Breadth-first search

Letícia Rodrigues Bueno

Federal University of ABC (UFABC)

Erdös Number - the geek version of Bacon number :)

- Paul Erdös: famous Hungarian mathematician;
- He worked with hundreds of colleagues;
- Published more than 1.400 articles;
- Erdös number is an amused attribute created by his friends;
- Paul Erdös has Erdös number 0;
- Direct collaborators have number 1;
- Collaborators of collaborators have number 2 and so on;

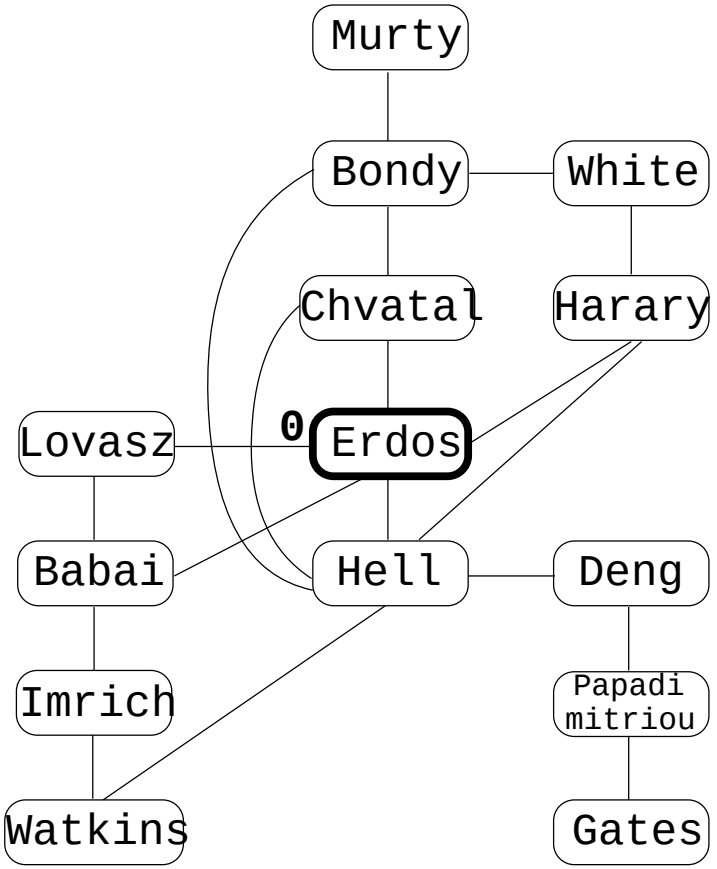
Definition

- Given a list of people and the relation of collaboration between them, which is Erdös number of each person?

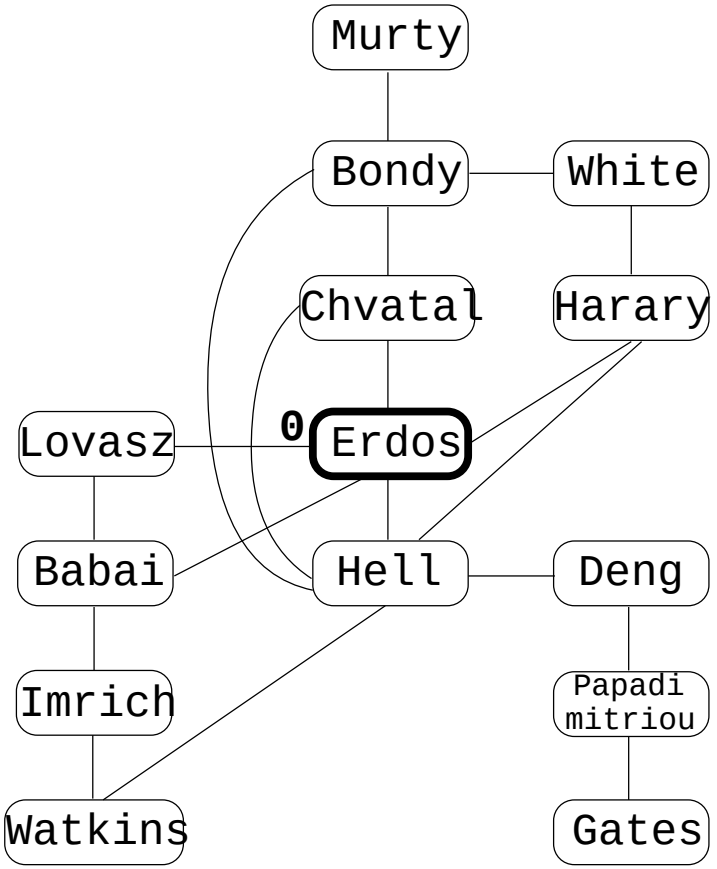
Definition

- Given a list of people and the relation of collaboration between them, which is Erdös number of each person?
- This problem can be modeled as a graph where:
  - People are vertices;
  - Collaborations are edges.

