

Breadth-first search

From Wikipedia, the free encyclopedia

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph) and explores the neighbor nodes first, before moving to the next level neighbours. Compare BFS with the equivalent, but more memory-efficient Iterative deepening depth-first search and contrast with depth-first search.

BFS was invented in the late 1950s by E. F. Moore, who used to find the shortest path out of a maze,^[1] and discovered independently by C. Y. Lee as a wire routing algorithm (published 1961).^[2]

Contents

- 1 Pseudocode
- 2 Analysis
 - 2.1 Time and space complexity
 - 2.2 Completeness and optimality
- 3 Applications
 - 3.1 Testing bipartiteness
- 4 See also
- 5 References
- 6 External links

Pseudocode

Input: A graph *G* and a vertex *v* of *G*

Output: All vertices reachable from *v* labeled as discovered

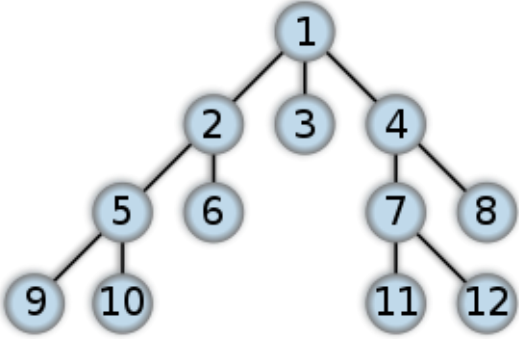
A non-recursive implementation of BFS:

```
1 procedure BFS(G, v) is
2   let Q be a queue
3   Q.push(v)
4   label v as discovered
5   while Q is not empty
6     v ← Q.pop()
7     for all edges from v to w in G.adjacentEdges(v) do
8       if w is not labeled as discovered
9         Q.push(w)
10        label w as discovered
```

The non-recursive implementation is similar to depth-first search but differs from it in two ways: it uses a queue instead of a stack, and it checks whether a vertex has been discovered before pushing the vertex rather than delaying this check until the vertex is popped from the stack.

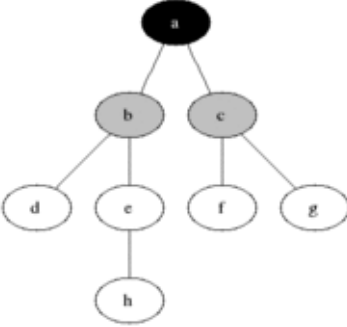
Analysis

Breadth-first search



Order in which the nodes are expanded

Class	Search algorithm
Data structure	Graph
Worst case performance	$O(E) = O(b^d)$
Worst case space complexity	$O(V) = O(b^d)$



Animated example of a breadth-first search

Time and space complexity

The time complexity can be expressed as $O(|V| + |E|)$ ^[3] since every vertex and every edge will be explored in the worst case. Note: $O(|E|)$ may vary between $O(1)$ and $O(|V|^2)$, depending on how sparse the input graph is.

When the number of vertices in the graph is known ahead of time, and additional data structures are used to determine which vertices have already been added to the queue, the space complexity can be expressed as $O(|V|)$ where $|V|$ is the cardinality of the set of vertices. If the graph is represented by an Adjacency list it occupies $\Theta(|V| + |E|)$ ^[4] space in memory, while an Adjacency matrix representation occupies $\Theta(|V|^2)$.^[5]

Completeness and optimality

In the analysis of algorithms, the input to breadth-first search is assumed to be a finite graph, represented explicitly as an adjacency list or similar representation. However, in the application of graph traversal methods in artificial intelligence the input may be an implicit representation of an infinite graph. In this context, a search method is described as being complete if it is guaranteed to find a goal state if one exists. Breadth-first search is complete, but depth-first search is not: when applied to infinite graphs represented implicitly, it may get lost in parts of the graph that have no goal state and never return.^[6]

A search method is optimal if it is guaranteed to find the best solution that exists. In other words, it will find the path to the goal state that involves the least number of steps. Breadth-first search is an optimal search method, but depth-first search is not^[6]

