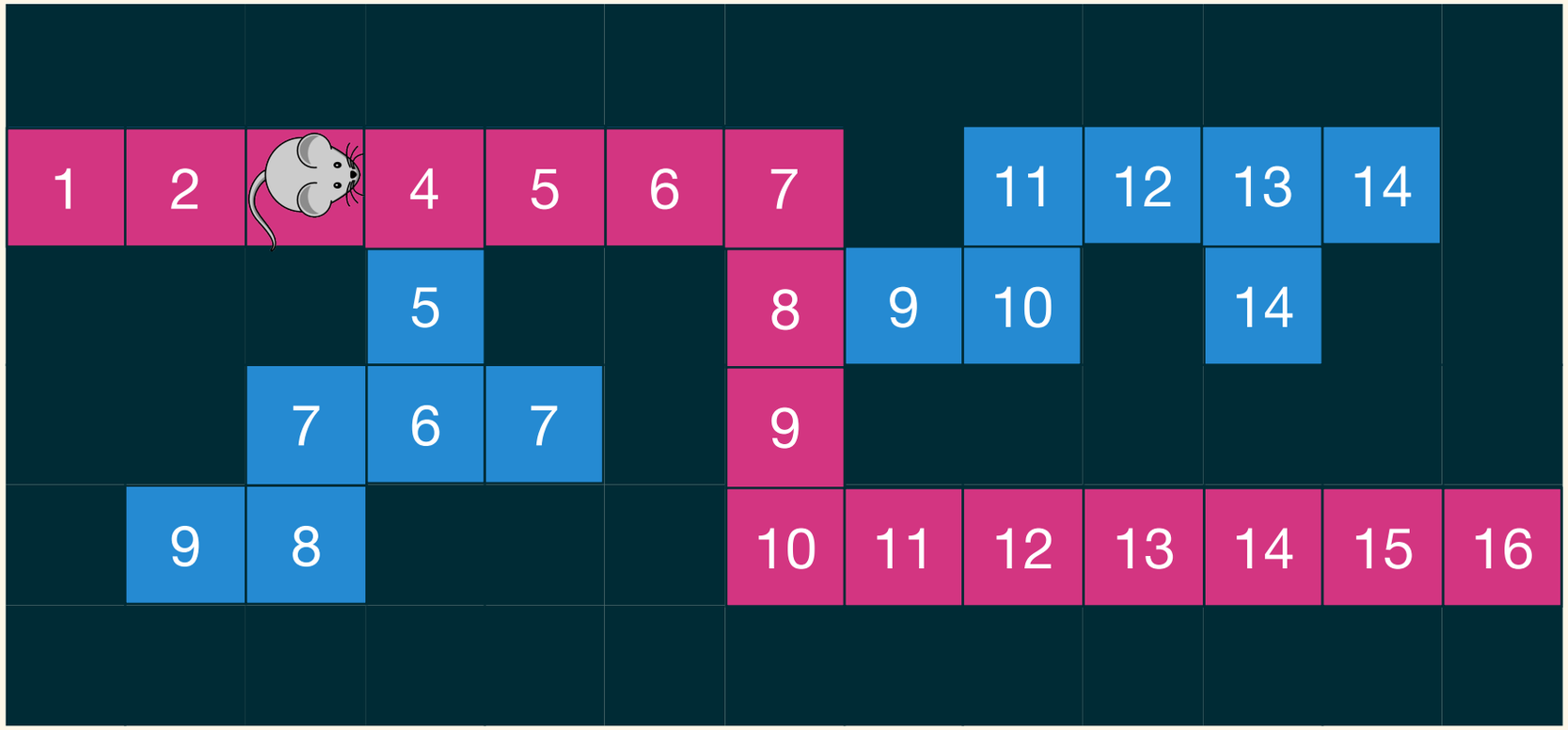
老鼠走迷宮 (BFS)



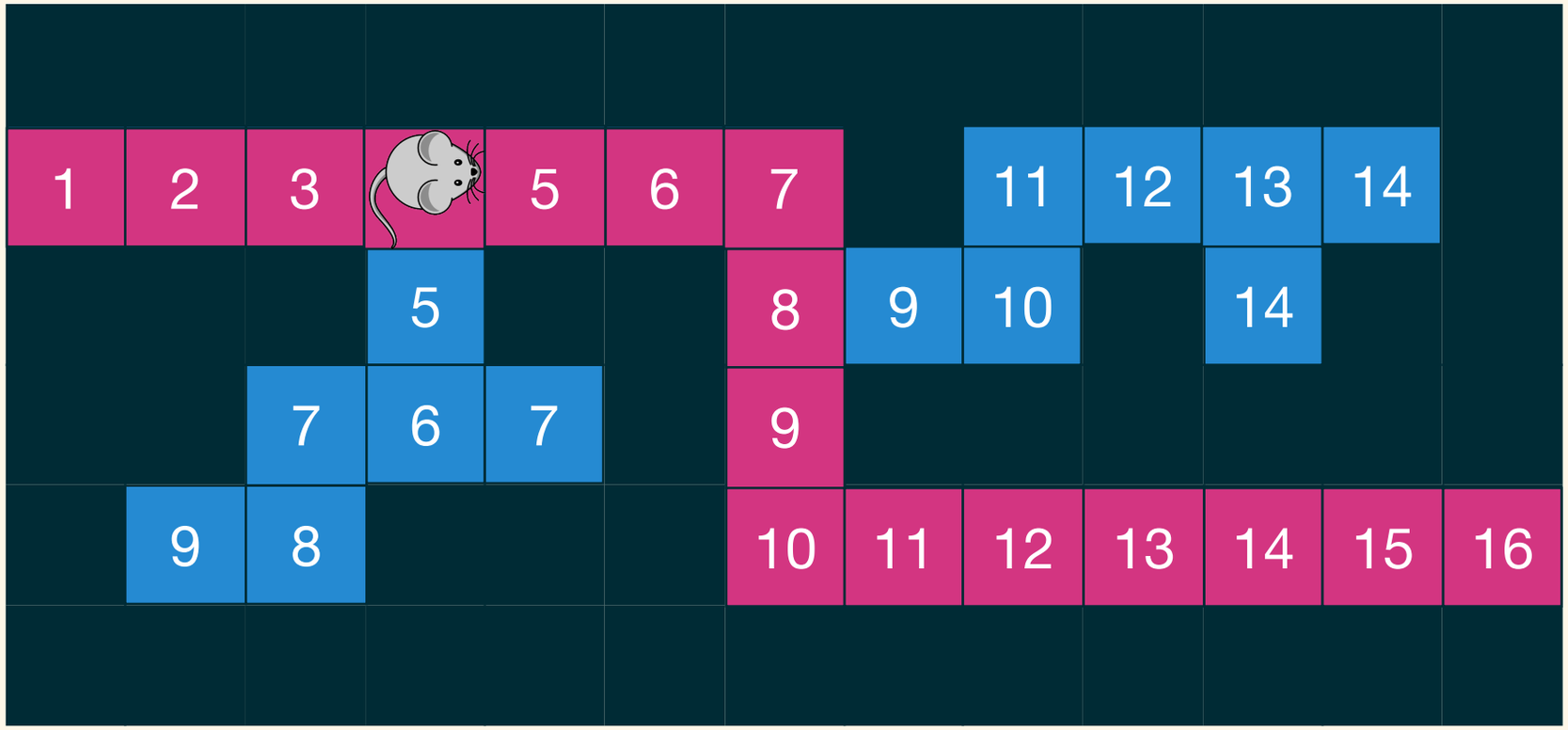
老鼠走迷宮 (BFS)



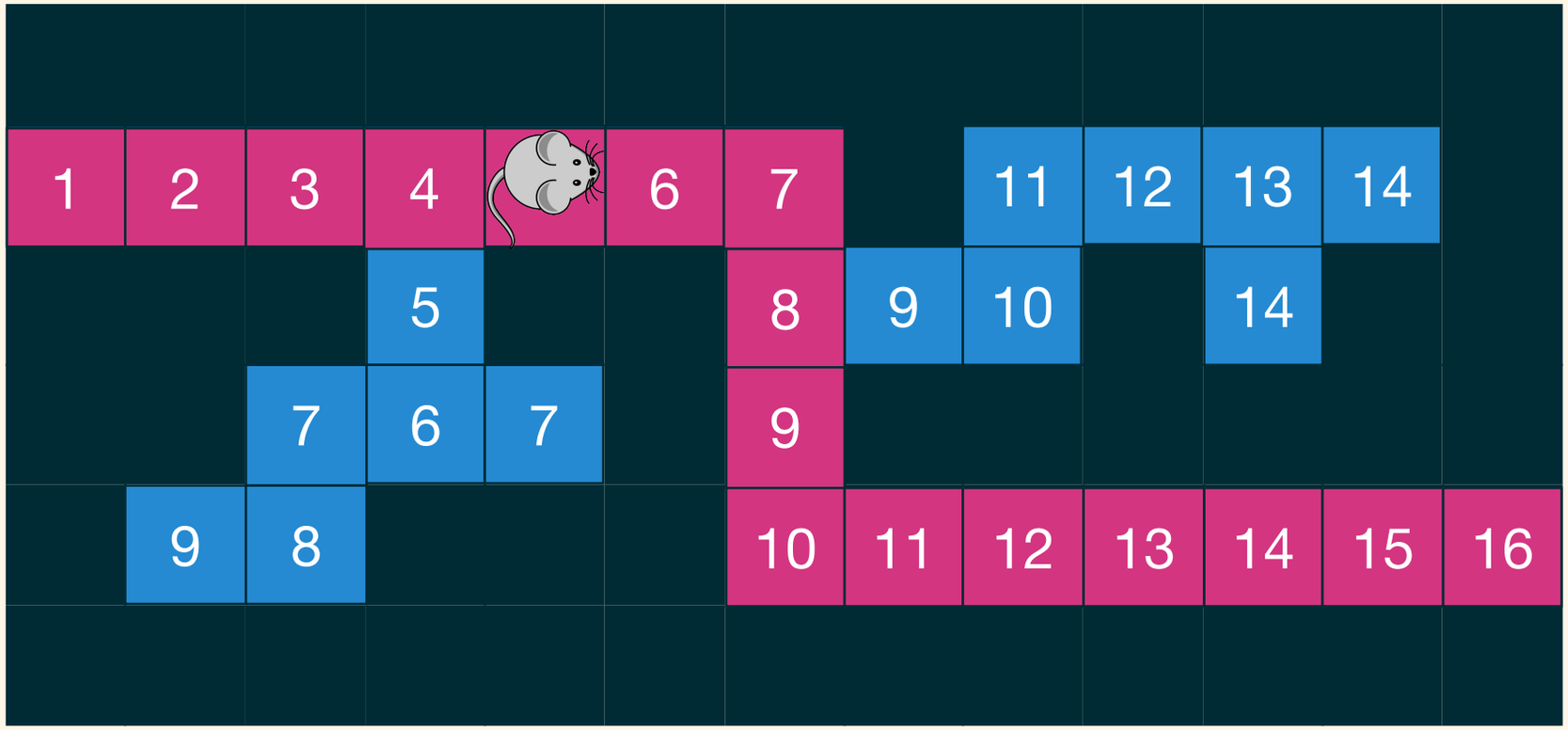
老鼠走迷宮 (BFS)



