Hierarchical Path-Finding

Addie Audette, Bug Lee, Annorah Lewis, Luke Marks

December 12, 2022

Table of contents

Path finding

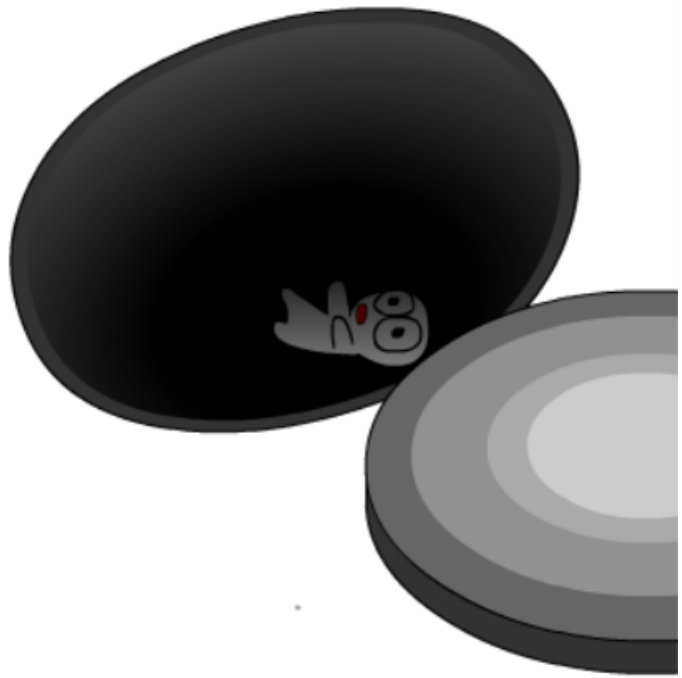
You are given

- Starting location S
- Destination location D

Path finding

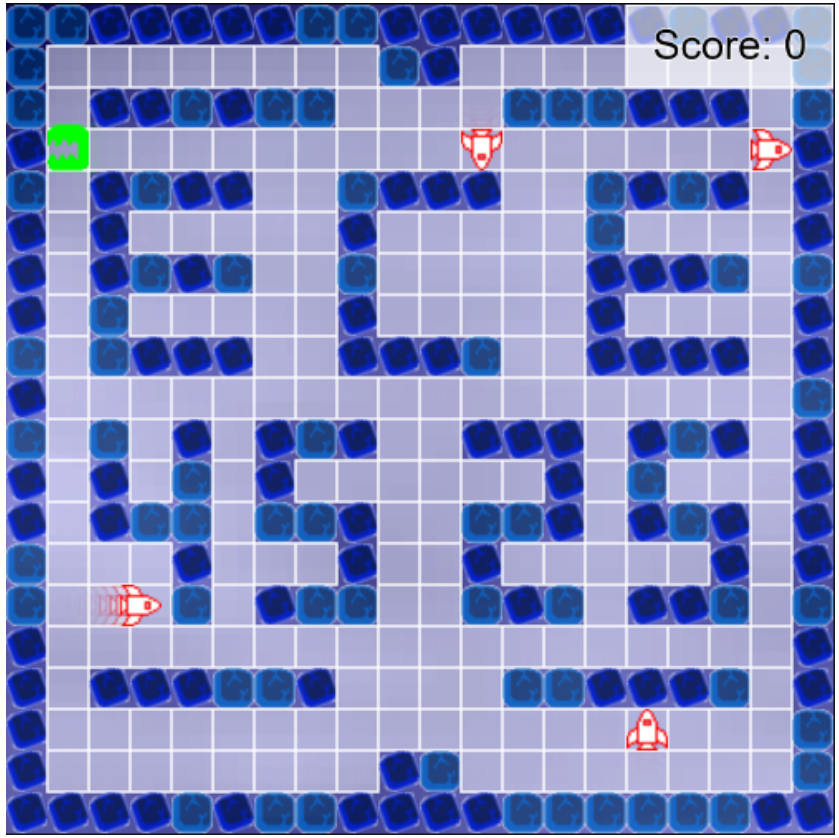
You want to avoid path from S to D that are

