Busca em Largura

Letícia Rodrigues Bueno

UFABC

Número de Erdõs - Equivalente Nerd do Número de Bacon :)

- Paul Erdõs: famoso matemático hungáro;
- Trabalhou com centenas de colaboradores;
- Publicou mais de 1.400 artigos;
- Número de Erdõs é um tributo divertido criado pelos amigos;
- Paul Erdõs tem número de Erdõs 0;
- Os colaboradores diretos tem número 1;
- Os colaboradores dos colaboradores tem número 2 e assim por diante;

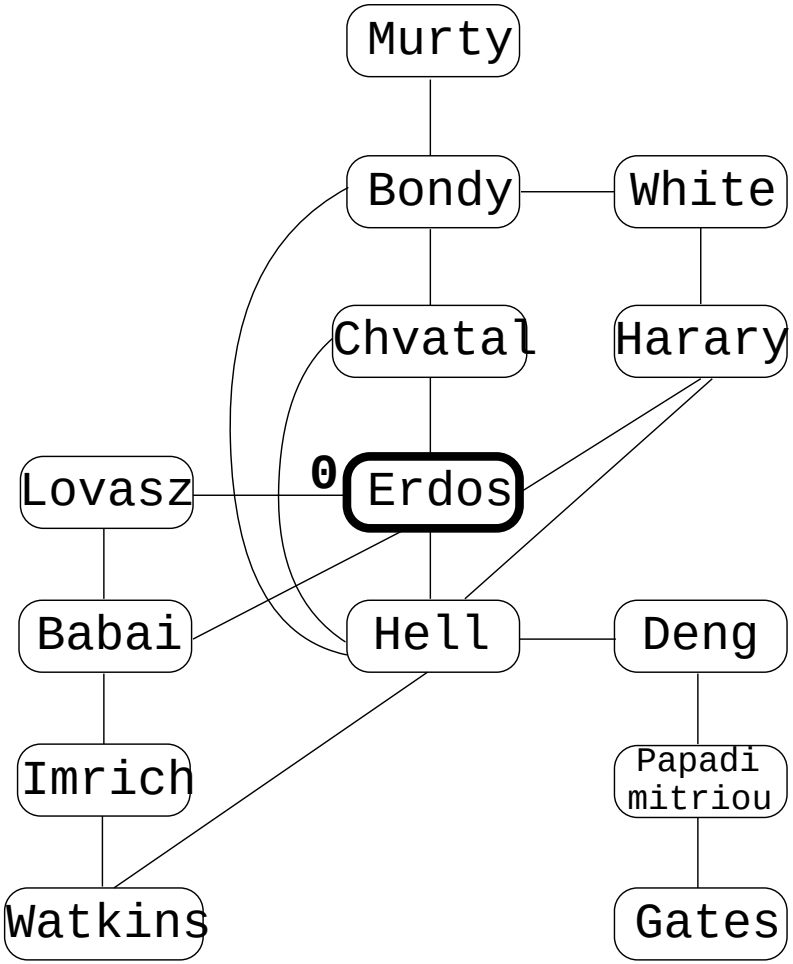
Definição

- Dada uma lista de pessoas e as relações de colaboração entre elas, qual é o número de Erdõs de cada pessoa?