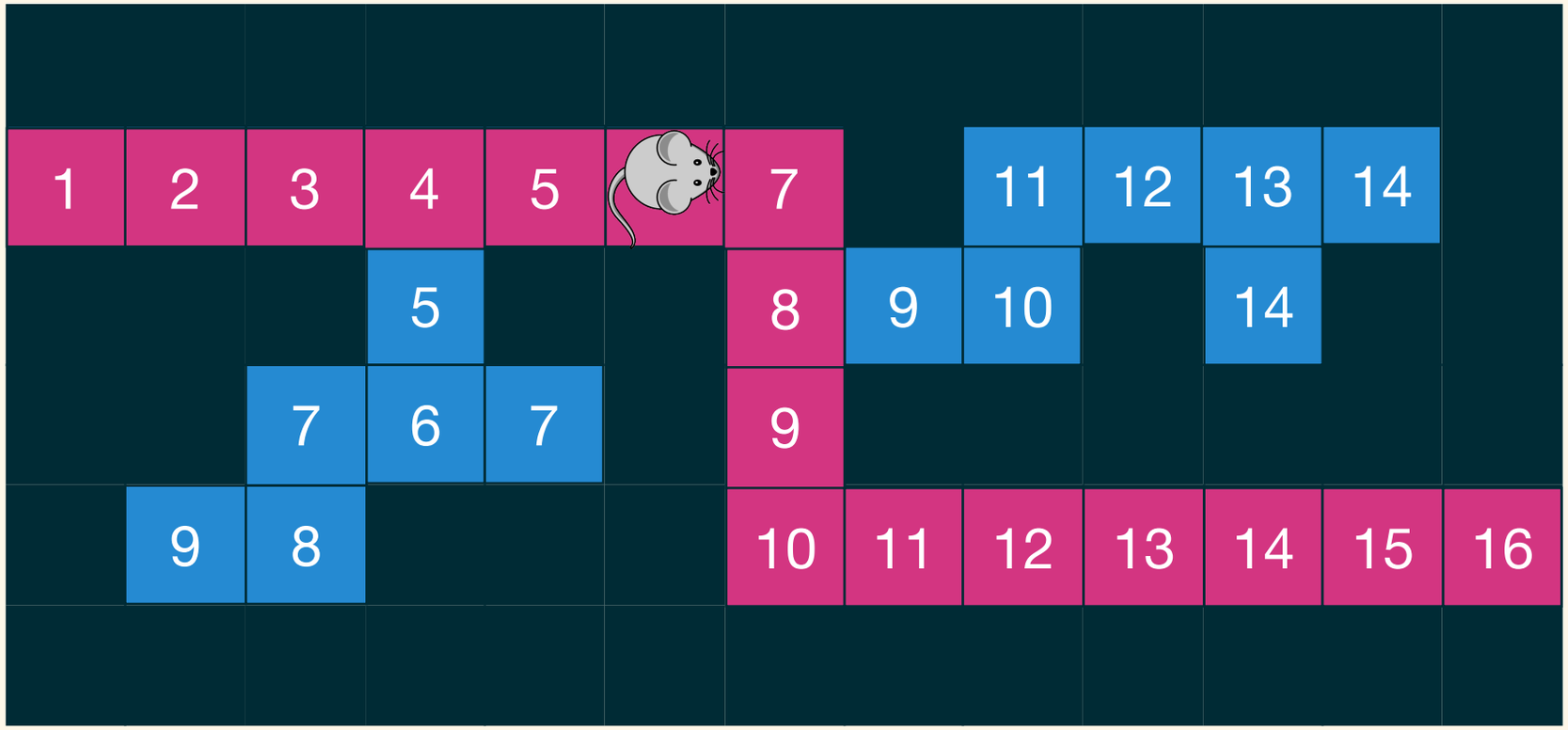
老鼠走迷宮 (BFS)



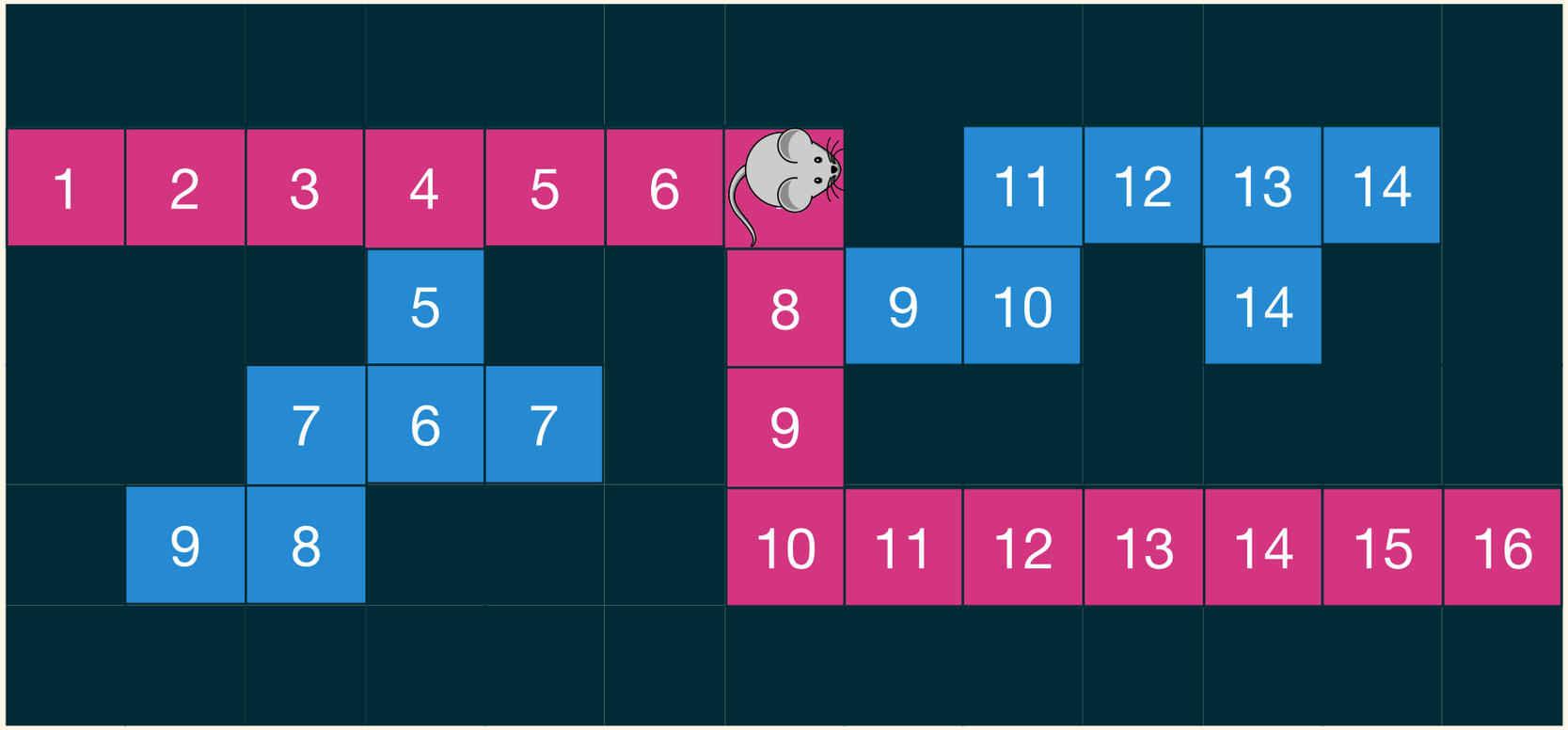
老鼠走迷宮 (BFS)