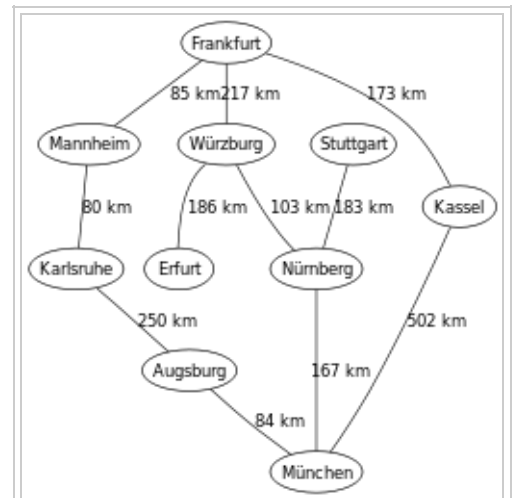
Applications

Breadth-first search can be used to solve many problems in graph theory, for example:

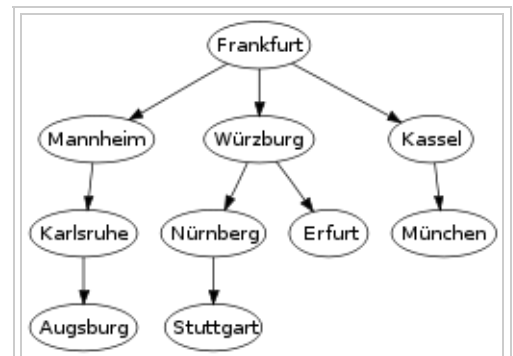
- Copying Collection, Cheney's algorithm
- Finding the shortest path between two nodes u and v (with path length measured by number of edges)
- Testing a graph for bipartiteness
- (Reverse) Cuthill–McKee mesh numbering
- Ford–Fulkerson method for computing the maximum flow in a flow network
- Serialization/Deserialization of a binary tree vs serialization in sorted order, allows the tree to be re-constructed in an efficient manner.
- Construction of the *failure function* of the Aho–Corasick pattern matcher.

Testing bipartiteness

BFS can be used to test bipartiteness, by starting the search at any vertex and giving alternating labels to the vertices visited during the search. That is, give label 0 to the starting vertex, 1 to all its neighbours, 0 to those neighbours' neighbours, and so on. If at any step a vertex has (visited) neighbours with the same label as itself, then the graph is not



An example map of Germany with some connections between cities



The breadth-first tree obtained when running BFS on the given map and starting in Frankfurt

bipartite. If the search ends without such a situation occurring, then the graph is bipartite.

See also

- Depth-first search
- Iterative deepening depth-first search
- Level structure
- Lexicographic breadth-first search

References

- ↑ Skiena, Steven (2008). *The Algorithm Design Manual*. Springer. p. 480. doi:10.1007/978-1-84800-070-4_4 (https://dx.doi.org/10.1007%2F978-1-84800-070-4_4).
 - ↑ Leiserson, Charles E.; Schardl, Tao B. (2010). *A Work-Efficient Parallel Breadth-First Search Algorithm (or How to Cope with the Nondeterminism of Reducers)* (http://web.mit.edu/neboat/www/papers/LeisersonSc10.pdf). ACM Symp. on Parallelism in Algorithms and Architectures.
 - ↑ Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. p.597
 - ↑ Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. p.590
 - ↑ Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. p.591
 - ↑ ^{*a*} ^{*b*} Coppin, B. (2004). Artificial intelligence illuminated. Jones & Bartlett Learning. pp. 79–80.
- Knuth, Donald E. (1997), *The Art Of Computer Programming Vol 1. 3rd ed.* (http://www-cs-faculty.stanford.edu/~knuth/taocp.html), Boston: Addison-Wesley, ISBN 0-201-89683-4

External links

- Breadth-First Explanation and Example (http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch14.html#QQ1-46-92)
- Open Data Structures - Section 12.3.1 - Bread-First Search (http://opendatastructures.org/versions/edition-0.1e/ods-java/12_3_Graph_Traversal.html#SECTION0015310000000000000000)



Wikimedia Commons has media related to ***Breadth-first search***.

Retrieved from "http://en.wikipedia.org/w/index.php?title=Breadth-first_search&oldid=648771980"

Categories: Graph algorithms | Search algorithms

- This page was last modified on 25 February 2015, at 10:53.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.