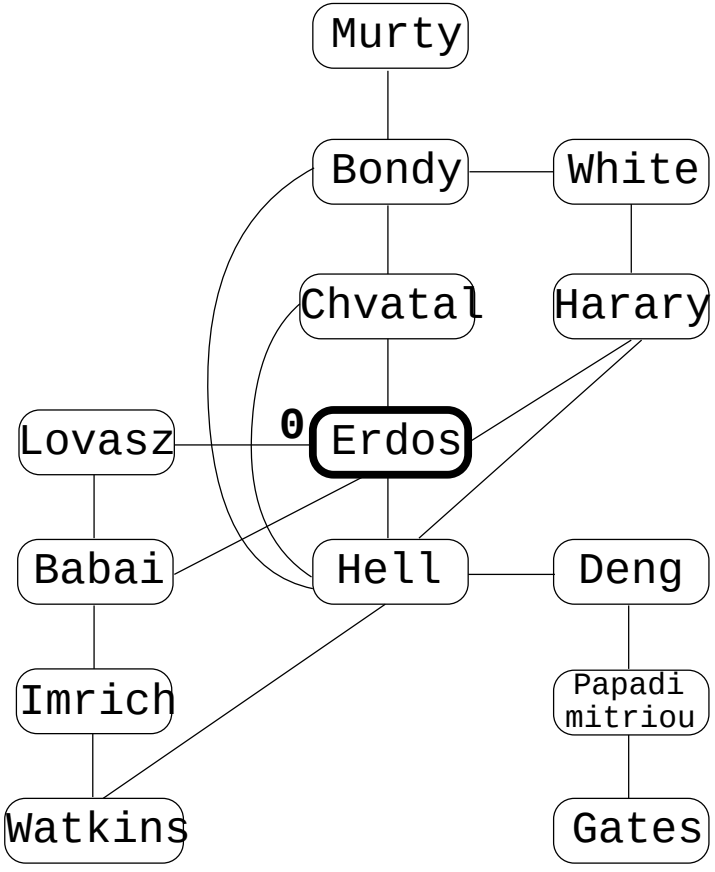
Definição

- Dada uma lista de pessoas e as relações de colaboração entre elas, qual é o número de Erdõs de cada pessoa?
- Este problema pode ser modelado através de um grafo:
  - As pessoas são os vértices;
  - As relações de colaboração são as arestas.

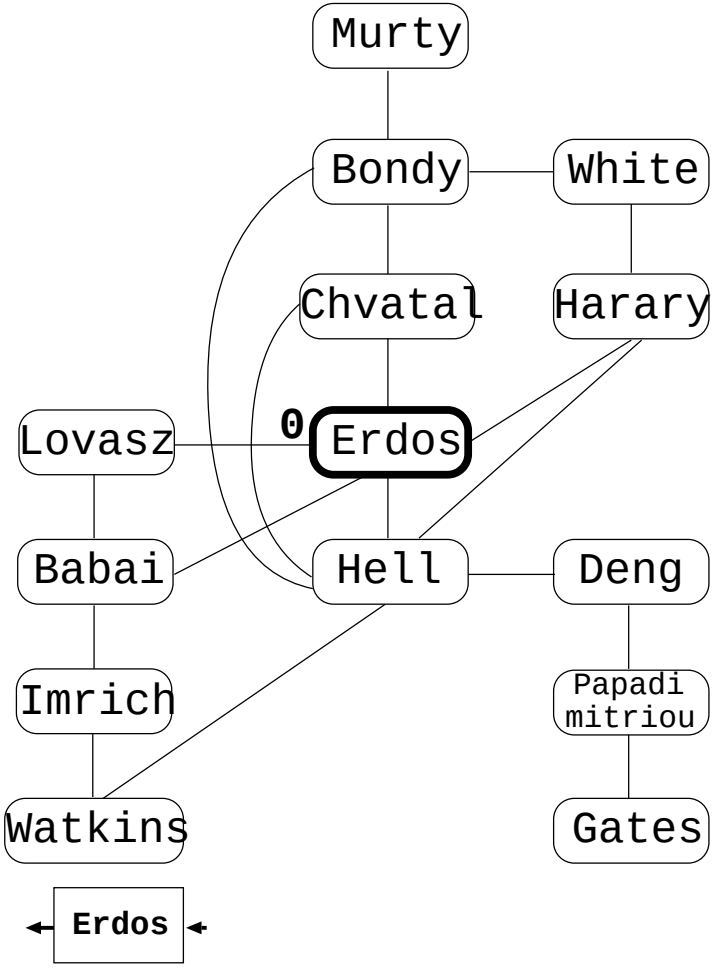
Exemplo


Murty
Bondy
White
Chvatal
Harary
0
Erdos
Lovasz
Babai
Hell
Deng
Imrich
Papadi
mitriou
Watkins
Gates