- Impossible
- Dangerous
- Unnecessary



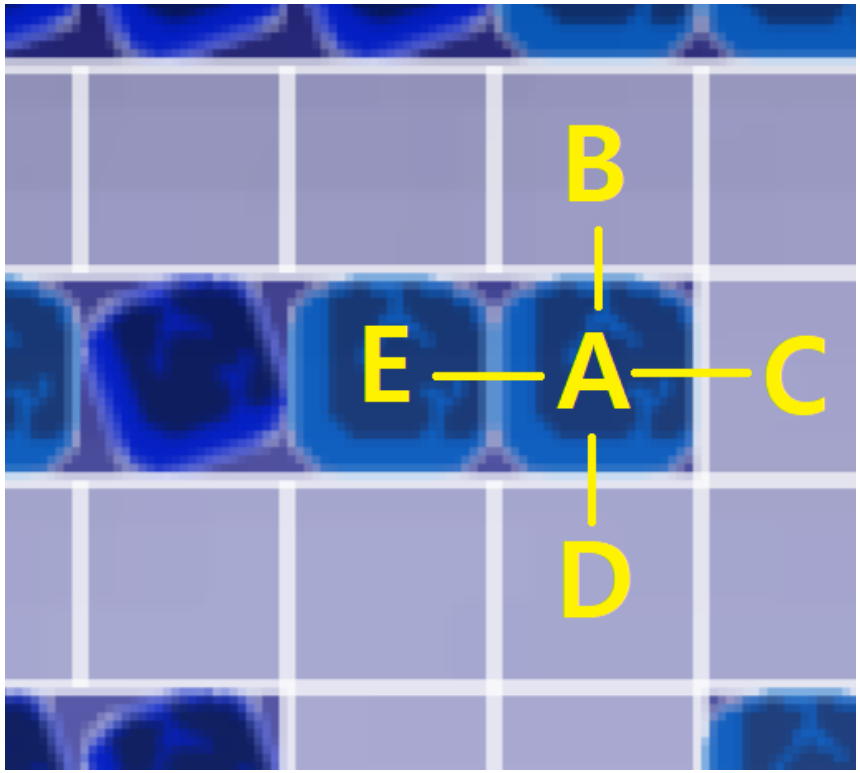
Viewing world with grid graph

Imagine world is made of grid, like mindcraft...



Viewing world with grid graph

Each vertex have degree 4 and edges are undirected with weight 1.



We will come back to the grid graph later...

Dijkstra

- Dijkstra algorithm finds the shortest path to all the vertices from source s, given that the graph $G = (V, E)$ contains only non-negative weight for all edges[3].
- In other words, it forms the tree that represents the shortest paths to all of the vertices in the graph.