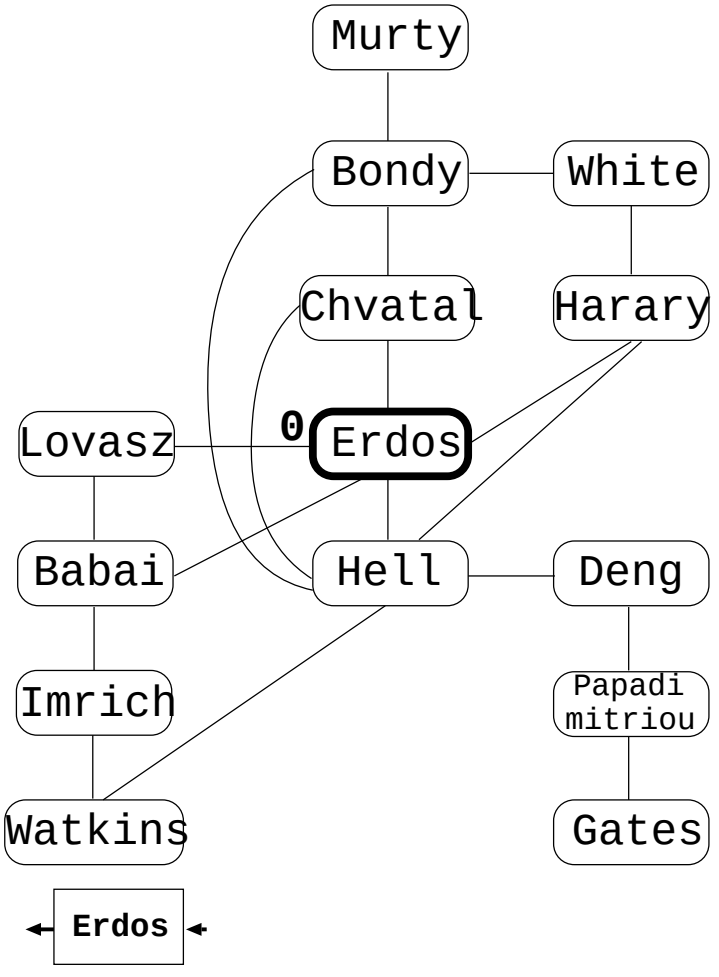
Example


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

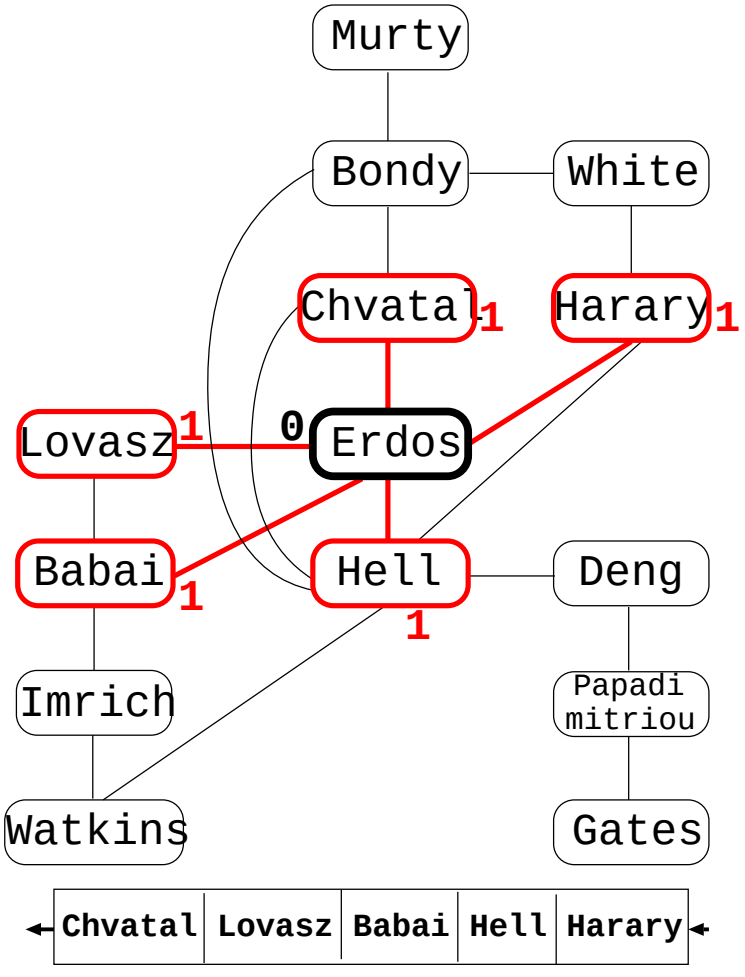
Breadth-first search (BFS)