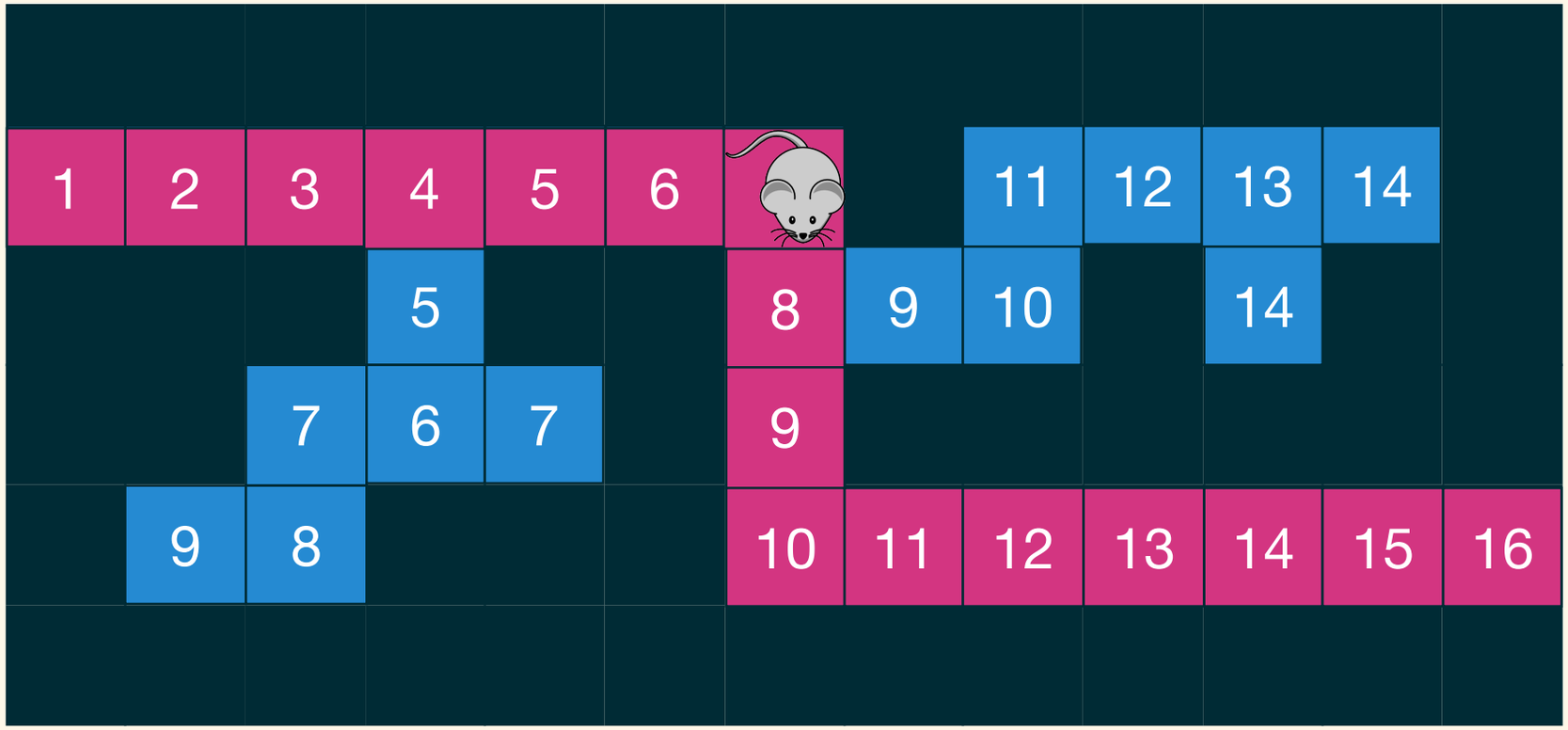
老鼠走迷宮 (BFS)



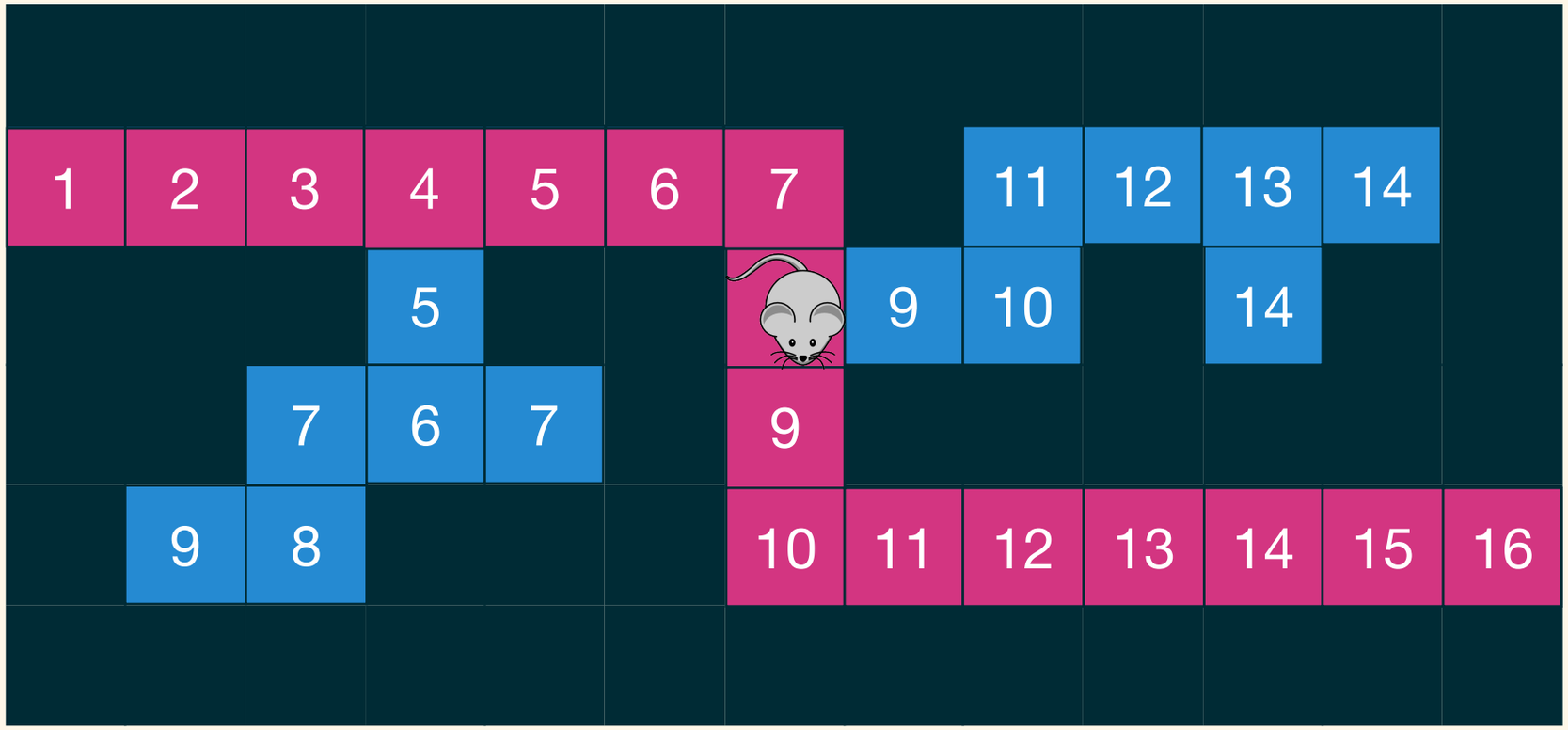
老鼠走迷宫 (BFS)



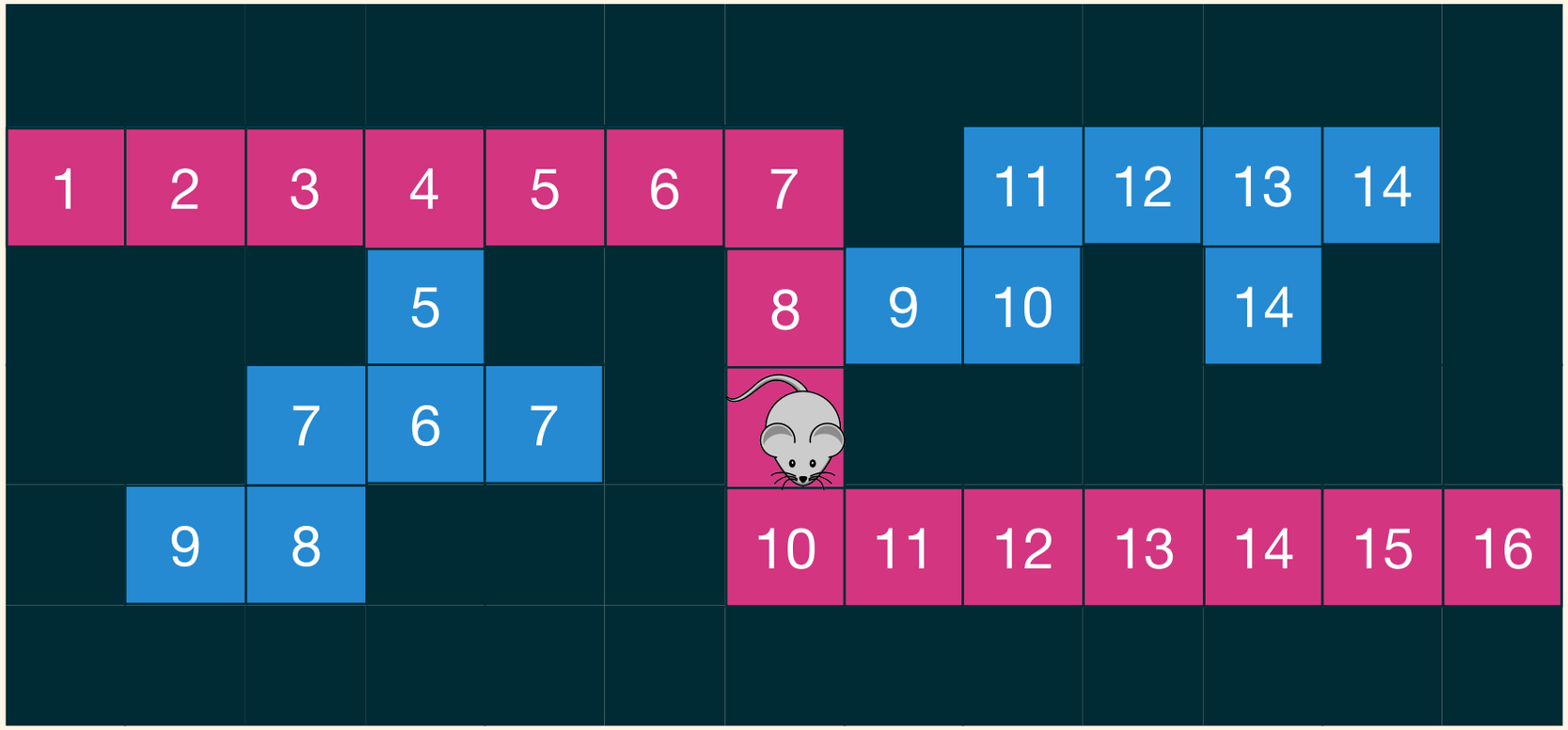
老鼠走迷宮 (BFS)