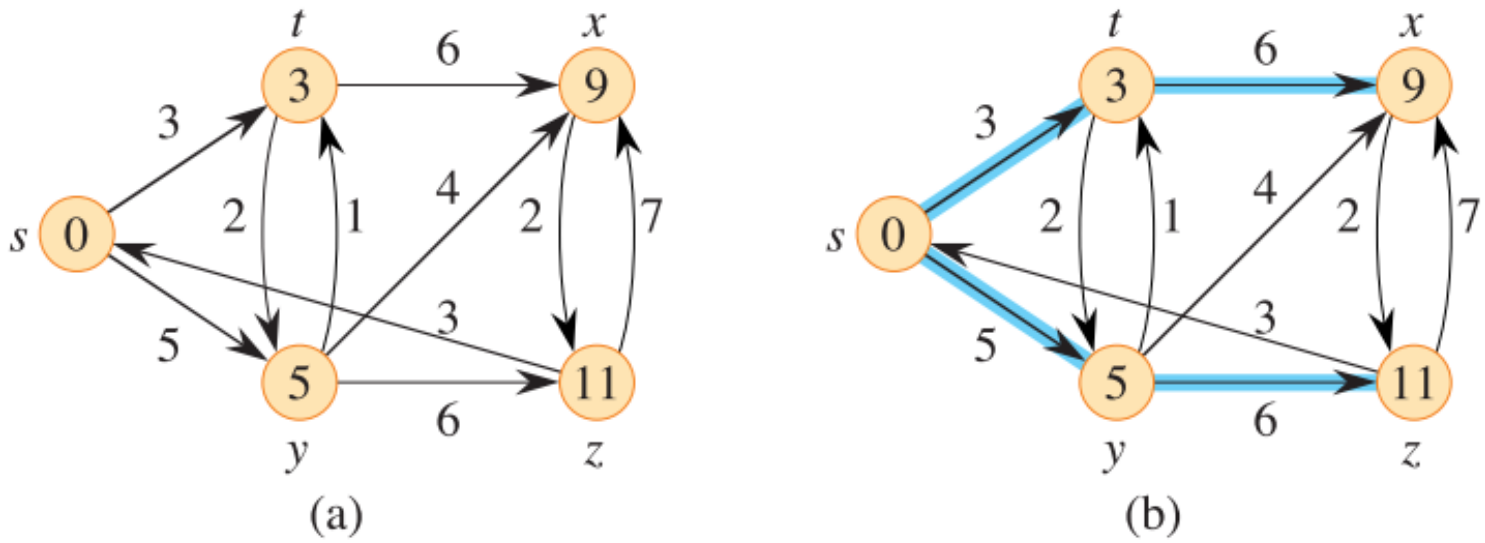


Figure: (a) A weighted, directed graph with source s. (b) The blue edges represent the shortest-path tree rooted at the source s. The figure was taken from *Introduction to Algorithms* by CLRS[3].

A*

- "Dijkstra with a twist" [2]
- Dijkstra algorithm blindly selects vertex with minimum distance from s each step. Instead, make a clever guess in each step where the algorithm selects a vertex that is likely part of the shortest path from s to t.[2, 4]

A*: Clever guess?

For A* to work correctly and efficiently, the A* algorithm must guess each step that

- Heuristic:
  Minimize unnecessary computation on finding sub-paths that are obviously not part of the optimal path[4], but also
- Admissiblity:
  Should not ignore the sub-path that can be part of the optimal path[4].

Shortcomings

In practice, a naive A* algorithm is still not sufficient for many modern applications.

1. Many modern applications require computation to happen in real-time for hundreds, if not thousands, users/agents simultaneously[1].
2. The shift to mobile applications has put more limitations on memory and CPU usage[1].

Using hierarchy to reduce compleixty

The idea of Hierarchical Path Finding is to form a region by clustering the vertices and introducing high-level regional routes.

1. Compute a local (low-level) route to the source to the source region entrance,
2. Compute a regional (high-level) route from the source region to destination region, and
3. Compute a local (low-level) route to destination region exit to destination.

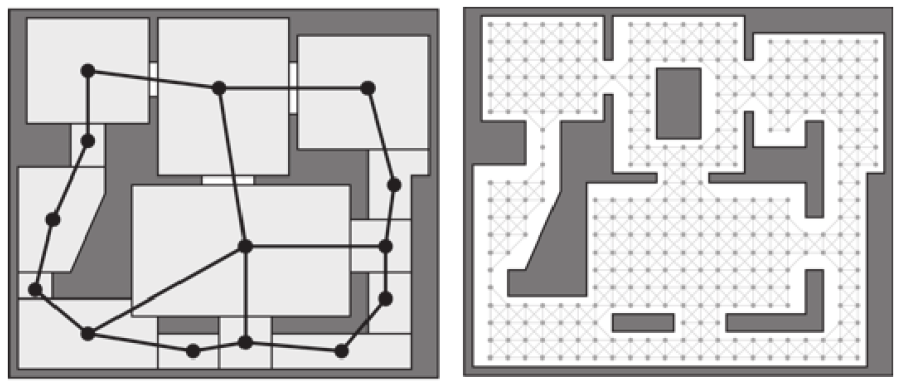


Figure: Forming regions (high-level graph) by clustering neighboring vertices. The figure was taken from *Programming Game AI by Example* by Buckland[2].

Challenges

Compare to Dijkstra and naive A*, there are more tunable variables that implementers need to consider

- The number of hierarchy levels,
- Cluster/region size, and
- Placement of regional entrances/exits.
- In practice, optimizations like preprocessing/caching regional routes and path smoothing may be necessary.

Simplification

To avoid overwhelming the algorithm with optimization details, our implementation assume the following simplifications:

1. An input graph is static and known in advance,
2. 2 level of the hierarchy,
3. Randomly distributes regional entrances/exits in reachable locations,
4. No preprocessing/caching, and
5. No path refinement or smoothing.

