Graph traversal and Path-finding Algorithms used in Video Games

Marius Rabenarivo

[AlgoMada]

A brief history of AlgoMada

2021	2022	2023	PRESENT
Association creation	4 Contests organized, Newbie, Rookie and Pro	Clean Code Hackathon +Webinar, Bot Programming Contest, Participation to Indabax	AlgoGames

About me

- Telecommunications, ESPA Alumni
- Computer Science, University of Reunion Island Alumni
- FaceDev Admin since 2012
- Founder member of AlgoMada
- Clojure dev
- Computer Science Enthusiast
- Current interests: Cryptocurrency, Clojure programming language
- Side project: BetaX Community

Definition of Computer Science

"We are about to study the idea of a computational process. Computational processes are abstract beings that inhabit computers. As they evolve, processes manipulate other abstract things called data. The evolution of a process is directed by a pattern of rules called a program. People create programs to direct processes. In effect, we conjure the spirits of the computer with our spells." Structure and Interpretation of Computer Programs, Harold Abelson and Gerald J. Sussman

What is a graph?

A data structure to represent link between objects. A graph is defined by a set of nodes V and a set of edges E.

We can summarize this definition by the following formula:

$$G = (E, V)$$

Example: https://www.redblobgames.com/pathfinding/grids/graphs.html#properties

What's the difference between a graph and a tree?

A graph can contain cycles (a node can be visited twice).

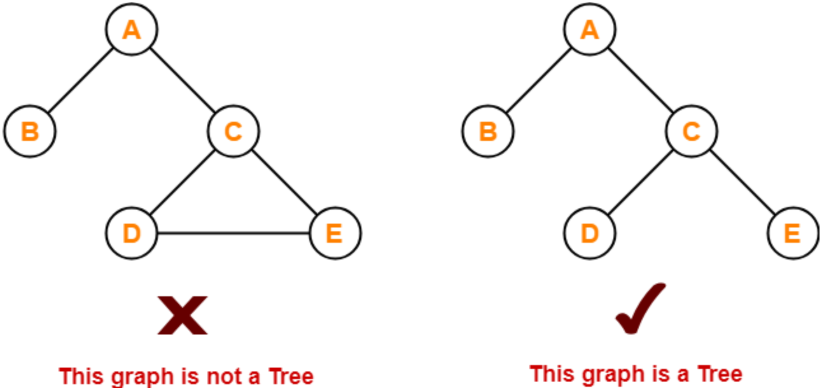


Figure 1: Tree vs Graph

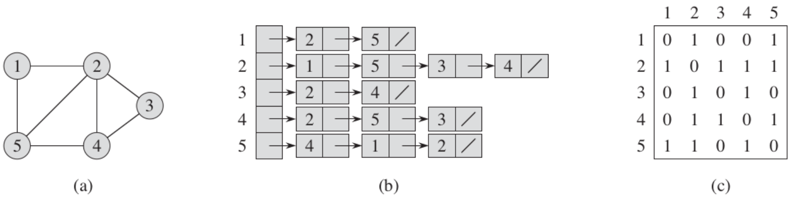
Different type of graphs

- Acyclic Graph A graph that has no cycle.
- Cyclic Graph A graph that has at least one cycle.
- Directed Graph A graph in which edge has direction. That is the nodes are ordered pairs in the definition of every edge.
- Undirected Graph A graph in which edge are not directed. Meaning, the edges are defined by an unordered pair of nodes.
- Directed Acyclic Graph A graph that is both directed and acyclic.
- Connected graph Every pair of nodes has a path linking them. Put in another way, there are no inaccessible node.
- Disconnected graph A graph in which there is at least one inaccessible node.
- A multigraph A graph that can have multiple edges between the same nodes.

Different way to represent a graph

There are 2 ways to represent a graph:

- adjacency list For each node, provide a list of other nodes that are adjacent to it.
- adjacency matrix A matrix construct by aligning the nodes in the row and the columns and putting a value if the nodes are linked by an edge.