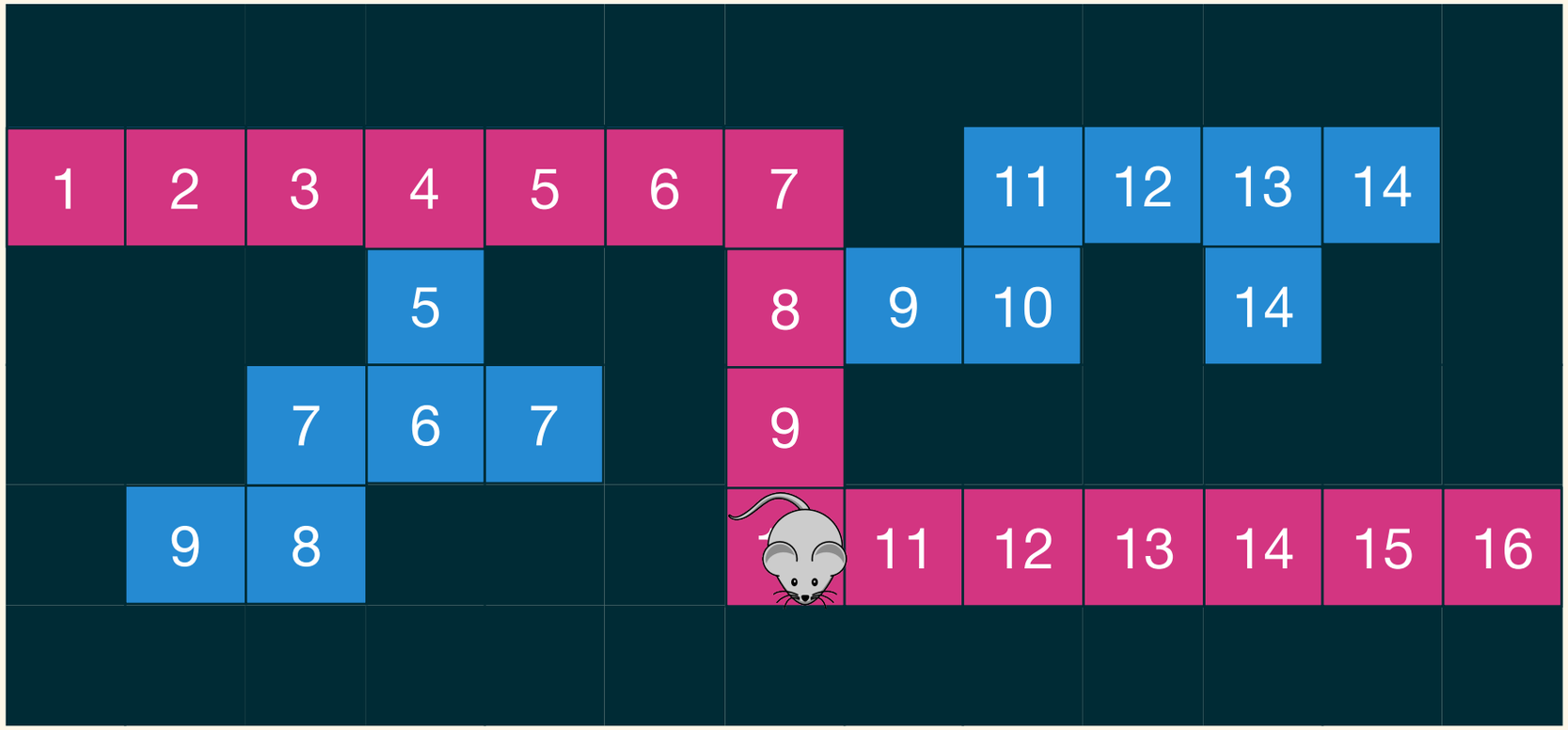
老鼠走迷宮 (BFS)



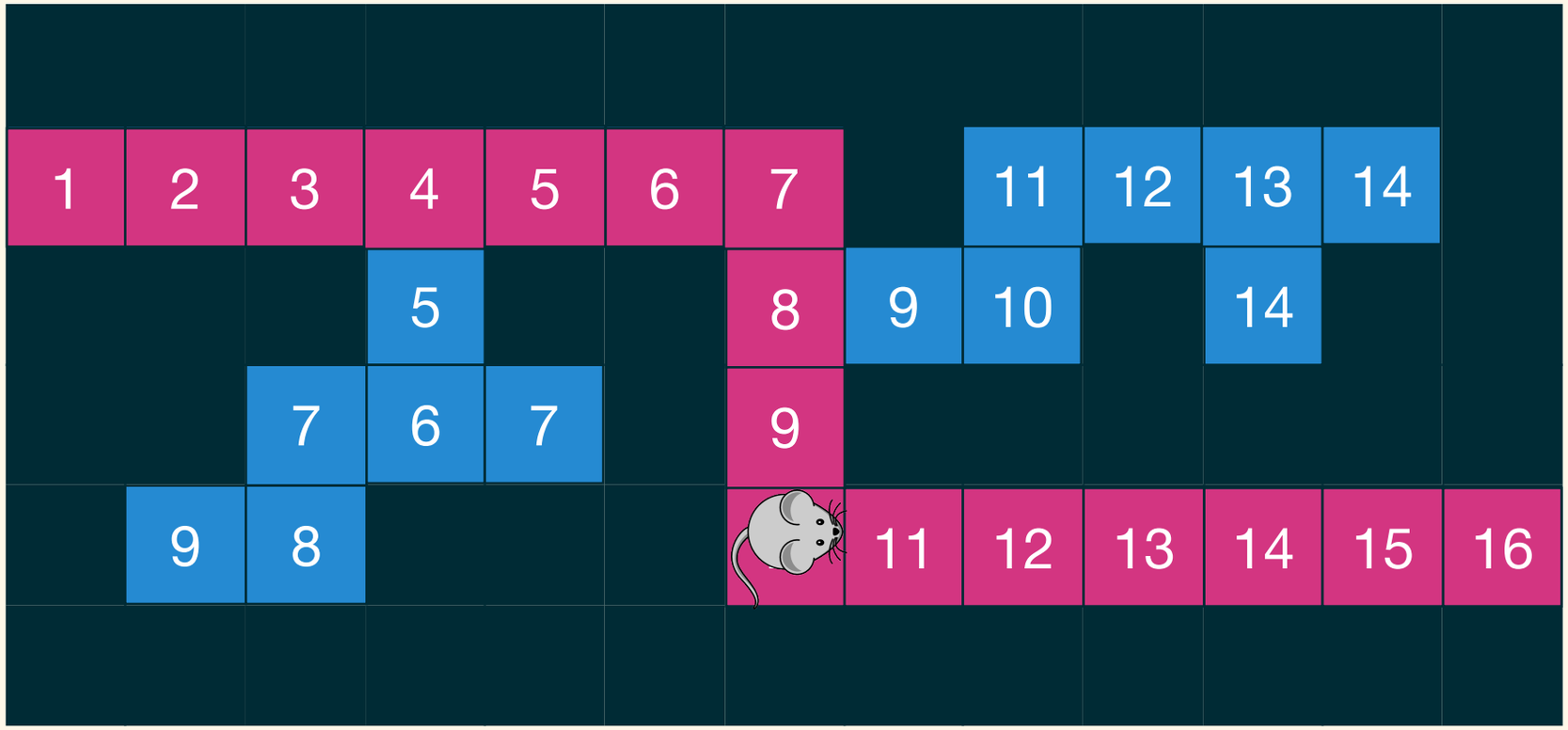
老鼠走迷宫 (BFS)



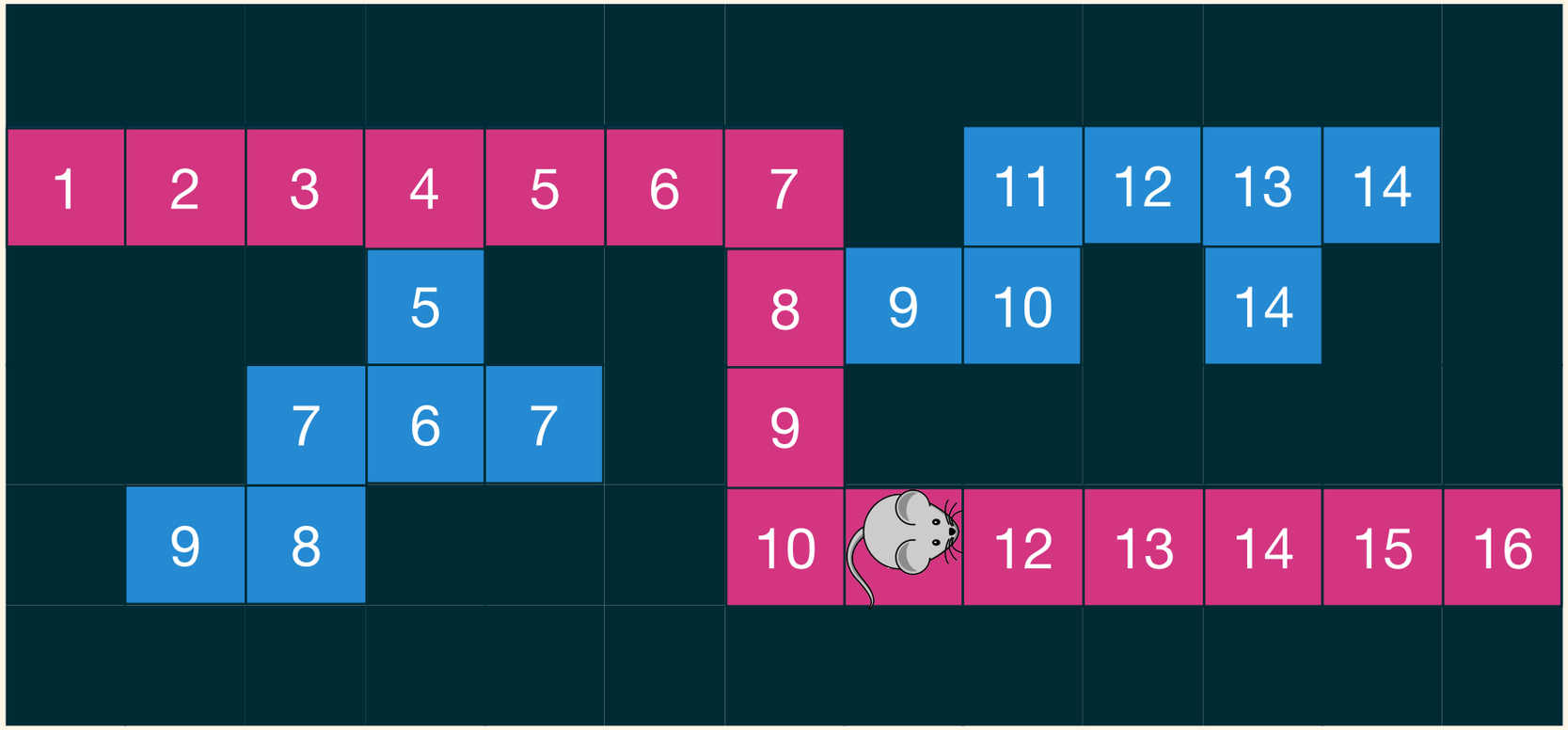
老鼠走迷宫 (BFS)



