Travessia de Grafos

Breadth-First Search

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

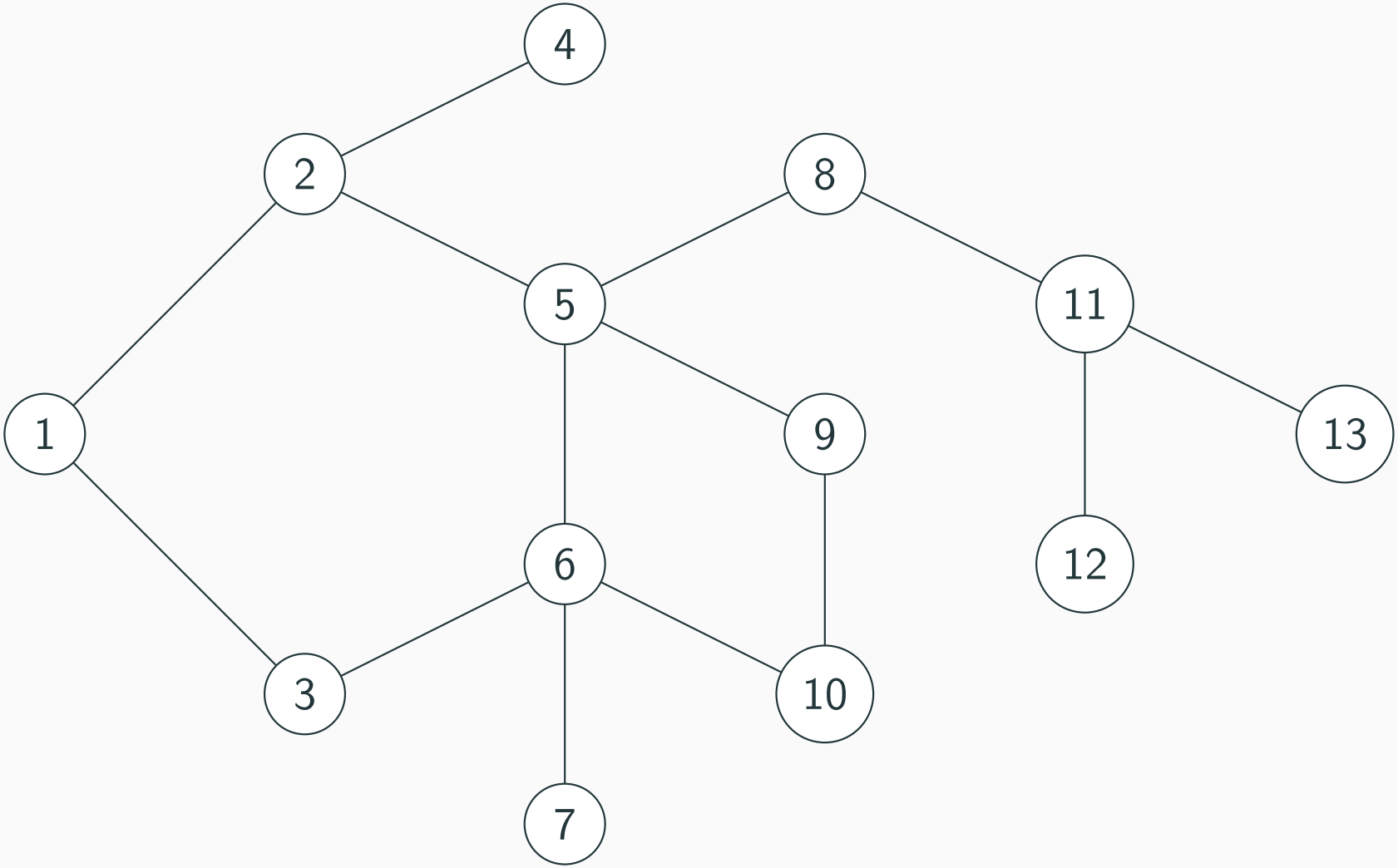
1. Definição

Definição

Breadth-First Search

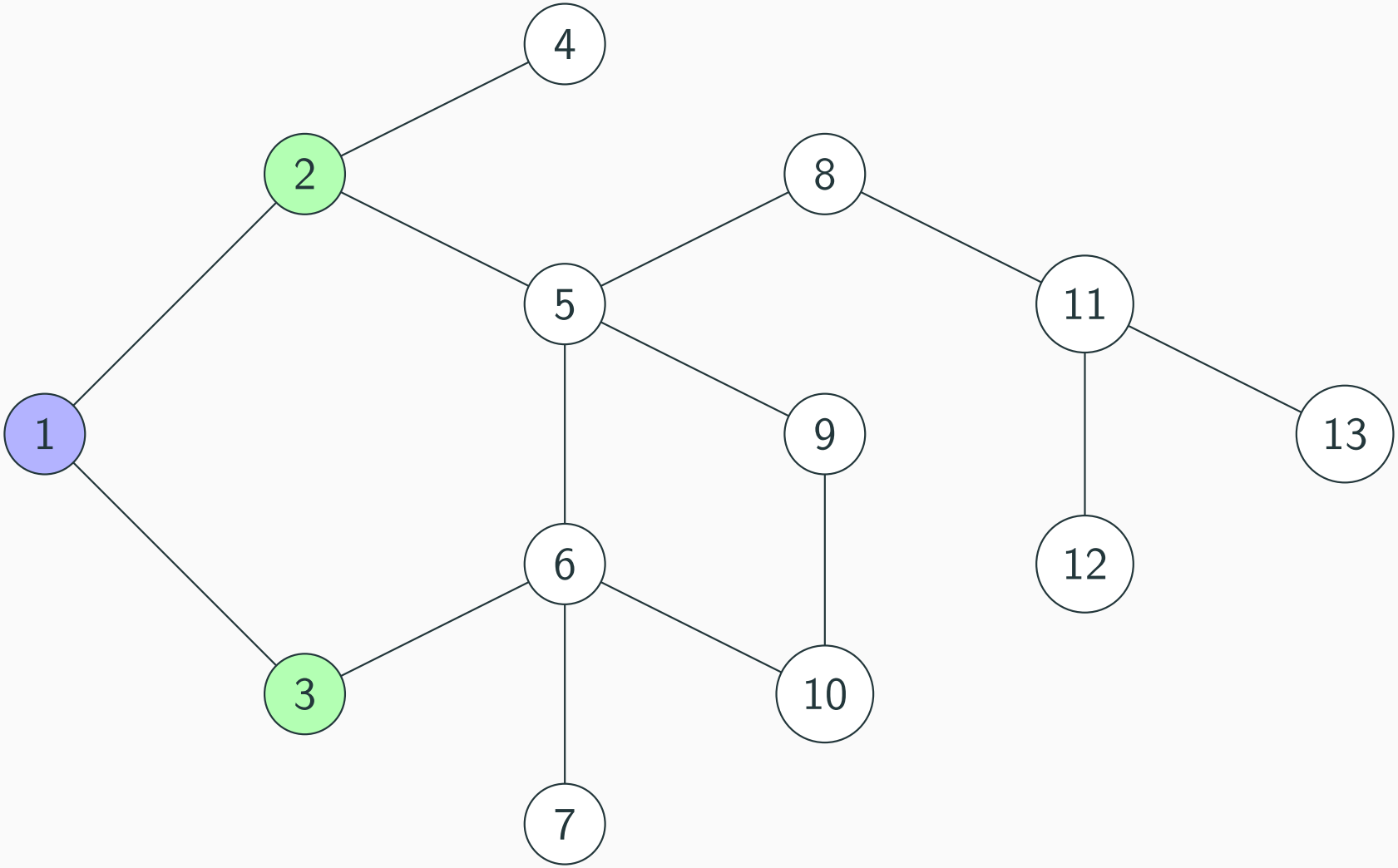
- A travessia por largura (*Breadth-First Search – BFS*) visita os nós em ordem crescente em relação à distância ao nó inicial s
- Desta forma, um subproduto da BFS é a distância de todos os nós que conectados até s
- A implementação da BFS é mais trabalhosa do que a da DFS, porque não se vale de recursão, sendo necessário o uso de uma fila explicitamente
- Ambas travessias visitam os mesmos nós, porém em ordens diferentes
- A complexidade da BFS é $O(N + M)$, onde N é o número de vértices e M o número de arestas do grafo conectado