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Figure 2: (a) Undirected graph with 5 vertices and 7 edges (b) Adjacency-list representation (c) Adjacency-matrix representation

Graph traversal algorithms

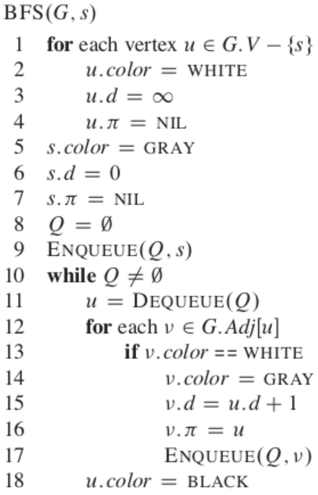
BFS (Breadth-First Search)

A graph traversal algorithm in which one explore every possible node in the current depth level before going to the next. Usually used to find shortest path distance from the start to a given vertex and the associated predecessor subgraph.

DFS (Depth-First Search)

A graph traversal algorithm in which one start with a root node (arbitrarily chosen) then explore as far as possible along each branch before backtracking. Usually used as a subroutine in another algorithm.

BFS (Breadth-First Search)



```
BFS($G, s$)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$Q = \emptyset$
ENQUEUE($Q, s$)
while $Q \neq \emptyset$
    $u$ = DEQUEUE($Q$)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
            $v.color$ = GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE($Q, v$)
    $u.color$ = BLACK
```

Figure 3: Breadth-first search pseudo-code

BFS (Breadth-First Search)

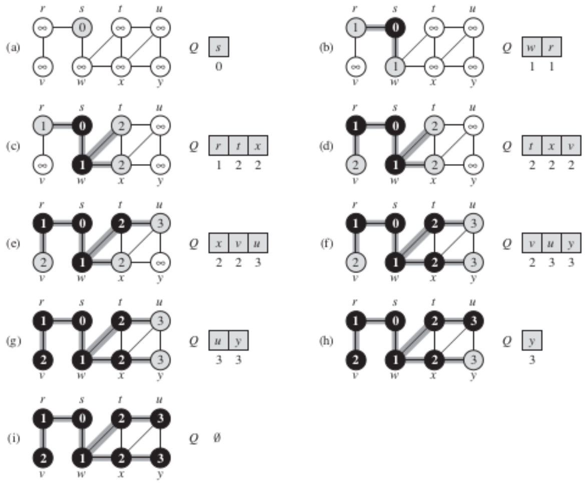
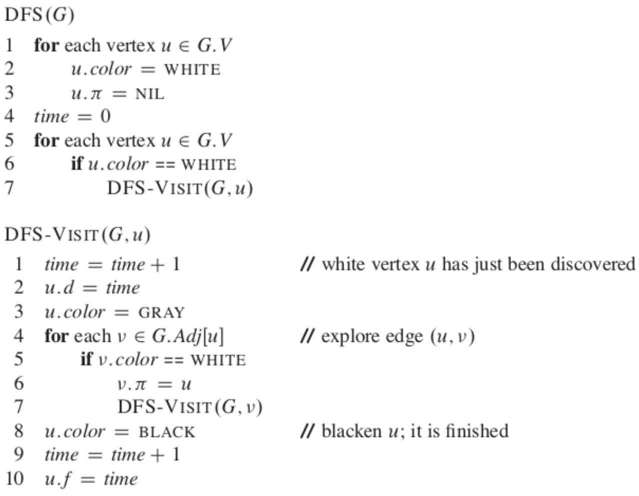


Figure 4: Operation of BFS on an undirected graph

Depth-First Search



```
DFS($G$)
for each vertex $u \in G.V$
    $u.color$ = WHITE
    $u.\pi$ = NIL
$time = 0$
for each vertex $u \in G.V$
    if $u.color$ == WHITE
        DFS-VISIT($G, u$)

DFS-VISIT($G, u$)
$time = time + 1$              // white vertex $u$ has just been discovered
$u.d = time$
$u.color$ = GRAY
for each $v \in G.Adj[u]$      // explore edge $(u, v)$
    if $v.color$ == WHITE
        $v.\pi = u$
        DFS-VISIT($G, v$)
$u.color$ = BLACK              // blacken $u$; it is finished
$time = time + 1$
$u.f = time$
```