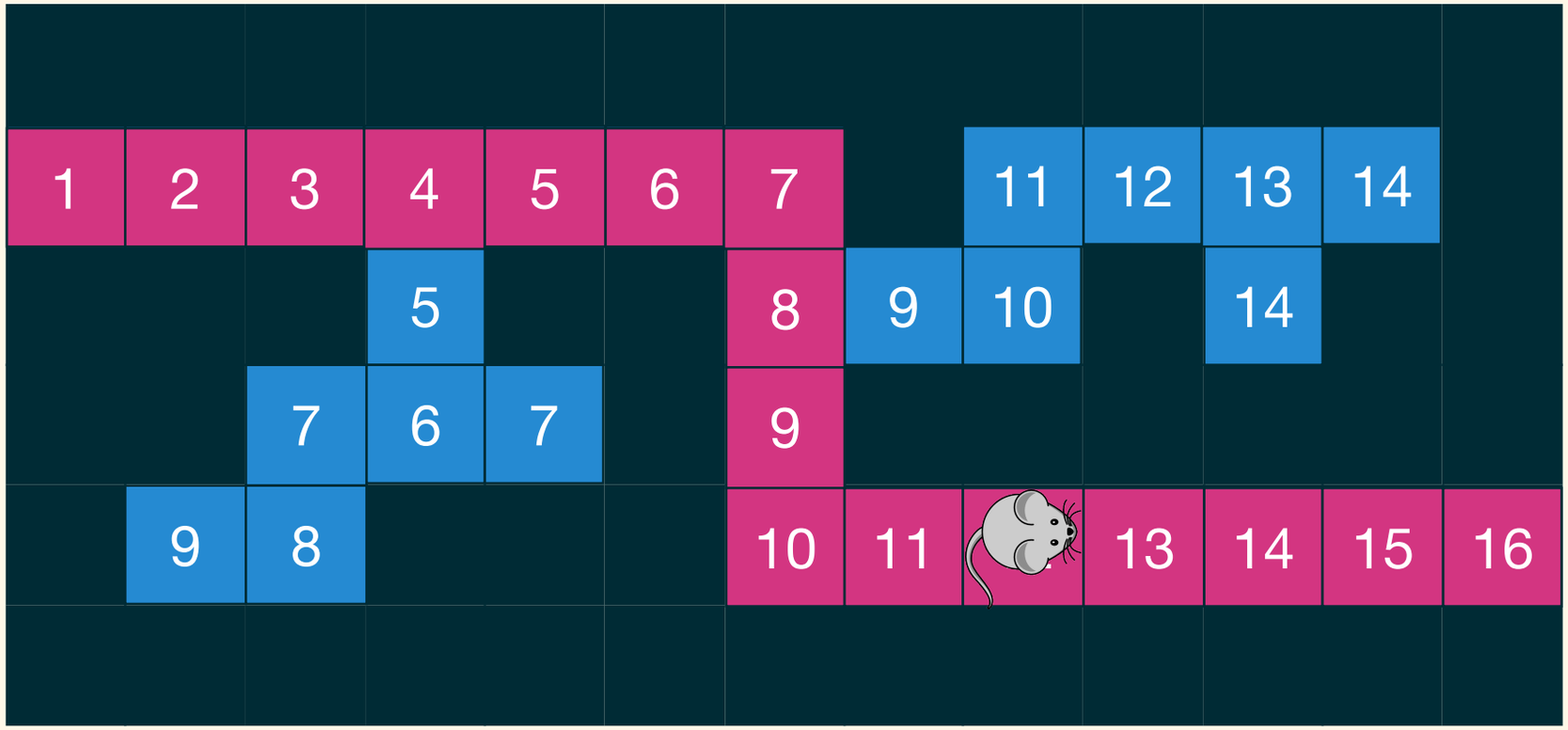
老鼠走迷宮 (BFS)



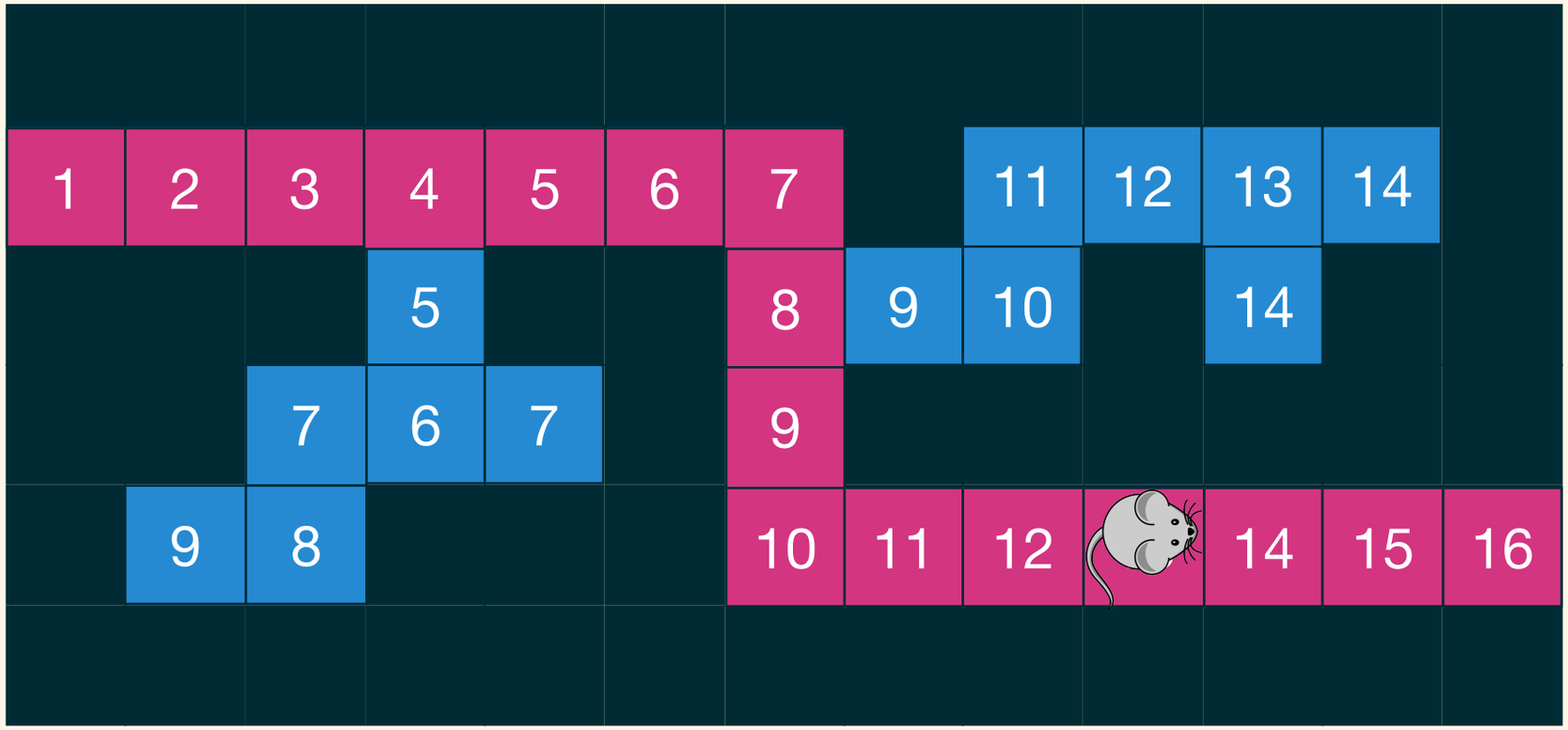
老鼠走迷宮 (BFS)



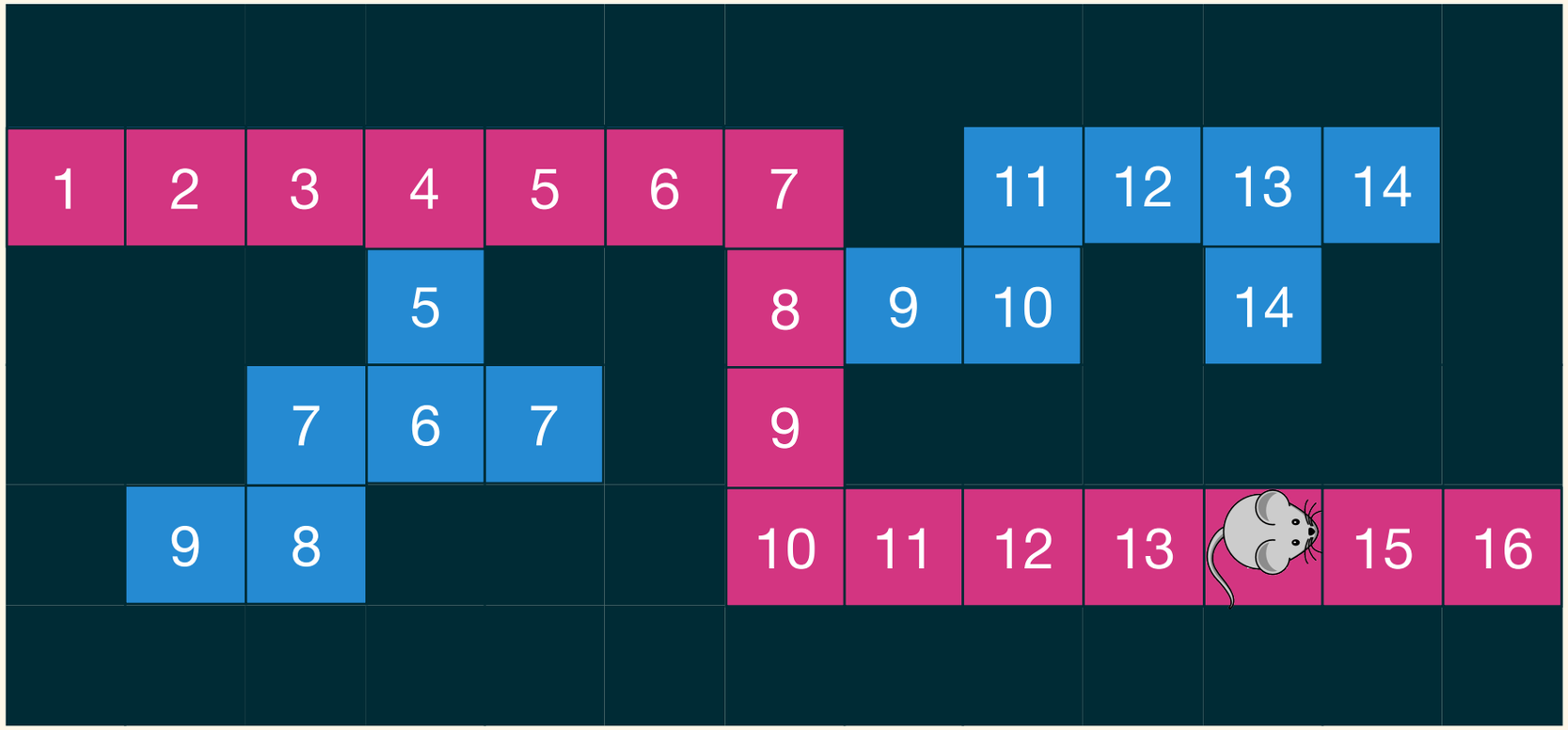
老鼠走迷宫 (BFS)