Visualização da BFS



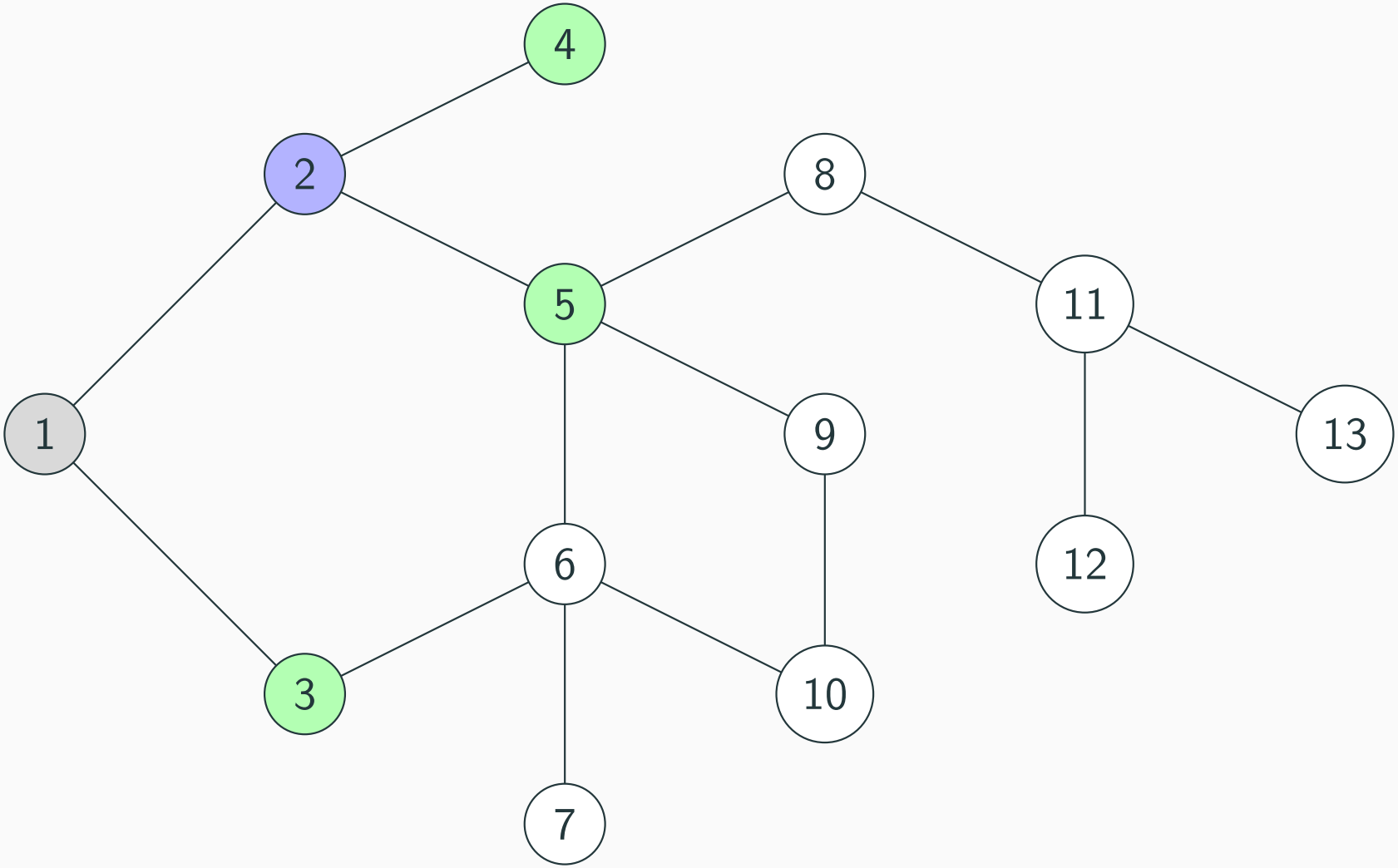
Fila: 1

Visualização da BFS



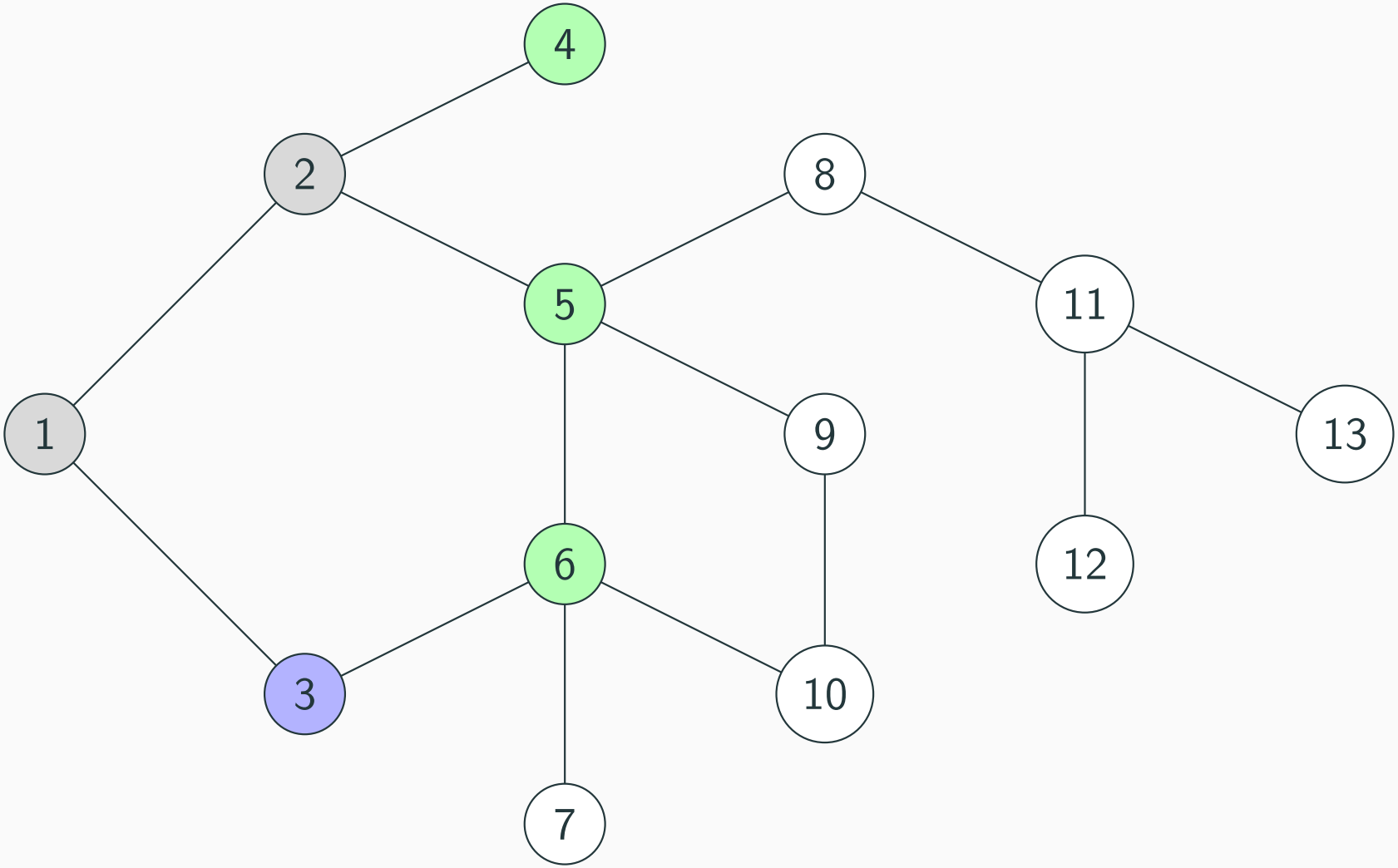
Fila: 2 3

Visualização da BFS



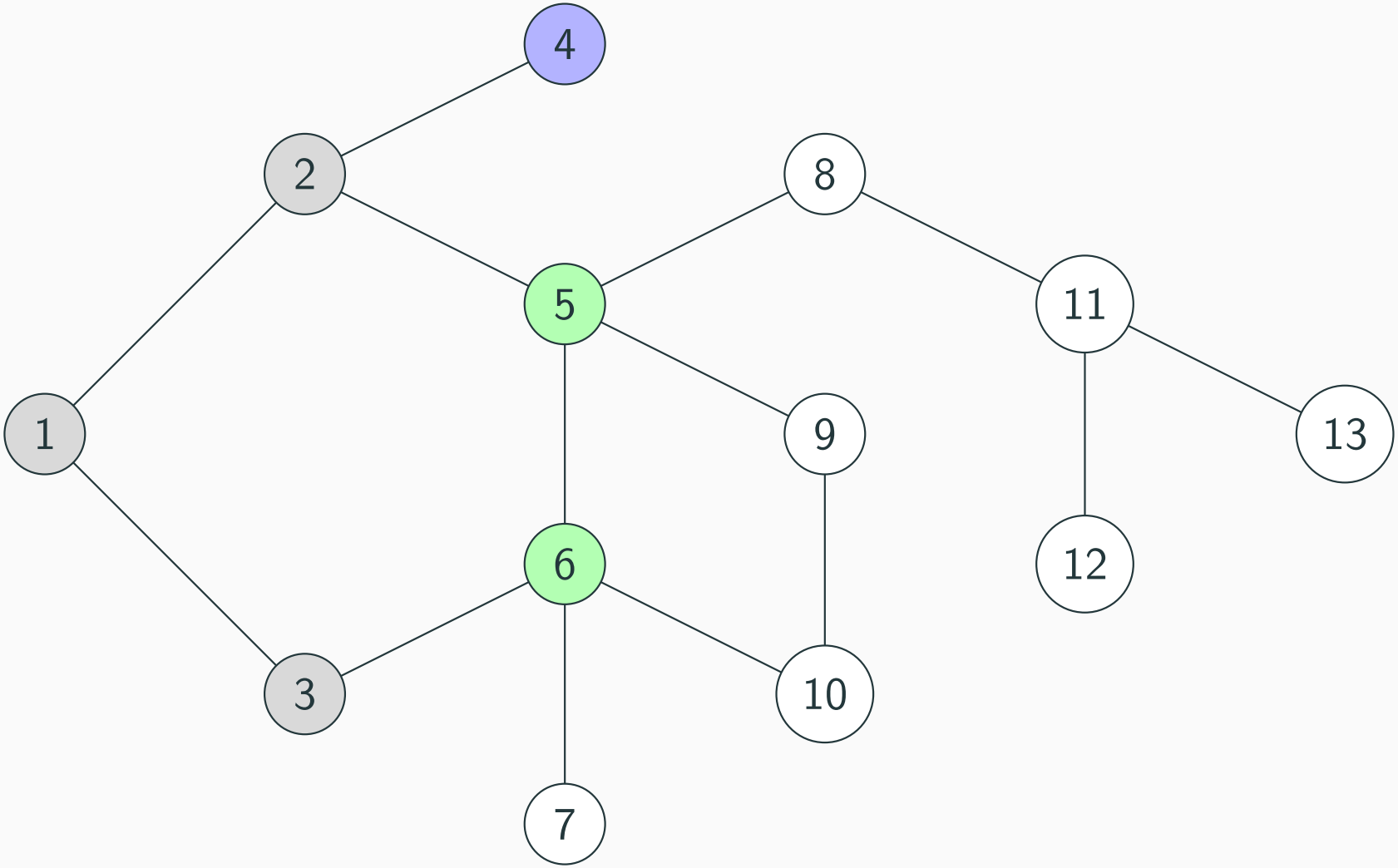
Fila: 3 4 5

Visualização da BFS



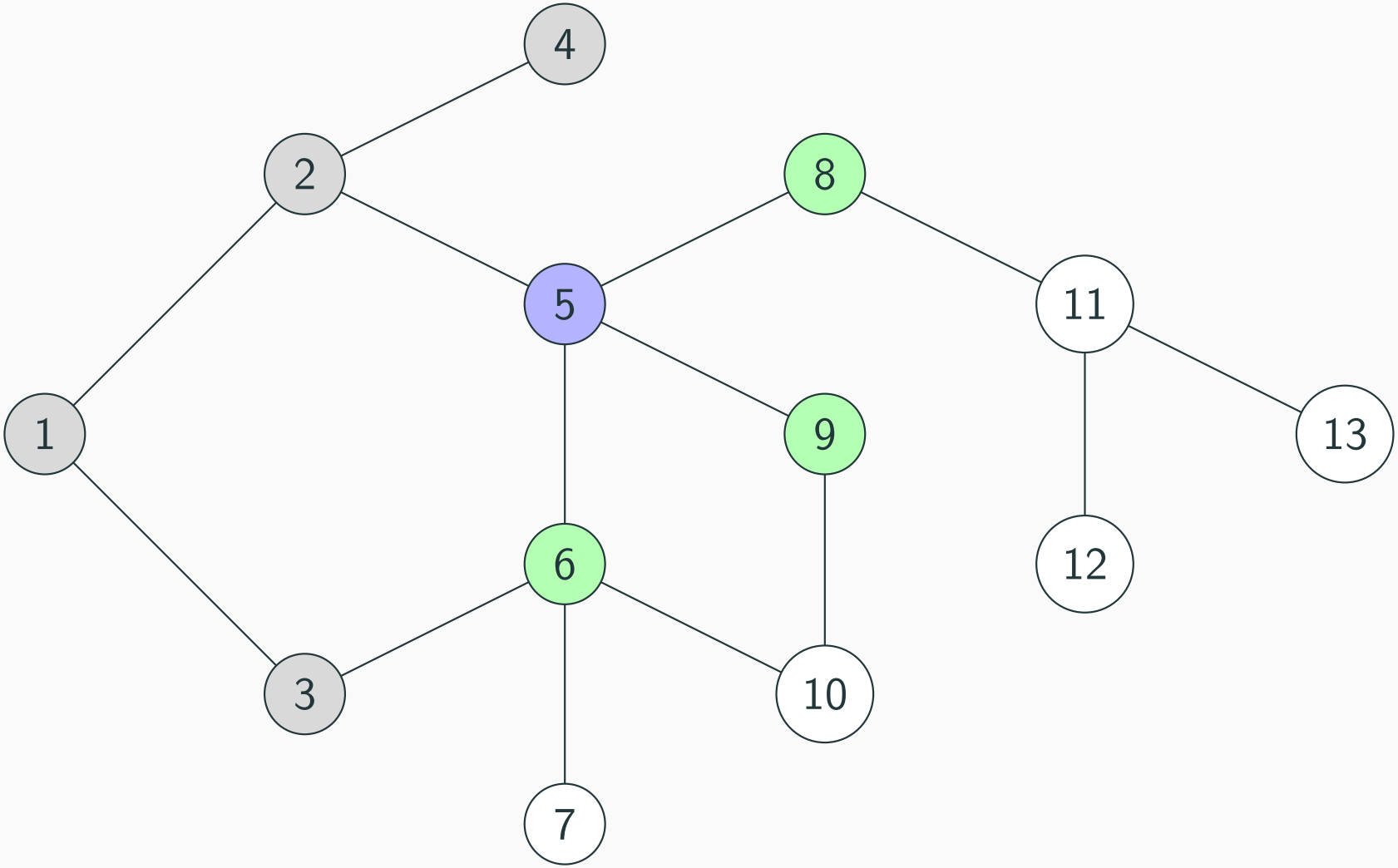
Fila: 4 5 6

Visualização da BFS



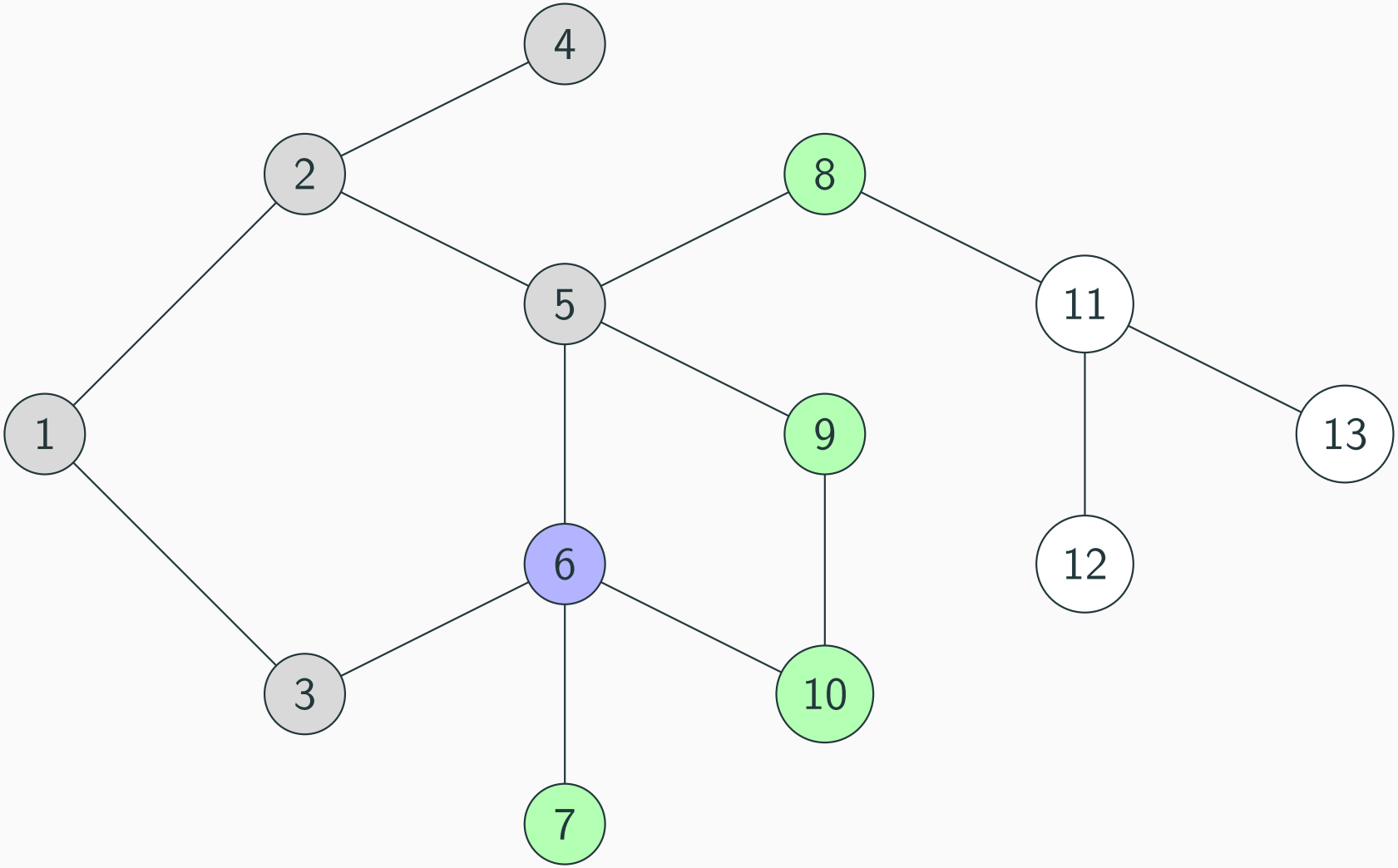