Figure 5: Depth-First Search Pseudocode

Depth-First Search

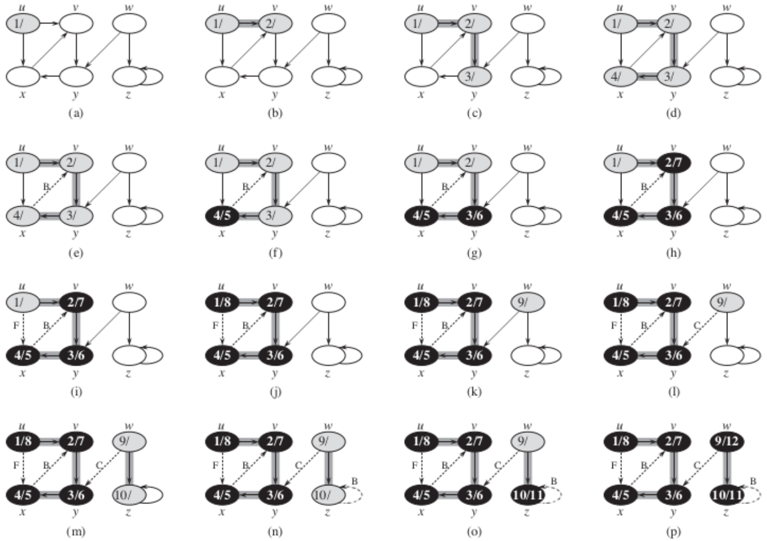


Figure 6: Depth-First Search progress on a directed graph

Path finding algorithms

A* algorithm

A* (pronounced “A-Star”) is a graph traversal and path-finding algorithm. Given a source and a goal node, the algorithm find the shortest-path (with respect to given weights) from source to goal.

Dijkstra algorithm

Dijkstra algorithm solves the single-source shortest-paths problem on a weighted directed graph for the case in which all weights are non-negative.

Path-finding

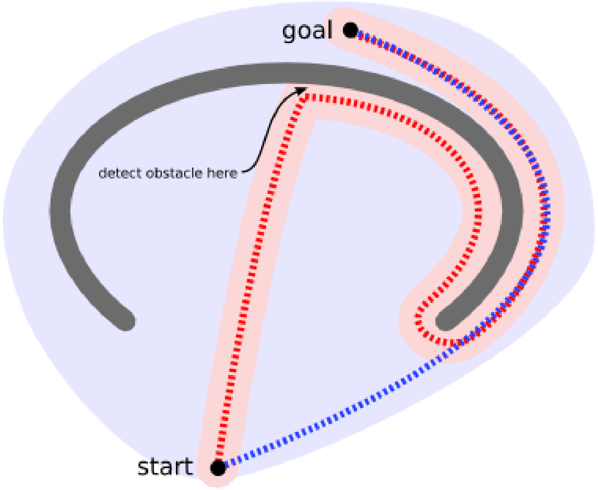


Figure 7: Example path-finding situation

Path-finding

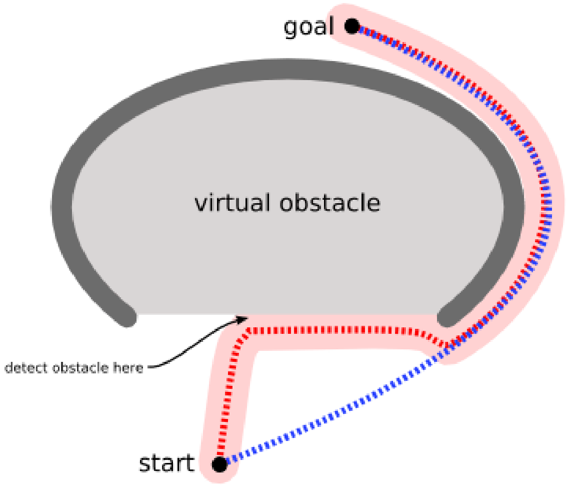


Figure 8: Example path-finding situation

A* Algorithm
History

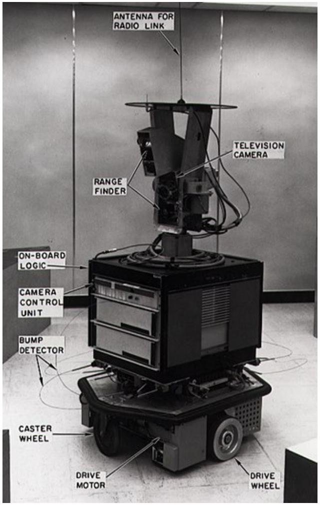


Figure 9: A* was invented by researchers working on Shakey the Robot's path planning.