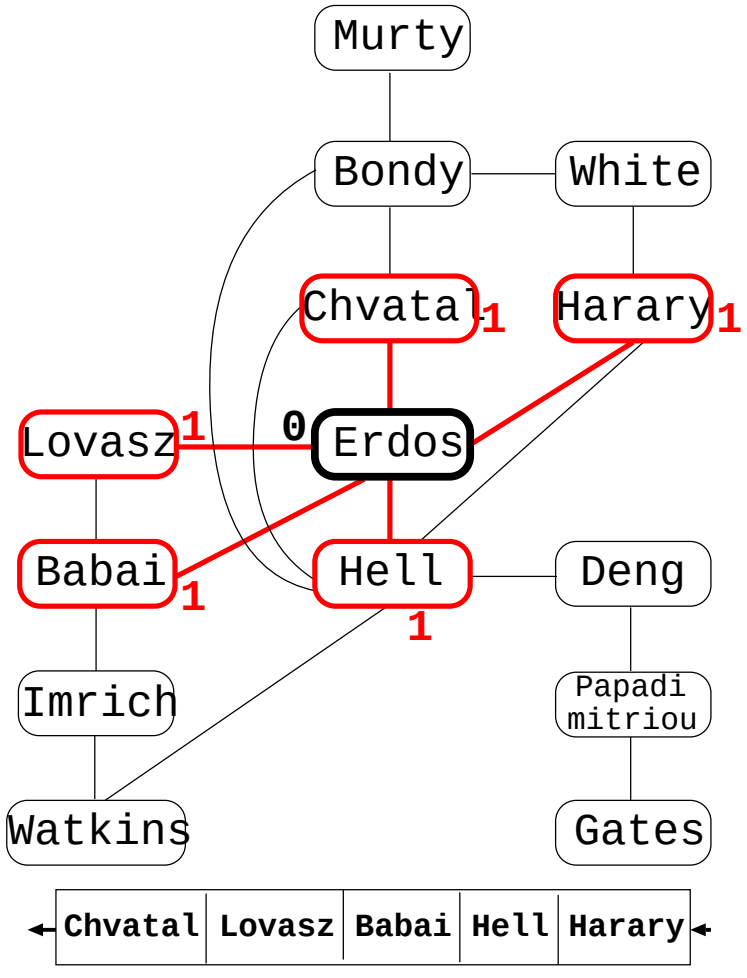
Busca em Largura (BFS - *Breadth-First Search*)