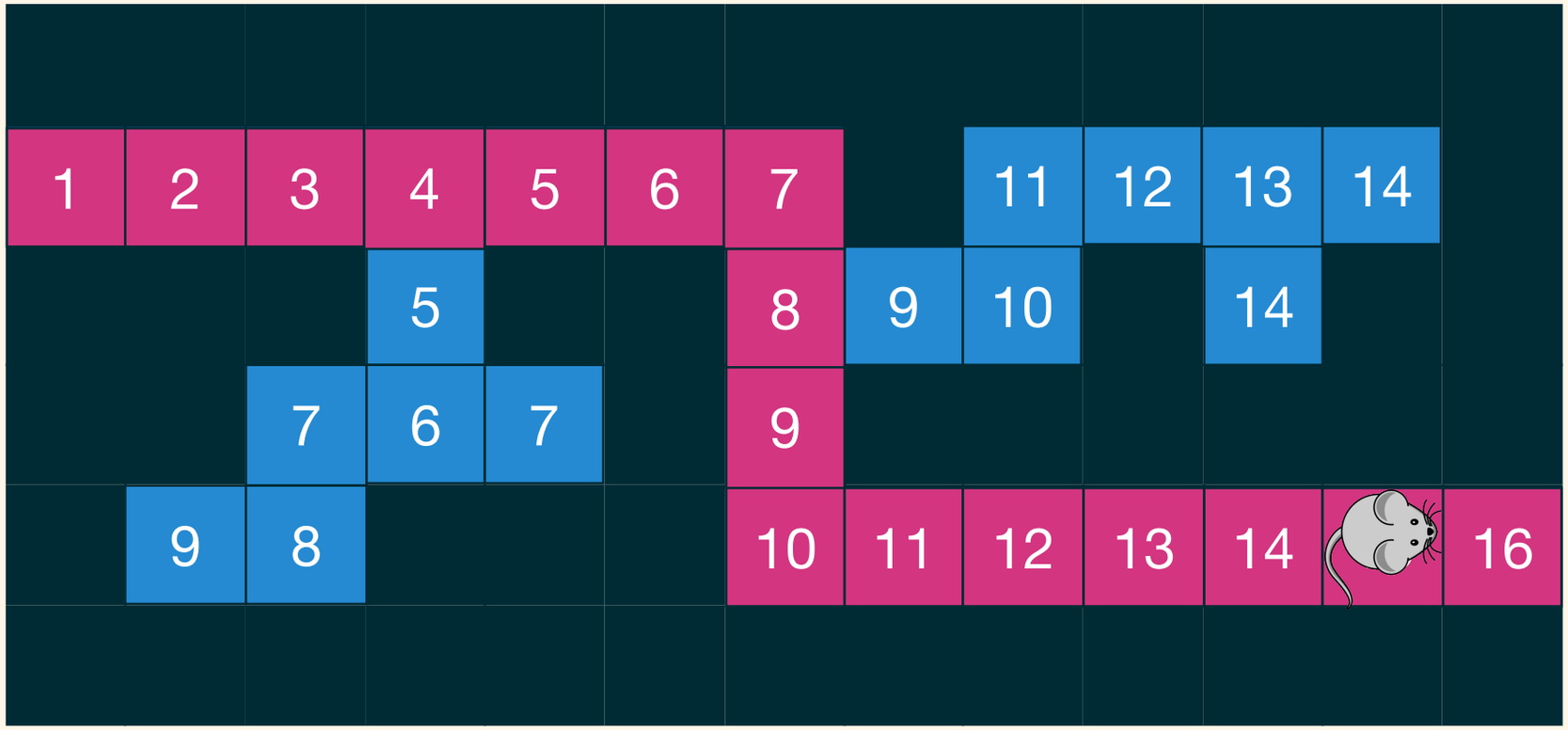
老鼠走迷宮 (BFS)



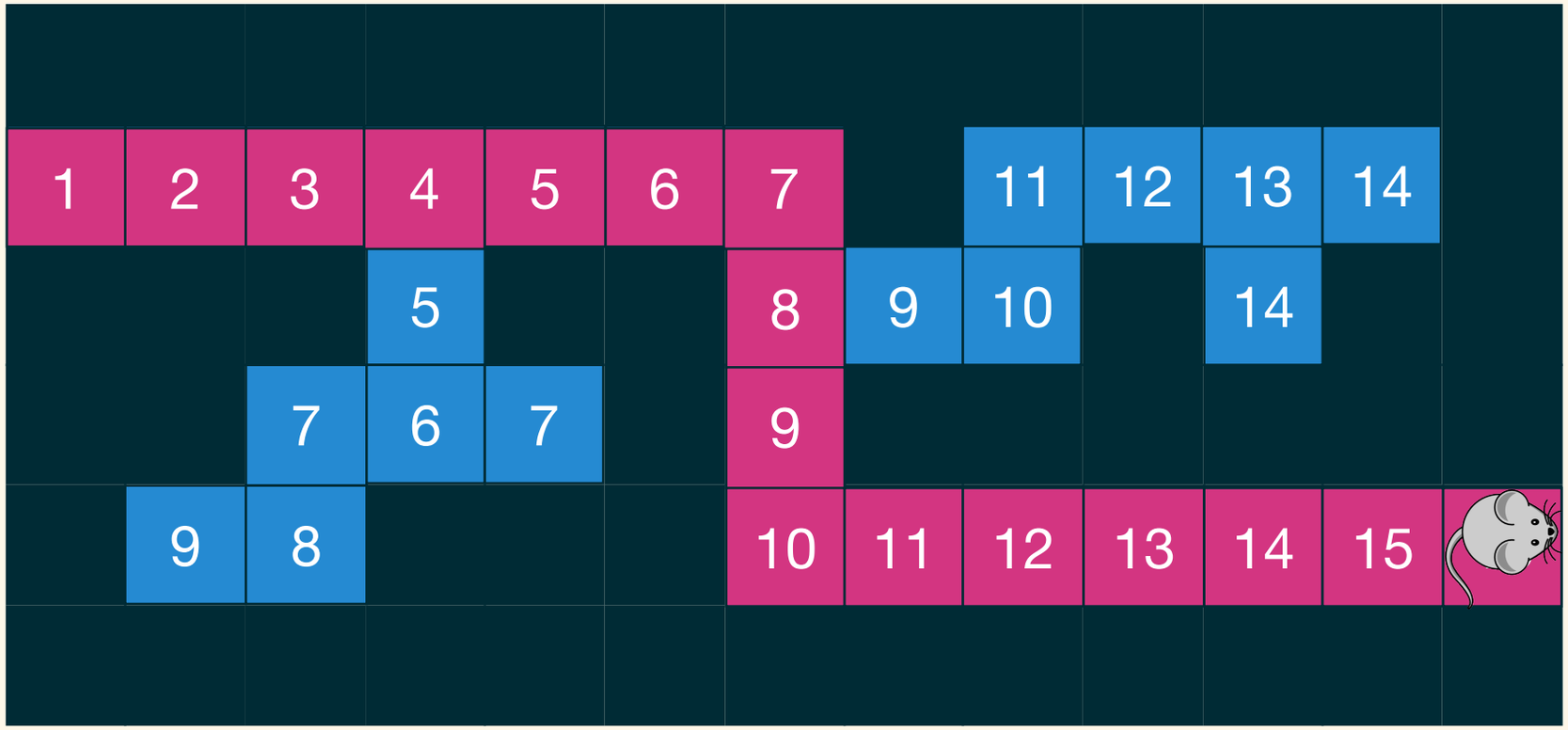
老鼠走迷宫 (BFS)