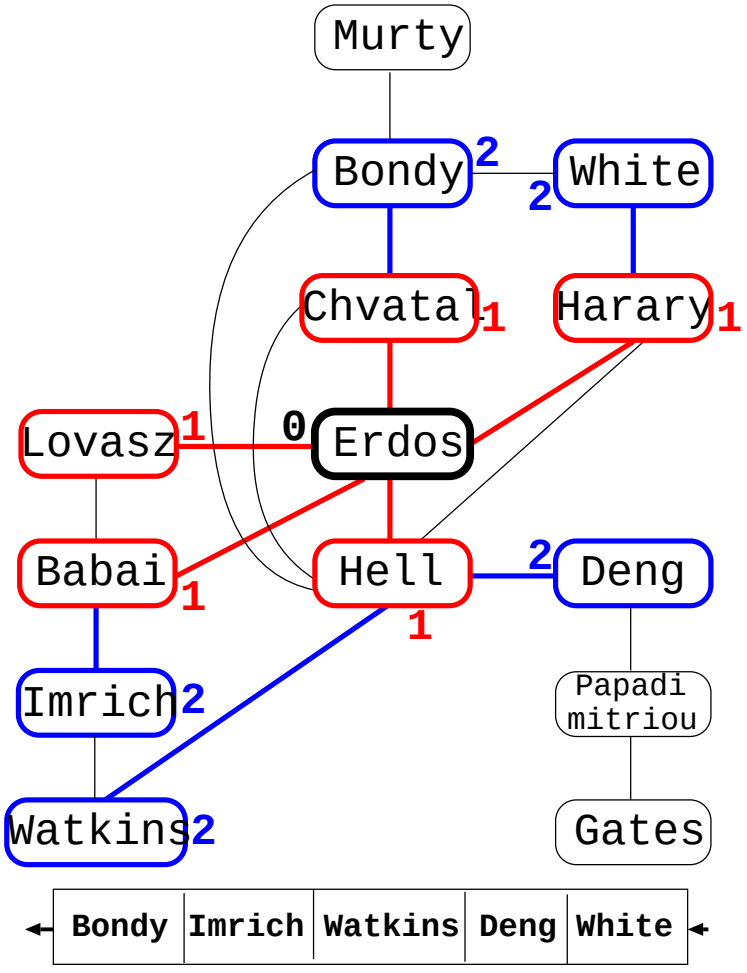
Breadth-first search (BFS)