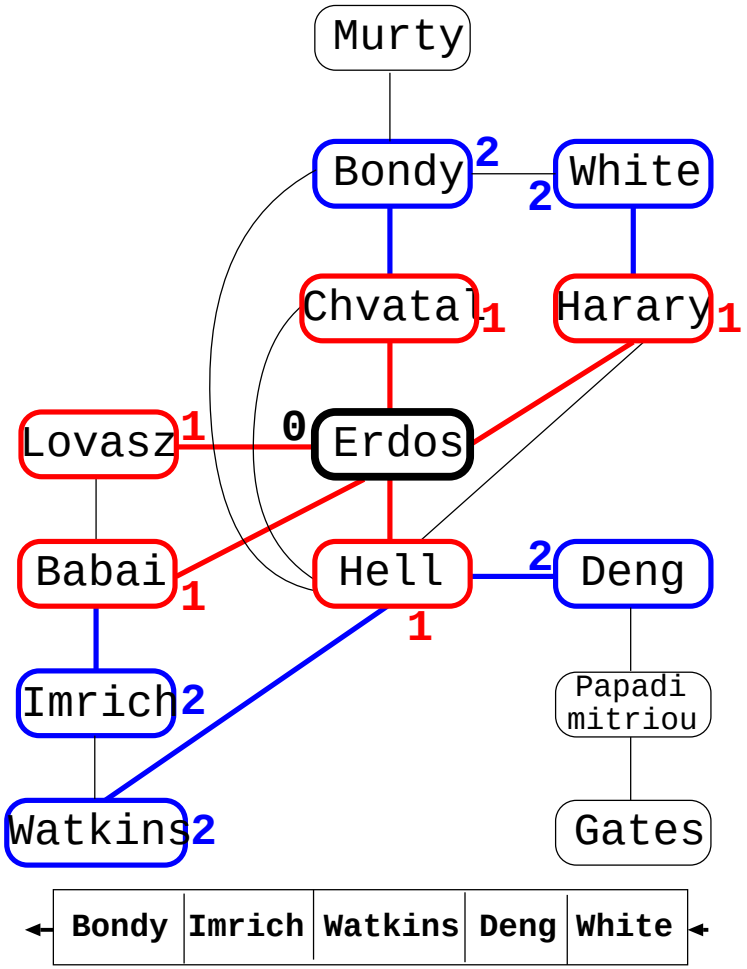
Busca em Largura (BFS - *Breadth-First Search*)



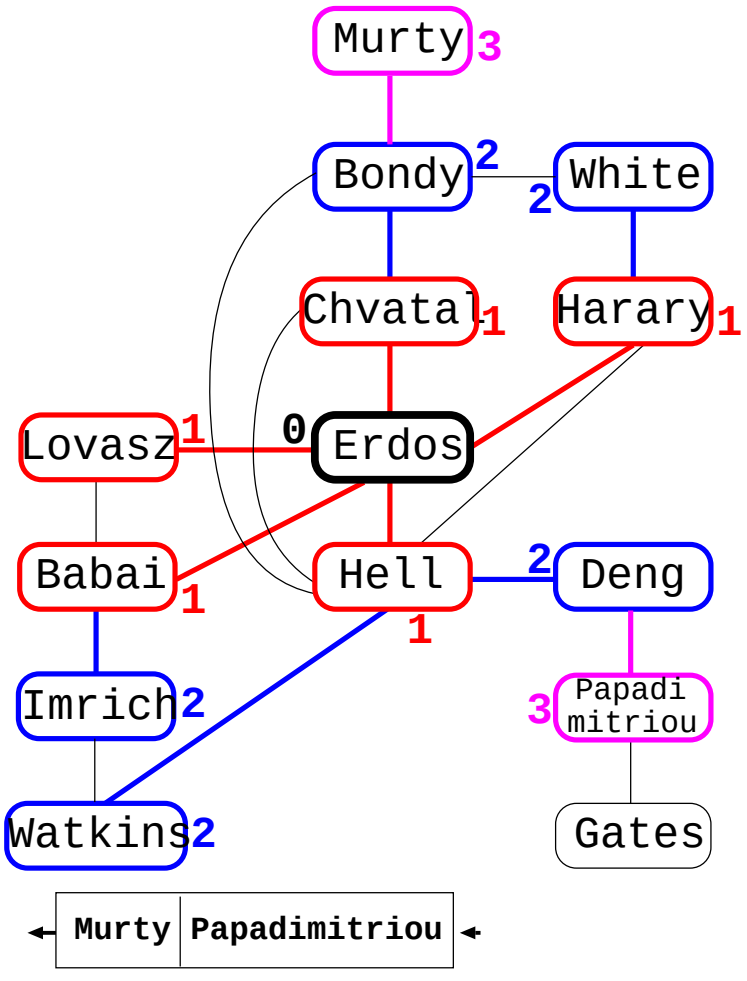
Busca em Largura (BFS - *Breadth-First Search*)