Fila: 5 6

Visualização da BFS



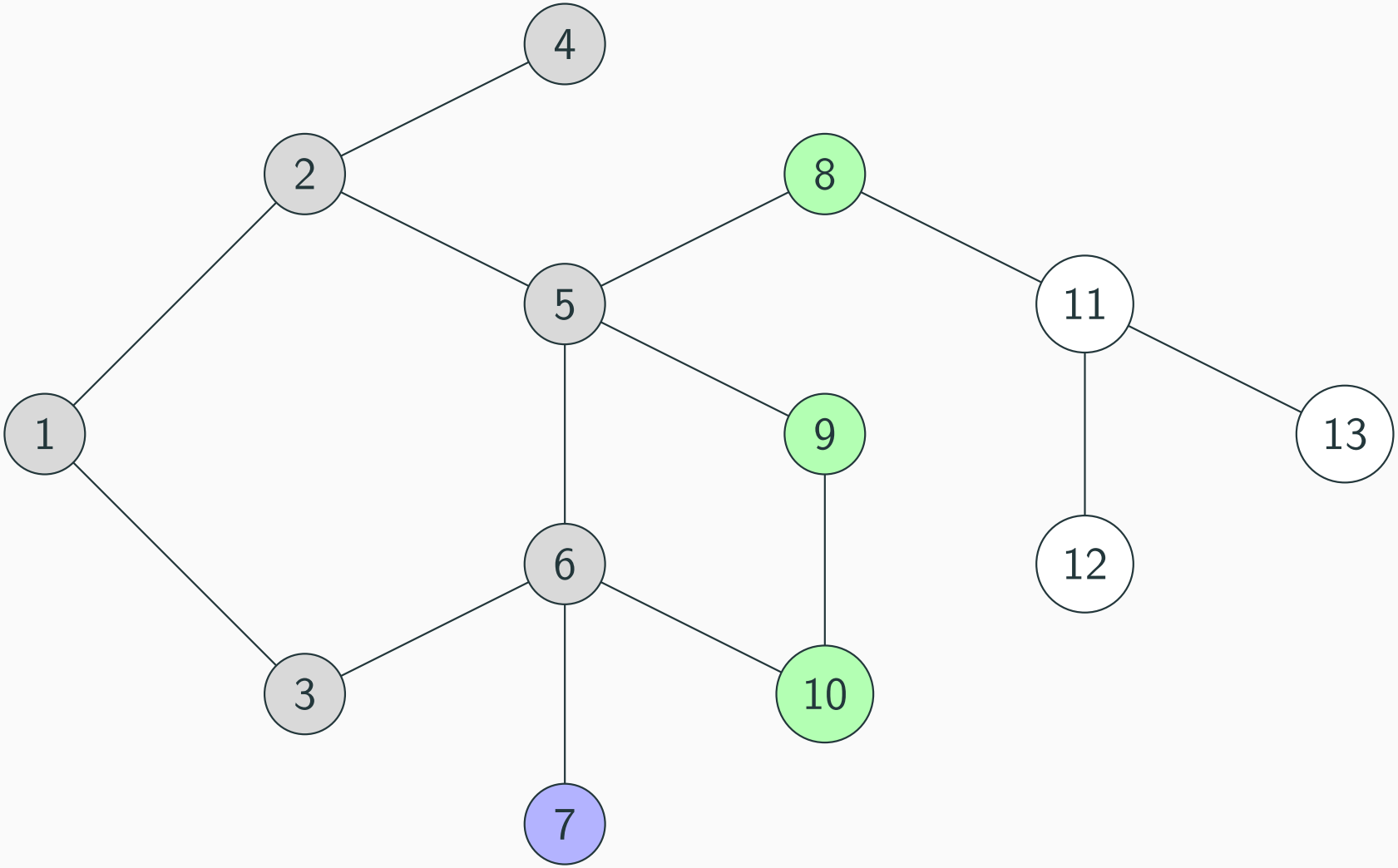
Fila: 6 8 9

Visualização da BFS



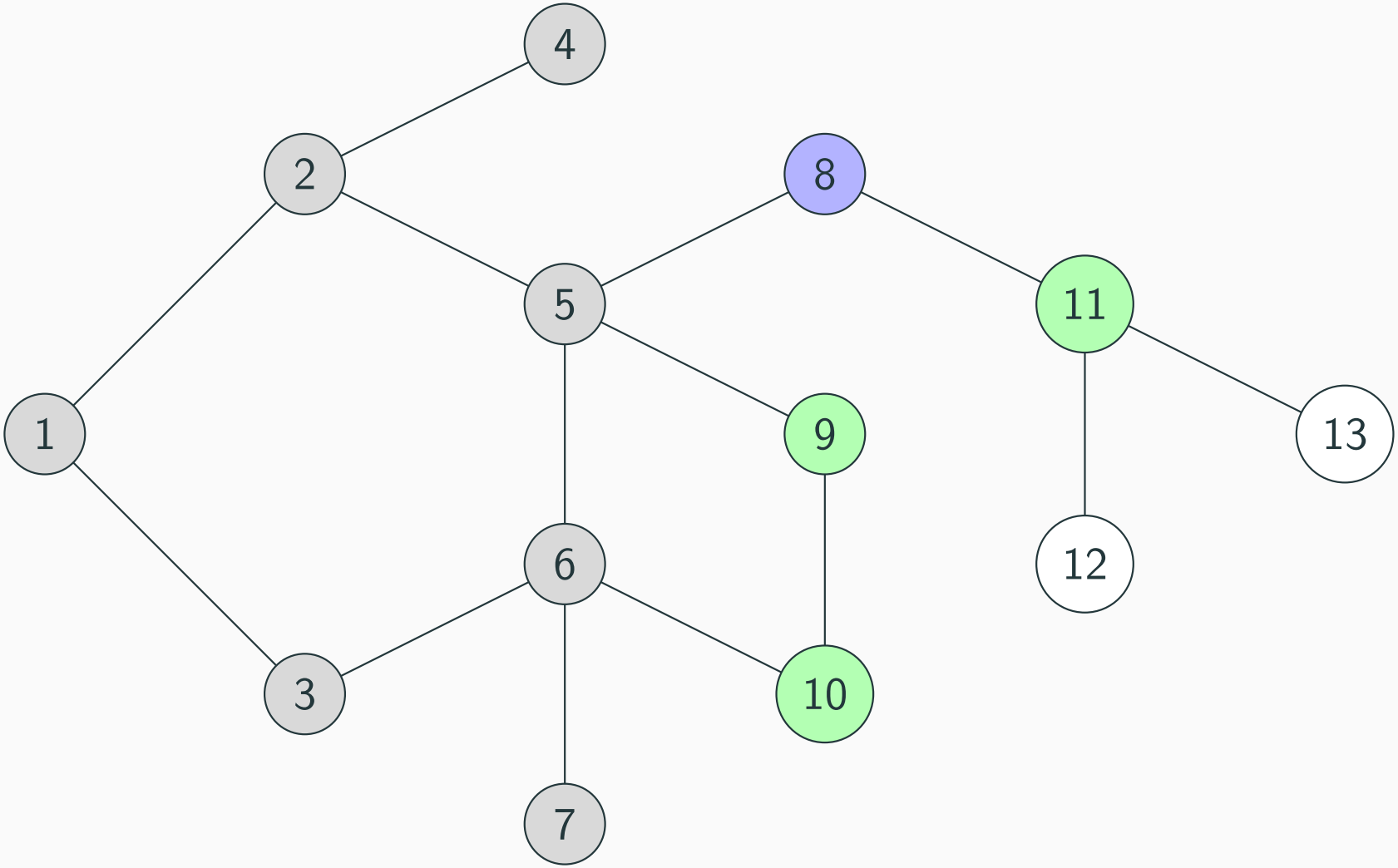
Fila: 8 9 7 10

Visualização da BFS



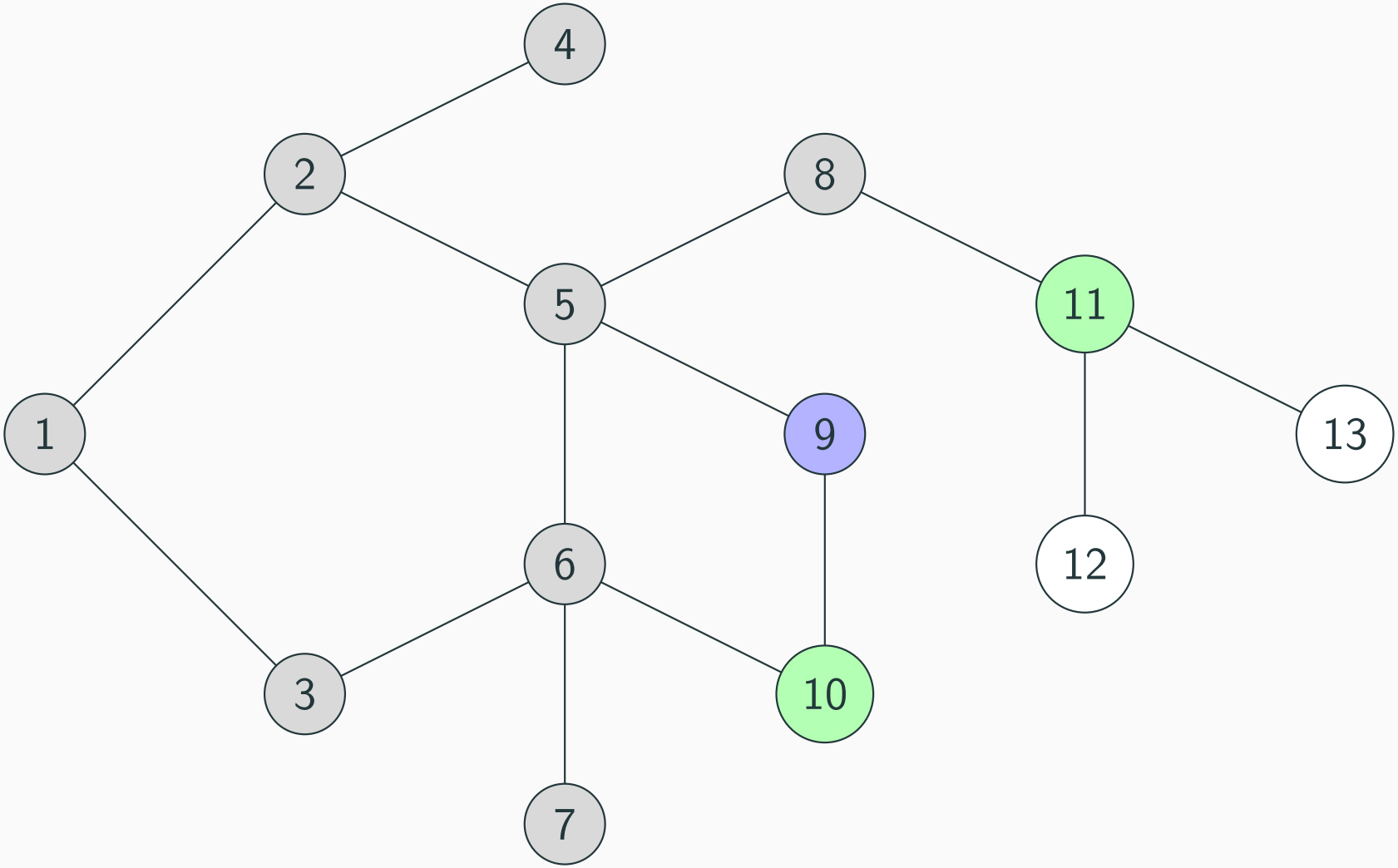
Fila: 8 9 10

Visualização da BFS



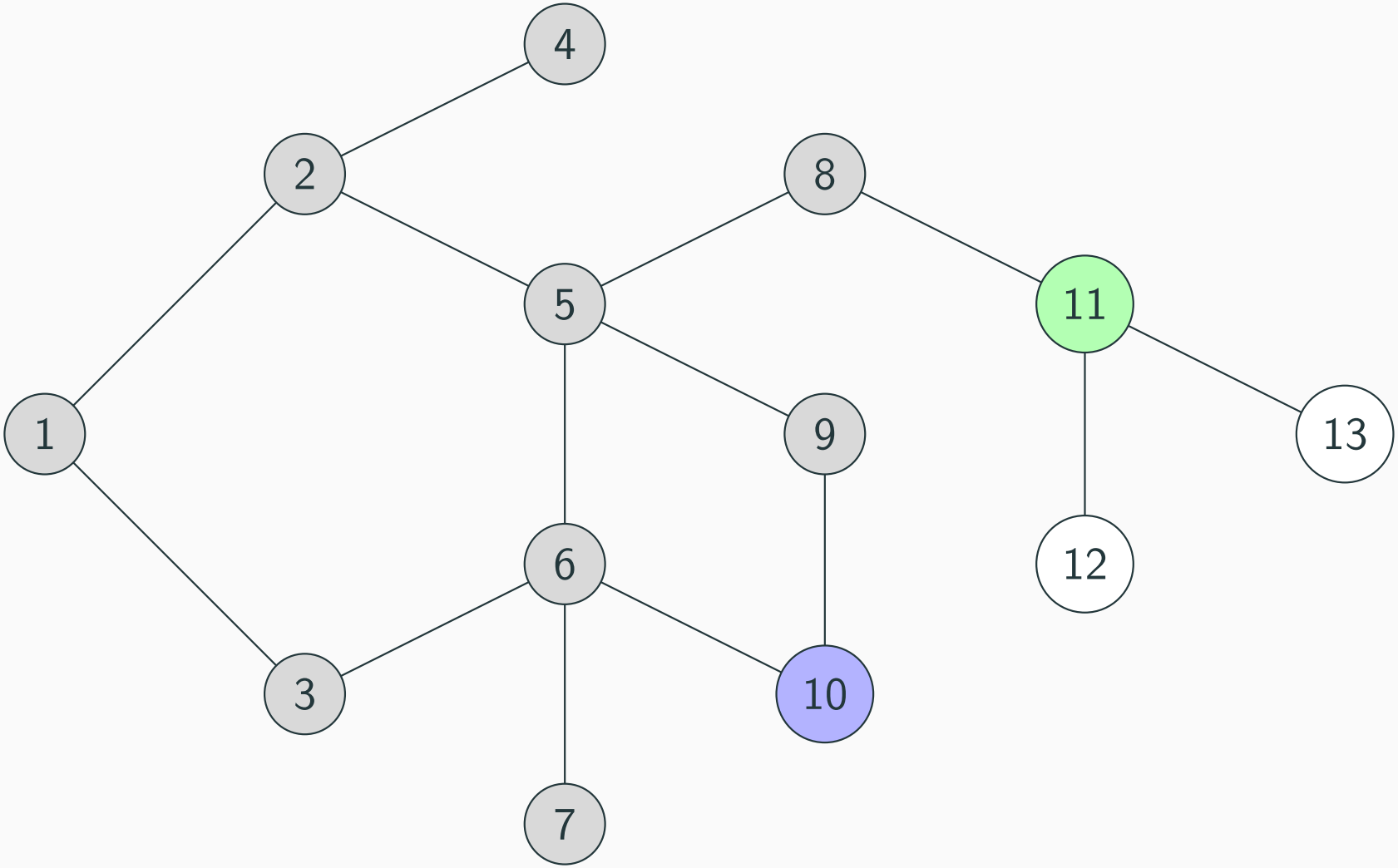
Fila: 9 10 11