Experiment setup

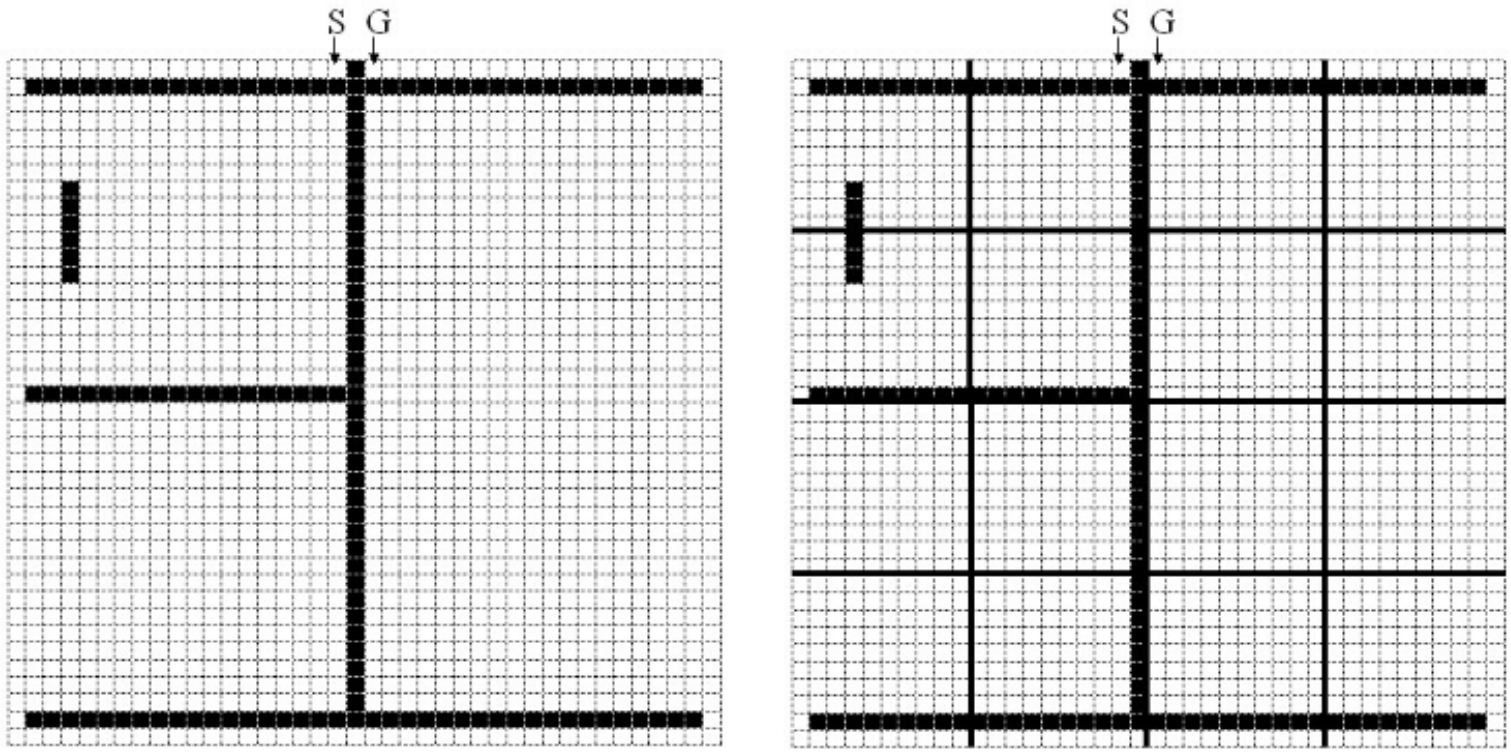


Figure: (a) The 40 X 40 maze used in our example. The obstacles are painted in black. S and G are the start and the goal nodes. (b) The bold lines show the boundaries of the 10x 10 clusters[1].

Experiment results

Algorithm	# of vertices visited	Time (ms)
Dijkstra	0	0
A*	0	0
Hierarchical Path Finding	0	0

** We are still working on experimentation. We hope to provide experiment results during revision.

Limitations

- Hierarchical Path Finding does not guarantee the shortest path between the source and destination.
- The *Near Optimal Hierarchical Path-Finding* apply a path-smoothing procedure to make a path found by Hierarchical Path Finding within 1% optimal compared to shortest path[1].

References


A. Botea, M. Müller, and J. Schaeffer.
Near optimal hierarchical path-finding.
J. Game Dev., 1:1–30, 2004.


M. Buckland.
Programming Game AI by Example, pages 241–247, 359–360, 373.
Wordware Publishing, 2005.


T. Cormen, C. Leiserson, R. Rivest, and C. Stein.
Introduction to Algorithms, pages 620–624.
The MIT Press, 4th ed edition, 2022.


P. Hart, N. Nilsson, and B. Raphael.
A formal basis for the heuristic determination of minimum cost paths.
Transctions on systems science and cybernetics, pages 100–107, 1968.