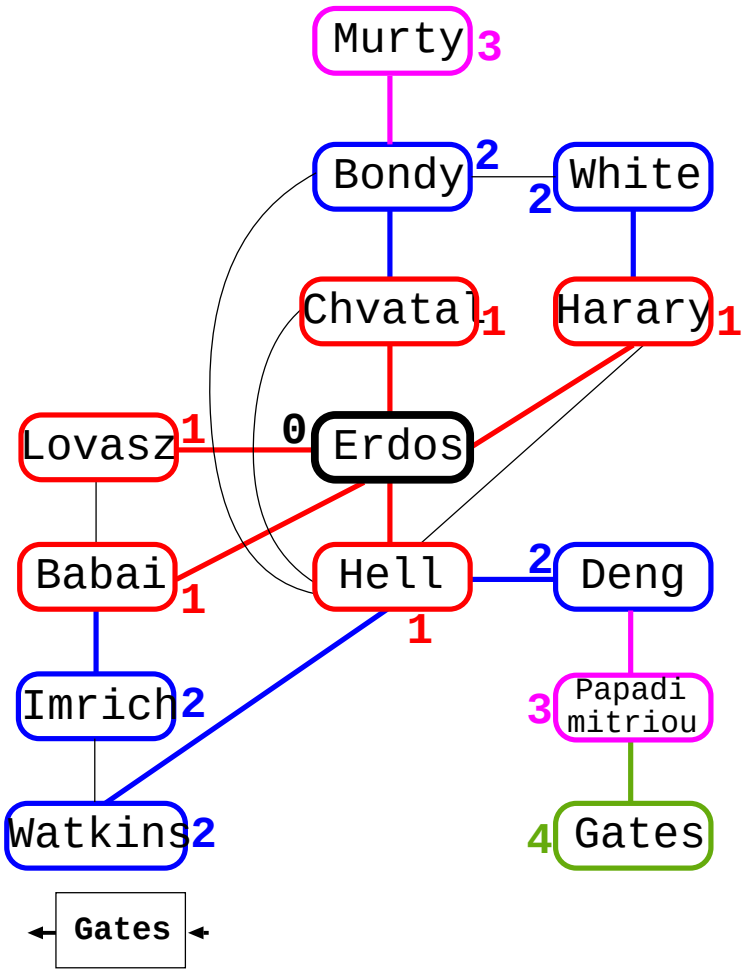
Busca em Largura (BFS - *Breadth-First Search*)



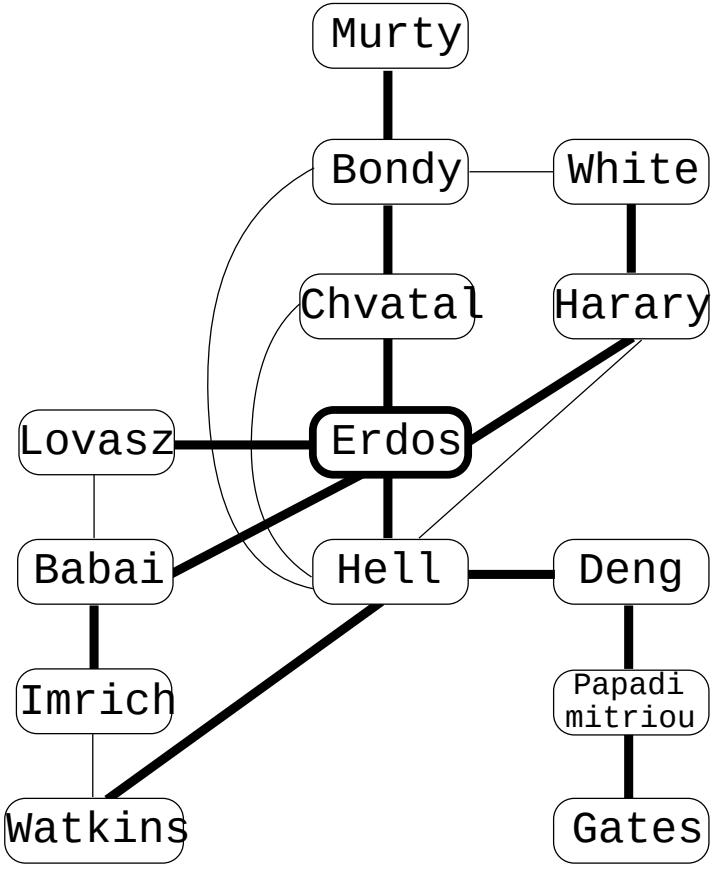
Busca em Largura (BFS - *Breadth-First Search*)



