Math Notes on Breadth-First Search

Breadth-first search (BFS) is a search strategy that models the growth of a fire in which the search proceeds radially outward from the initial search locations. The search maintains a queue of grid cells that are on the boundary of the searched area. To expand the search, the method repeatedly dequeues the first cell of this boundary queue and then, for each neighbor of this cell, checks whether this neighbor cell has not been visited during the search. If the neighbor has not been visited, the cell is marked as visited and this cell is enqueued on the boundary queue. This demo illustrates the basic behavior of breadth-first search on a grid.

Recording whether a cell has been visited can be done efficiently using a grid. The grid cells are initialized to `EMPTY` to indicate that the cells have not been visited. Each time a cell is visited, the corresponding entry in the grid is set to `FULL`.

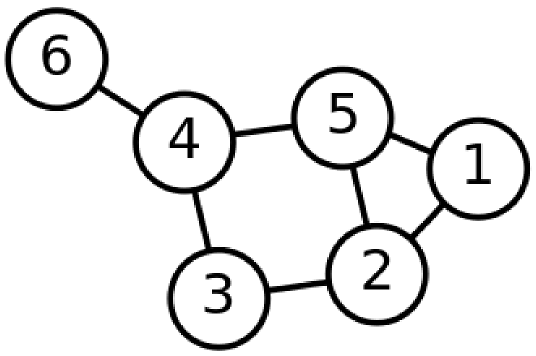
At the start of the search, the queue **`boundary`** is initialized to contain one or more cells. The entries in the grid **`visited`** corresponding to these cells should also be set as `FULL` to indicate that these cells have been visited by the search. With that background, here is a high-level description of the search in English:

```
while boundary is not empty:
    current_cell  ←  dequeue boundary
    for all neighbors neighbor_cell of current_cell:
        if neighbor_cell is not in visited:
            add neighbor_cell to visited
            enqueue neighbor_cell onto boundary
```

Note that use of a queue in this search causes the search to be radial in nature. At any point in the search, breadth-first search process all of the cells on the current boundary of the search before processing any new cells added to the search boundary. Replacing the queue by a stack in BFS yields an entirely different kind of search strategy call *depth=first search (DFS)*. During depth-first search, the search propagates outward in one preferred direction until it can no longer proceed in that direction.

Graphs

Both breadth-first search and depth-first search are also applicable to more general types of data structures. A graph is a data structure that consists of two set of objects; a collection of *nodes* and a collection of *edges* that link pairs of neighboring nodes. The nodes and edges in a graph can be visualized as a collection of labeled circles (nodes) and line segments (edges) connecting neighboring nodes. This diagram (from the Wikipedia) shows a graph with six nodes and seven edges.



Note that breadth-first search can be applied to graphs with almost no modification. If we represent the neighbors of a node using data structure like a list or dictionary, implementing breadth-first search for graphs is simple. The follow up class to this course, "Algorithmic Thinking", will focus heavily on the topic of graph search.

Mark as completed