Visualização da BFS



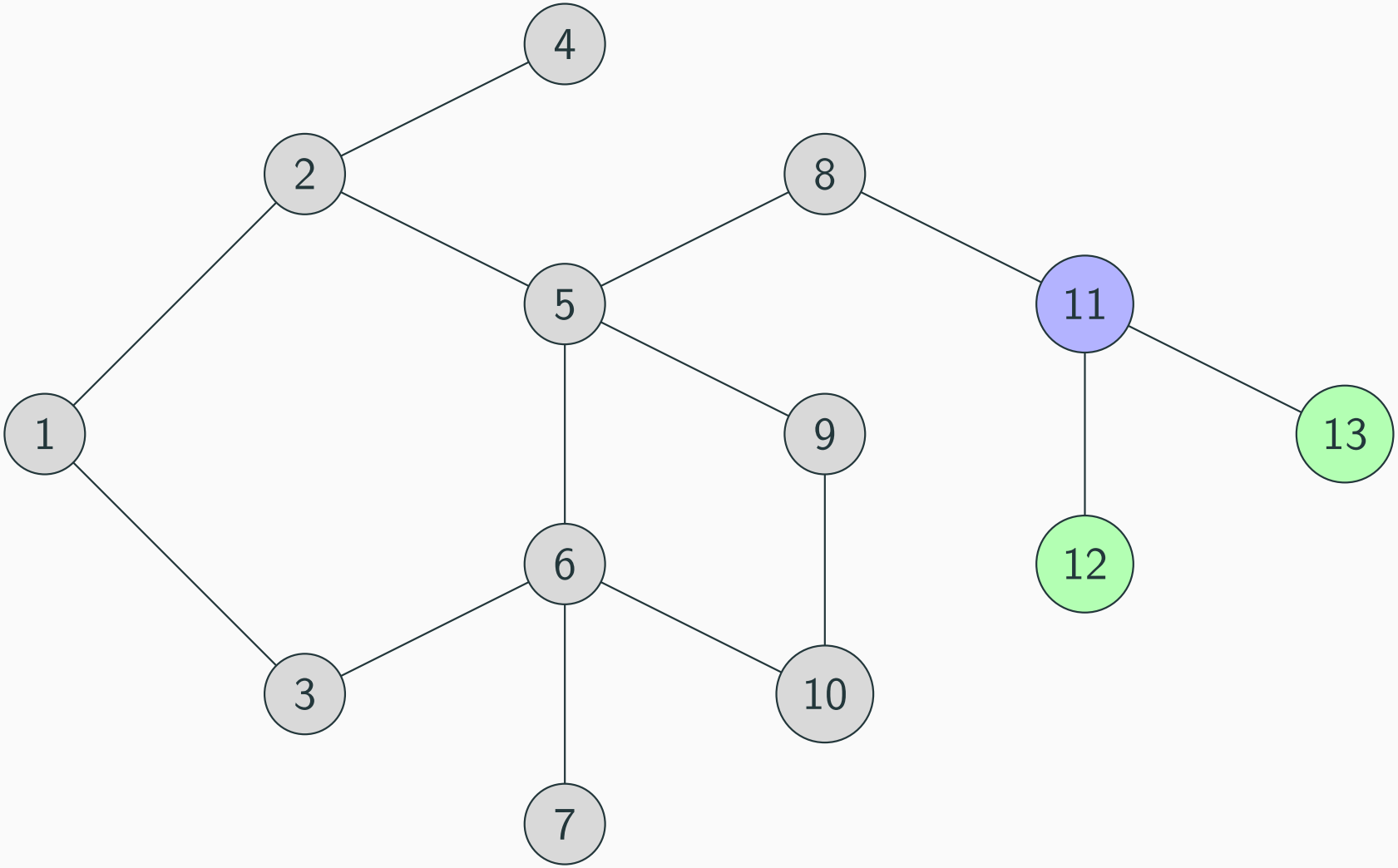
Fila: 10 11

Visualização da BFS



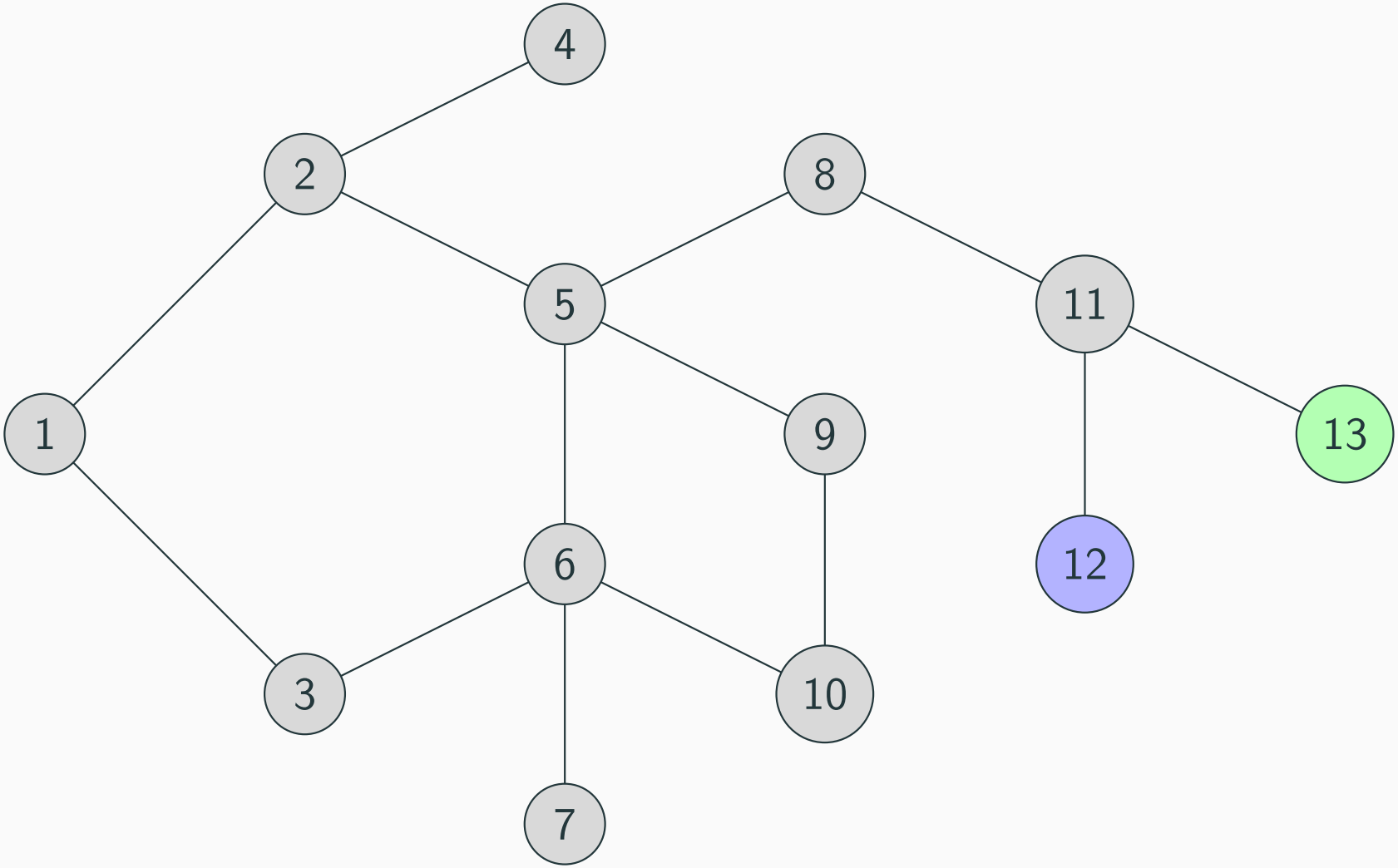
Fila: 11

Visualização da BFS



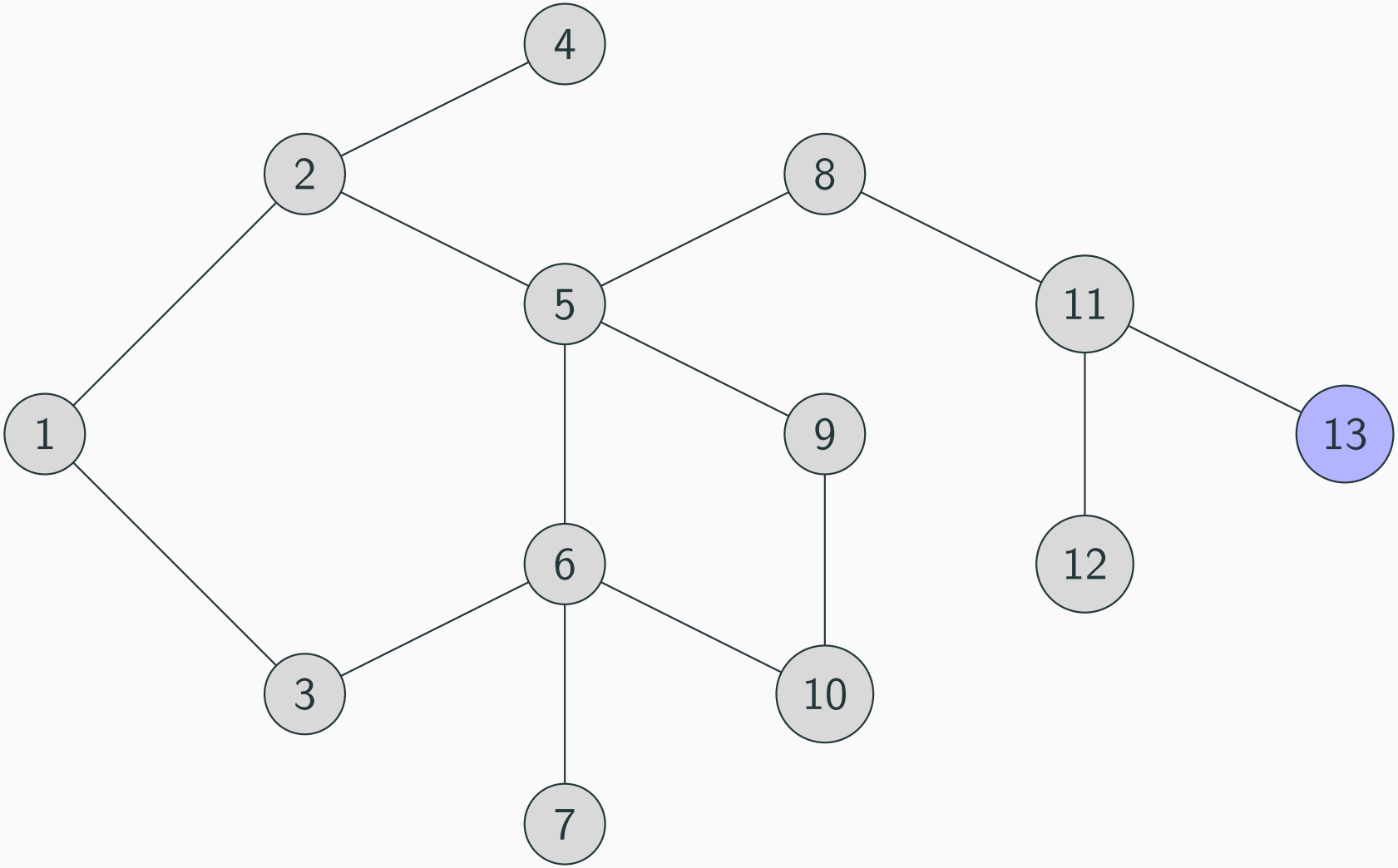
Fila: 12 13

Visualização da BFS



Fila: 13

Visualização da BFS



Fila: vazia

Implementação da BFS em C++

```cpp
#include <iostream>