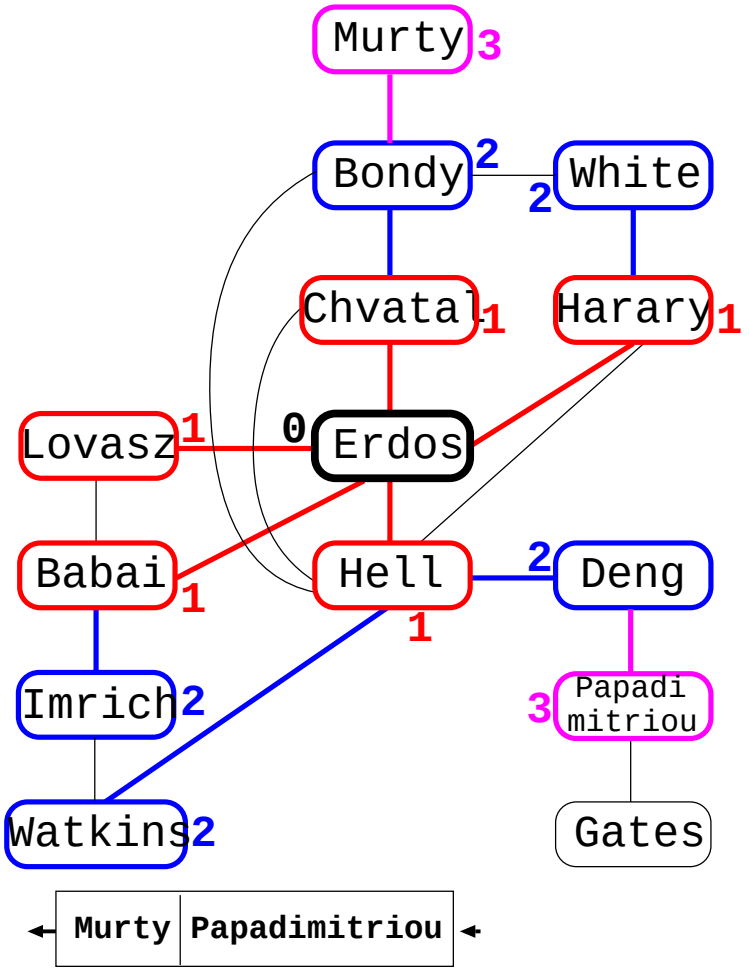
Breadth-first search (BFS)



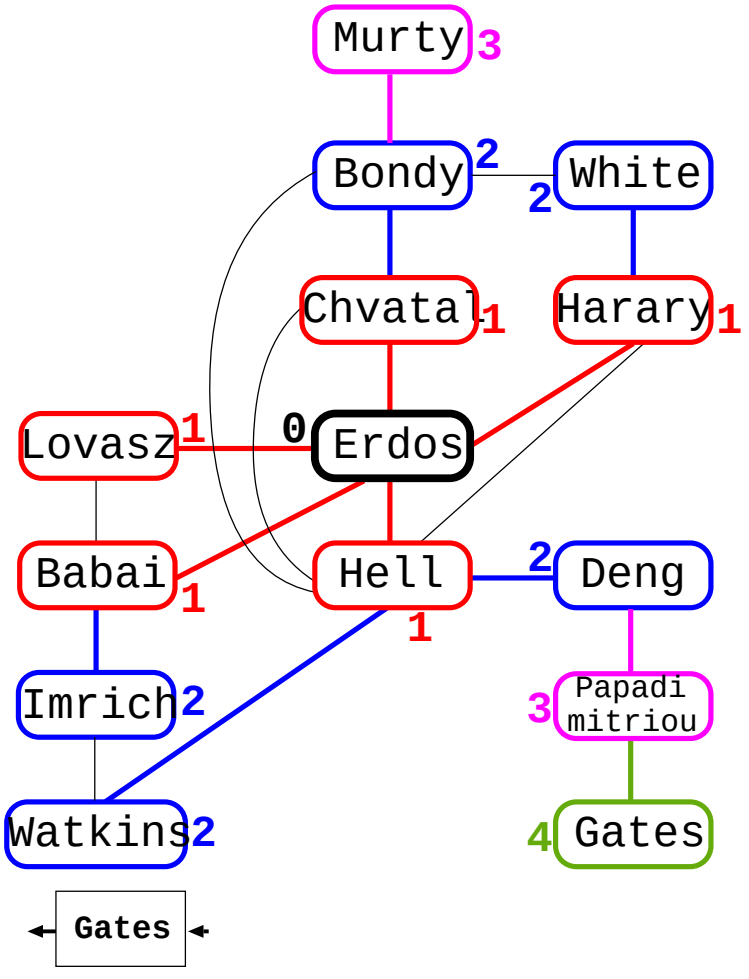
Breadth-first search (BFS)