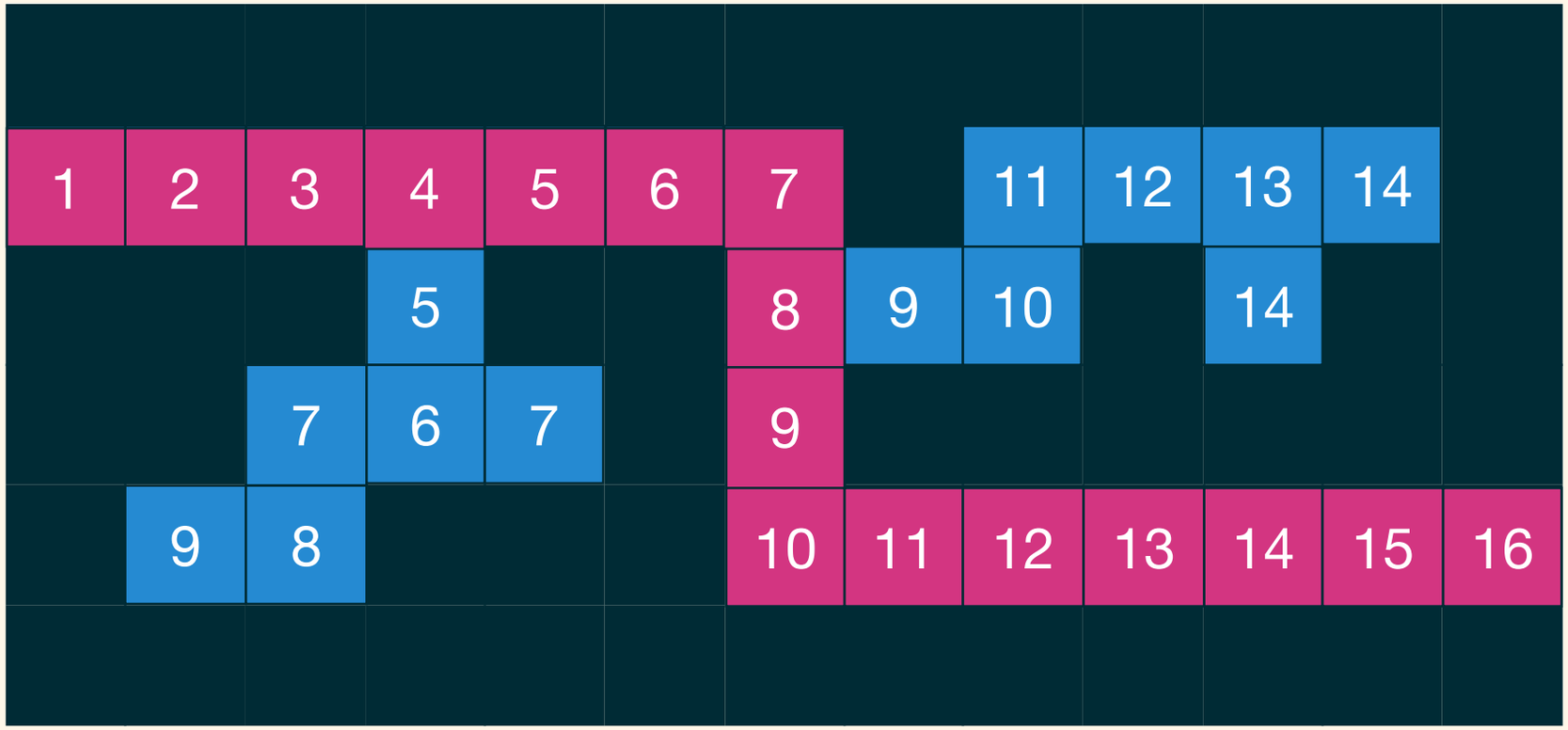
老鼠走迷宮 (BFS)



老鼠走迷宫 (BFS)



老鼠走迷宫 (BFS)



Practice Now

[UVa] 352 - The Seasonal War

352 - The Seasonal War

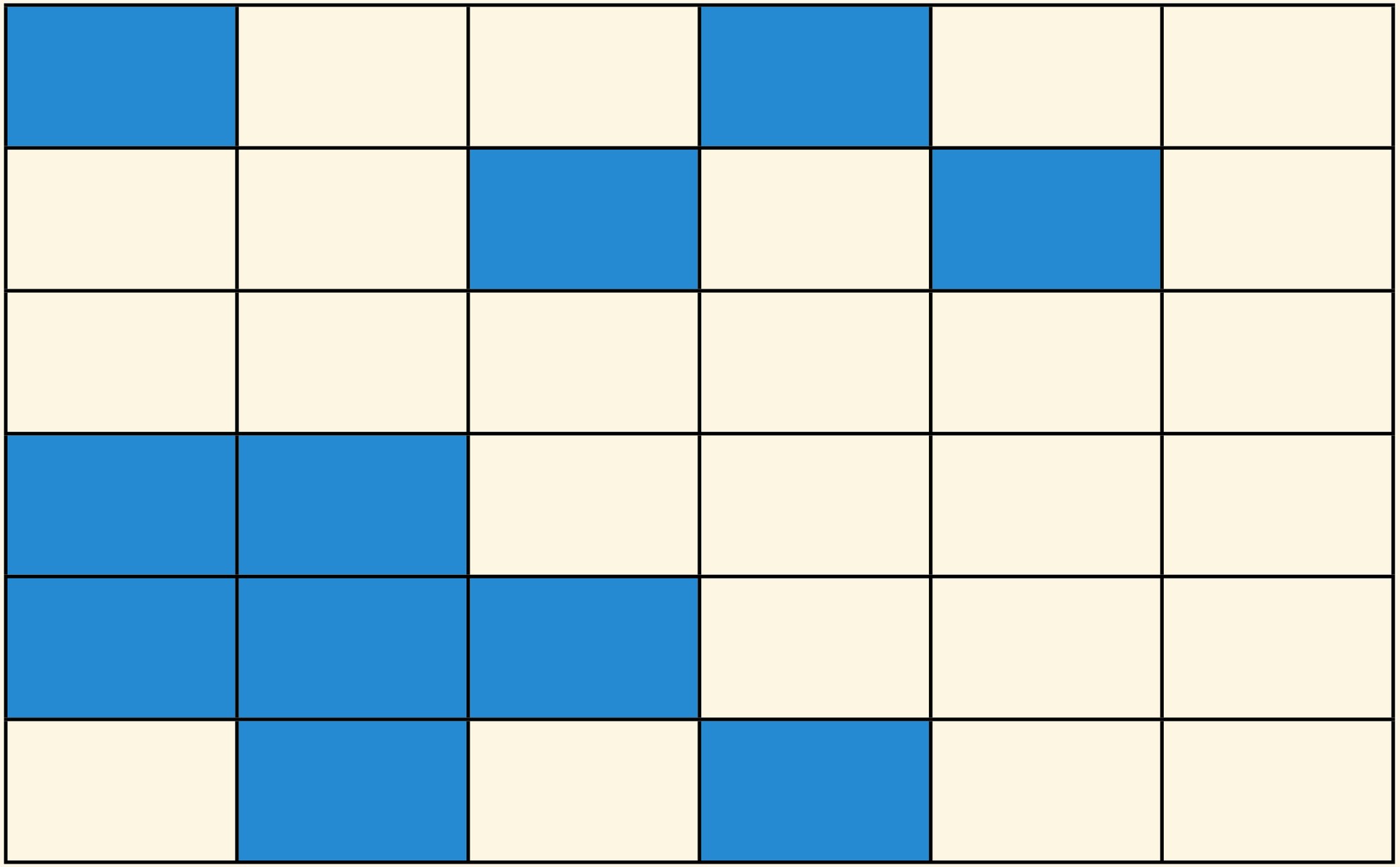
Sample Input

```
6
100100
001010
000000
110000
111000
010100
8
01100101
01000001
00011000
00000010
11000011
10100010
10000001
01100000
```

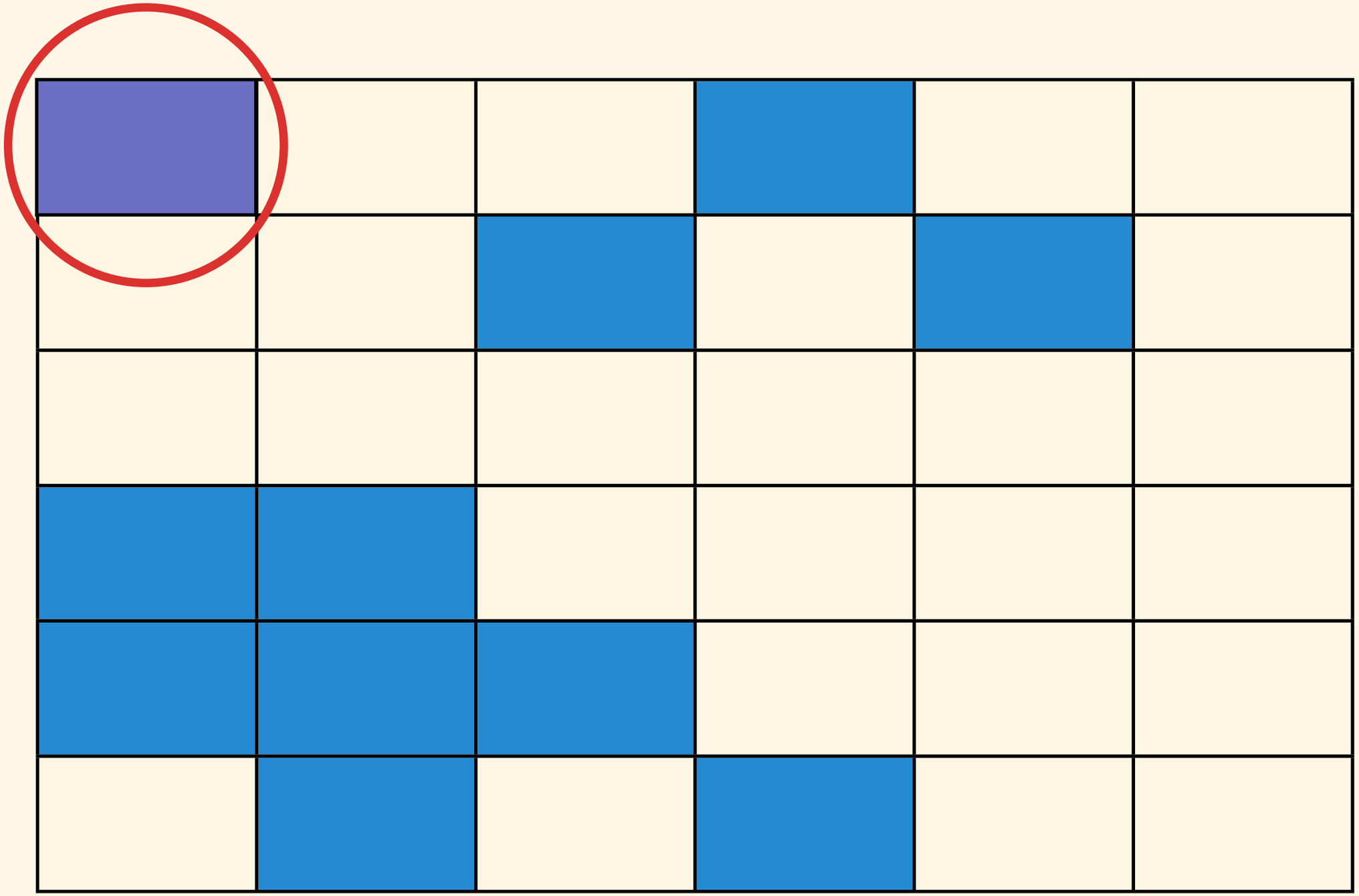
Sample Output

```
Image number 1 contains 3 war eagles.
Image number 2 contains 6 war eagles.
```