Busca em Largura (BFS - *Breadth-First Search*)



Busca em Largura (BFS - *Breadth-First Search*)



Algoritmo BFS

```
 1  bfs(G, s):
 2      para u em V(G) faça
 3          u.visitado = False
 4          u.d = ∞
 5          u.p = None
 6      s.visitado = True
 7      s.d = 0
 8      Q = Fila()
 9      insere(Q, s)
10      enquanto tamanho(Q) > 0 faça
11          u = remove(Q)
12          para v em adj(u) faça
13              se não v.visitado então
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visitado = True
17                  insere(Q, v)
```

Algoritmo BFS

```
 1  bfs(G, s):
 2      para u em V(G) faça
 3          u.visitado = False
 4          u.d = ∞
 5          u.p = None
 6      s.visitado = True
 7      s.d = 0
 8      Q = Fila()
 9      insere(Q, s)
10      enquanto tamanho(Q) > 0 faça
11          u = remove(Q)
12          para v em adj(u) faça
13              se não v.visitado então
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visitado = True
17                  insere(Q, v)
```

Análise da complexidade:

Algoritmo BFS

```
 1  bfs(G, s):
 2      para u em V(G) faça
 3          u.visitado = False
 4          u.d = ∞
 5          u.p = None
 6      s.visitado = True
 7      s.d = 0
 8      Q = Fila()
 9      insere(Q, s)
10      enquanto tamanho(Q) > 0 faça
11          u = remove(Q)
12          para v em adj(u) faça
13              se não v.visitado então
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visitado = True
17                  insere(Q, v)
```

Análise da complexidade:

- Cada vértice adicionado uma vez na fila (linha 17): $O(n)$

Algoritmo BFS

```
 1  bfs(G, s):
 2      para u em V(G) faça
 3          u.visitado = False
 4          u.d = ∞
 5          u.p = None
 6      s.visitado = True
 7      s.d = 0
 8      Q = Fila()
 9      insere(Q, s)
10      enquanto tamanho(Q) > 0 faça
11          u = remove(Q)
12          para v em adj(u) faça
13              se não v.visitado então
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visitado = True
17                  insere(Q, v)
```

Análise da complexidade:

- Cada vértice adicionado uma vez na fila (linha 17): $O(n)$
- A lista de adjacência de cada vértice percorrida uma vez (linha 12): $O(m)$

Algoritmo BFS

```
 1  bfs(G, s):
 2      para u em V(G) faça
 3          u.visitado = False
 4          u.d = ∞
 5          u.p = None
 6      s.visitado = True
 7      s.d = 0
 8      Q = Fila()
 9      insere(Q, s)
10      enquanto tamanho(Q) > 0 faça
11          u = remove(Q)
12          para v em adj(u) faça
13              se não v.visitado então
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visitado = True
17                  insere(Q, v)
```

Análise da complexidade:

- Cada vértice adicionado uma vez na fila (linha 17): $O(n)$
- A lista de adjacência de cada vértice percorrida uma vez (linha 12): $O(m)$
- Complexidade total: $O(n + m)$

Bibliografia Utilizada

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. e STEIN, C. *Introduction to Algorithms*, 3ª edição, MIT Press, 2009.