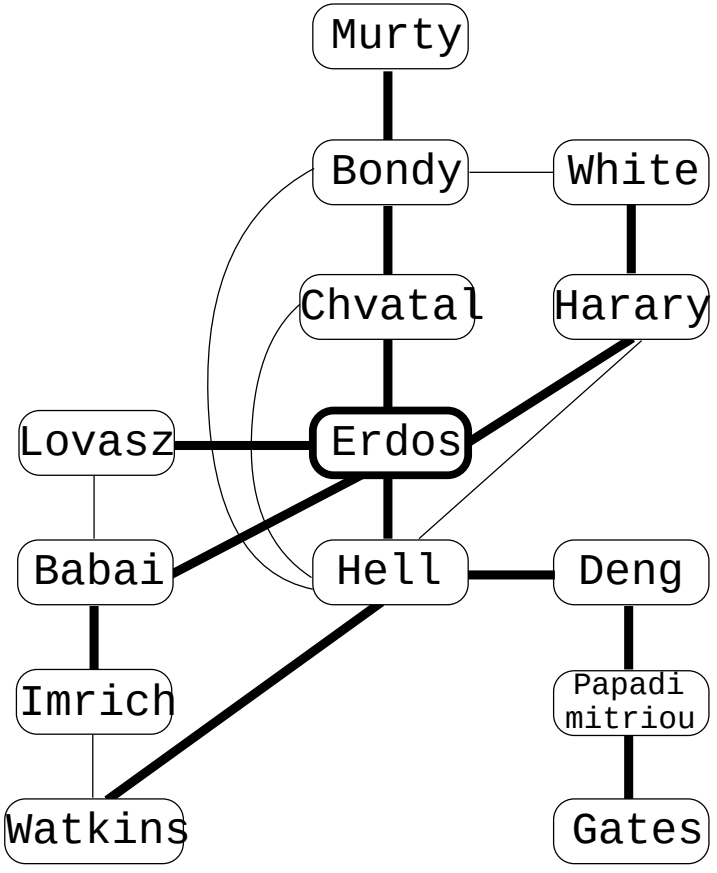
Breadth-first search (BFS)



Breadth-first search (BFS)



Breadth-first search (BFS)



Algorithm BFS

```
 1  bfs(G, s):
 2      for u in V(G) do
 3          u.visited = False
 4          u.d = ∞
 5          u.p = None
 6      s.visited = True
 7      s.d = 0
 8      Q = Queue()
 9      insert(Q, s)
10      while length(Q) > 0 do
11          u = remove(Q)
12          for v in adj(u) do
13              if not v.visited then
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visited = True
17                  insert(Q, v)
```

Algorithm BFS

```
 1  bfs(G, s):
 2      for u in V(G) do
 3          u.visited = False
 4          u.d = ∞
 5          u.p = None
 6      s.visited = True
 7      s.d = 0
 8      Q = Queue()
 9      insert(Q, s)
10      while length(Q) > 0 do
11          u = remove(Q)
12          for v in adj(u) do
13              if not v.visited then
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visited = True
17                  insert(Q, v)
```

Analysis of Complexity:

Algorithm BFS

```
 1  bfs(G, s):
 2      for u in V(G) do
 3          u.visited = False
 4          u.d = ∞
 5          u.p = None
 6      s.visited = True
 7      s.d = 0
 8      Q = Queue()
 9      insert(Q, s)
10      while length(Q) > 0 do
11          u = remove(Q)
12          for v in adj(u) do
13              if not v.visited then
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visited = True
17                  insert(Q, v)
```

Analysis of Complexity:

- Each vertex is added once in the queue (line 17): $O(n)$;

Algorithm BFS

```
 1  bfs(G, s):
 2      for u in V(G) do
 3          u.visited = False
 4          u.d = ∞
 5          u.p = None
 6      s.visited = True
 7      s.d = 0
 8      Q = Queue()
 9      insert(Q, s)
10      while length(Q) > 0 do
11          u = remove(Q)
12          for v in adj(u) do
13              if not v.visited then
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visited = True
17                  insert(Q, v)
```

Analysis of Complexity:

- Each vertex is added once in the queue (line 17): $O(n)$;
- Adjacency list of each vertex is traversed once (line 12): $O(m)$;

Algorithm BFS

```
 1  bfs(G, s):
 2      for u in V(G) do
 3          u.visited = False
 4          u.d = ∞
 5          u.p = None
 6      s.visited = True
 7      s.d = 0
 8      Q = Queue()
 9      insert(Q, s)
10      while length(Q) > 0 do
11          u = remove(Q)
12          for v in adj(u) do
13              if not v.visited then
14                  v.d = u.d + 1
15                  v.p = u
16                  v.visited = True
17                  insert(Q, v)
```

Analysis of Complexity:

- Each vertex is added once in the queue (line 17): $O(n)$;
- Adjacency list of each vertex is traversed once (line 12): $O(m)$;
- Total complexity: $O(n+m)$.

Bibliography

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. and STEIN, C. *Introduction to Algorithms*, 3rd edition, MIT Press, 2009.