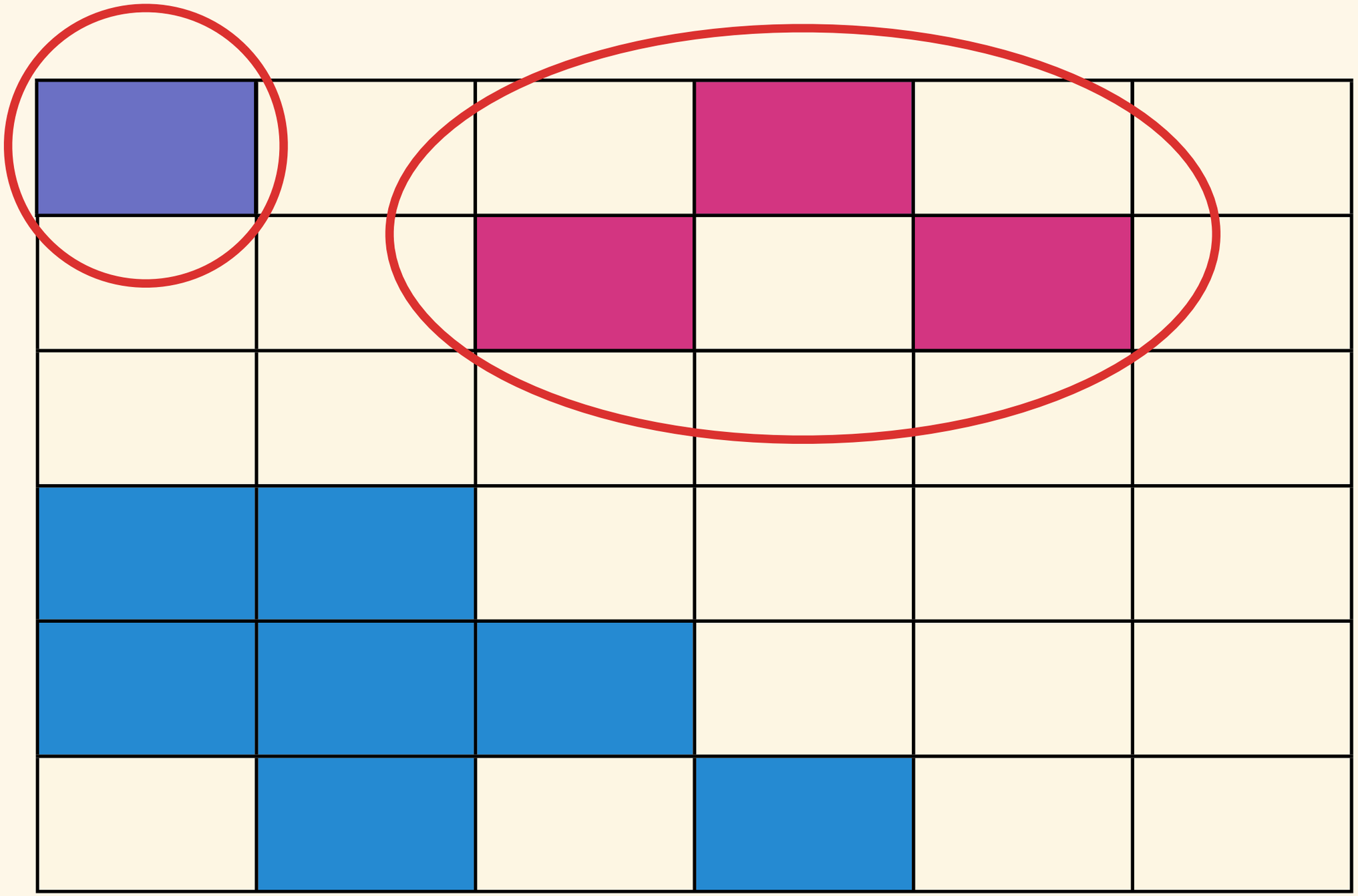
352 - The Seasonal War



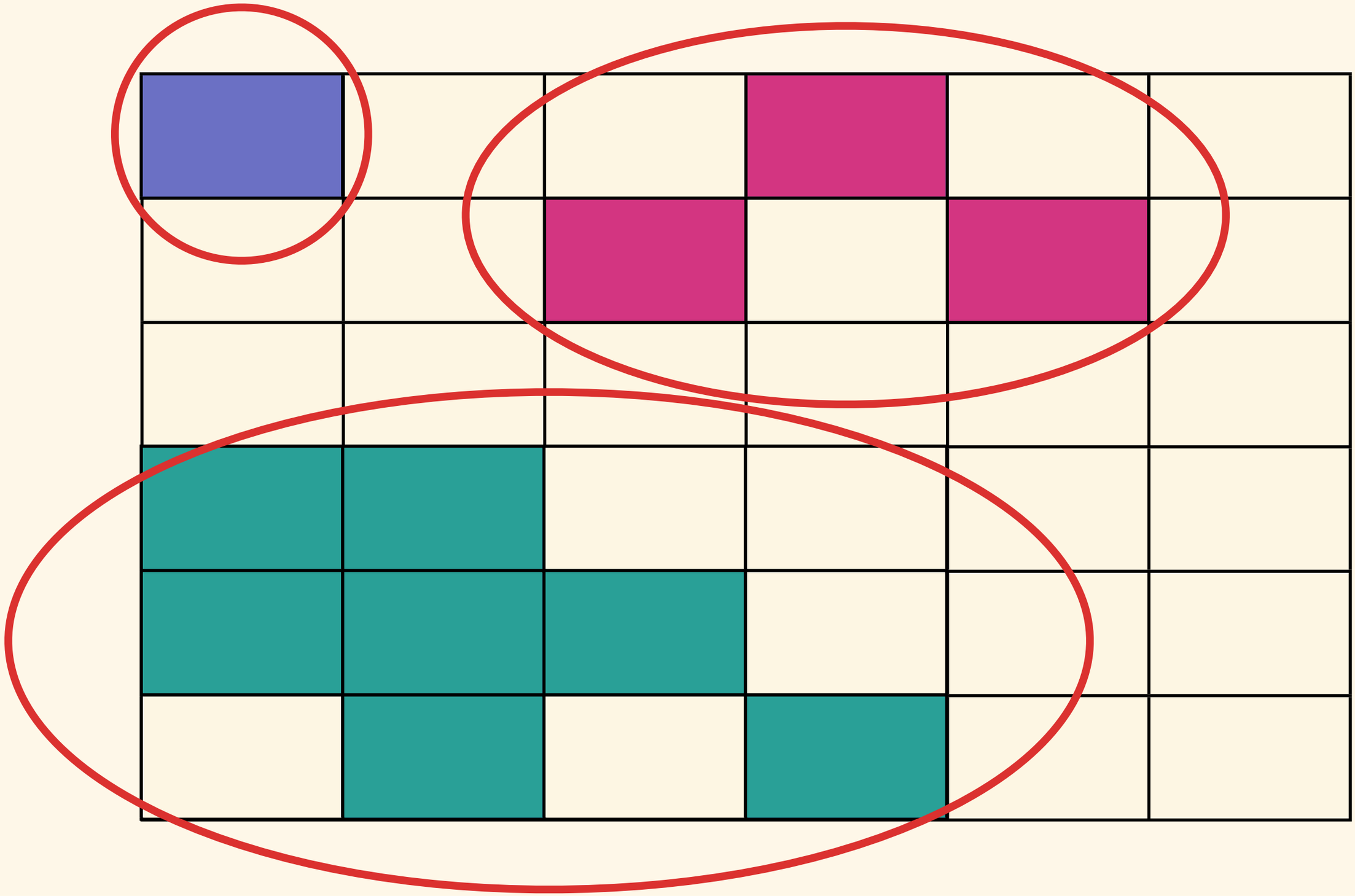
352 - The Seasonal War



352 - The Seasonal War



352 - The Seasonal War



Source Code

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

int n;
char map[ 30 ][ 30 ];
bool visited[ 30 ][ 30 ];
int step[ 8 ][ 2 ] = { { 1, 1 }, { 1, 0 }, { 1, -1 }, { 0, 1 },
{ 0, -1 }, { -1, 1 }, { -1, 0 }, { -1, -1 } };

void DFS( int r, int c );

int main() {
   int t = 0;
   while ( scanf( "%d", &n ) != EOF ) {
      int ret = 0;
      for ( int i = 0; i < n; ++i )
         scanf( "%s", map[ i ] );

      memeset( visited, 0, sizeof( visited ) );
```

Source Code

```
    for ( int i = 0; i < n; ++i )
       for ( int j = 0; j < n; ++j )
          if ( map[ i ][ j ] == '1' && visited[ i ][ j ] != true ) {
             DFS( i, j );
             ++ret;
          }
    printf( "Image number %d contains %d war eagles.\n", ++t, ret );
 }
    return 0;
 }
 void DFS( int r, int c ) {
    if ( visited[ r ][ c ] == true )
       return;
    visited[ r ][ c ] = true;
    for ( int i = 0; i < 8; ++i ) {
       int r2 = r + step[ i ][ 0 ], c2 = c + step[ i ][ 1 ];
       if ( r2 >= 0 && r2 < n && c2 >= 0 && c2 < n && map[ r2 ][ c2 ]
       == '1' )
          DFS( r2, c2 );
    }
 }
```

Practice Now

[UVa] 260 - Il Gioco dell'X

Thank You for Your Listening.