Application in Video Games

For 2D video games, a tile map can be transformed into a graph. Each cell of the grid will be a node in the graph and the edges are going to be the four directions: east, north, west, south.

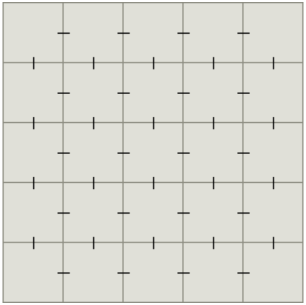


Figure 10: Map as graph

Example: https://www.redblobgames.com/pathfinding/grids/graphs.html#grids

Dijkstra's Algorithm

Dijkstra's Algorithm works by visiting vertices in the graph starting with the object's starting point. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined.

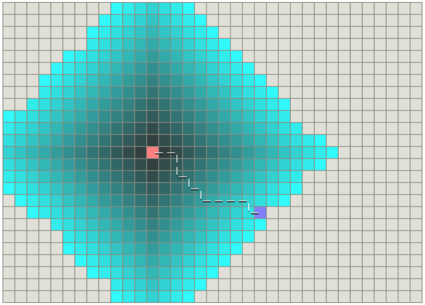


Figure 11: Dijkstra algorithm

Gready Best-First search

The Greedy Best-First-Search algorithm works in a similar way, except that it has some estimate (called a heuristic) of how far from the goal any vertex is. Instead of selecting the vertex closest to the starting point, it selects the vertex closest to the goal. Greedy Best-First-Search is not guaranteed to find a shortest path.

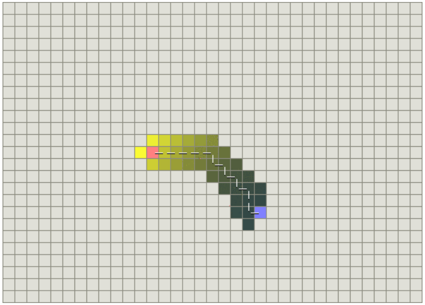


Figure 12: Greedy Best-first search

Dijkstra's Algorithm and Best-First-Search

Let's consider the concave obstacle as described in the previous section. Dijkstra's Algorithm works harder but is guaranteed to find a shortest path:

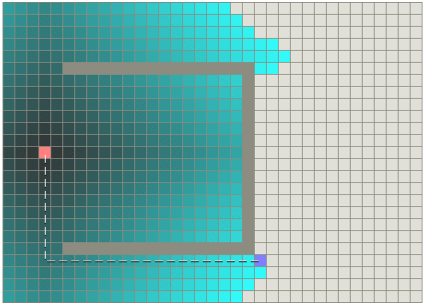


Figure 13: Dijkstra's Algorithm with obstacle

Dijkstra's Algorithm and Best-First-Search

Greedy Best-First-Search on the other hand does less work but its path is clearly not as good:

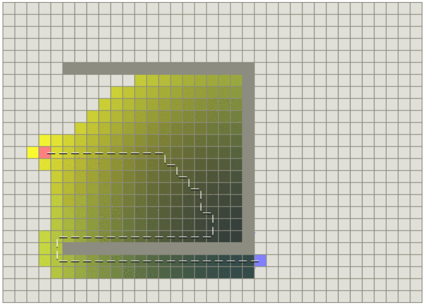


Figure 14: Greedy Best-First Search with trap

The A* Algorithm

A* is like Dijkstra's Algorithm in that it can be used to find a shortest path. A* is like Greedy Best-First-Search in that it can use a heuristic to guide itself. In the simple case, it is as fast as Greedy Best-First-Search:

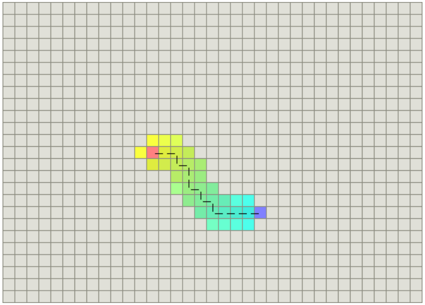


Figure 15: A* algorithm

References

- https://en.wikipedia.org/wiki/A*_search_algorithm
- http://theory.stanford.edu/~amitp/GameProgramming/AStarCompa