#include <vector>
#include <bitset>
#include <queue>

using namespace std;
using ii = pair<int, int>;

const int MAX { 100010 };

vector<int> adj[MAX];
bitset<MAX> visited;
int dist[MAX];

void bfs(int s)
{
    queue<int> q;
    visited.reset();
    q.push(s);
    visited[s] = true;
    dist[s] = 0;
```

Implementação da BFS em C++

```
22
23      while (not q.empty())
24      {
25          auto u = q.front();
26          q.pop();
27
28          cout << u << " ";
29
30          for (const auto& v : adj[u])
31          {
32              if (visited[v])
33                  continue;
34
35              visited[v] = true;
36              dist[v] = dist[u] + 1;
37              q.push(v);
38          }
39      }
40  }
41
```

Implementação da BFS em C++

```
42 int main()
43 {
44     ii edges[] { ii(1, 2), ii(1, 3), ii(2, 4), ii(2, 5), ii(3, 6),
45         ii(5, 6), ii(5, 8), ii(5, 9), ii(6, 7), ii(6, 10), ii(8, 11),
46         ii(9, 10), ii(11, 12), ii(11, 13) };
47
48     for (const auto& [u, v] : edges)
49     {
50         adj[u].push_back(v);
51         adj[v].push_back(u);
52     }
53
54     bfs(1);
55     cout << endl;
56
57     return 0;
58 }
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **SKIENA**, Steven S.; **REVILLA**, Miguel A. *Programming Challenges*, 2003.
4. **FILIPEK**, Bartlomiej. *C++17 in Detail*, 2018[1].

[1] https://leanpub.com/cpp17indetail