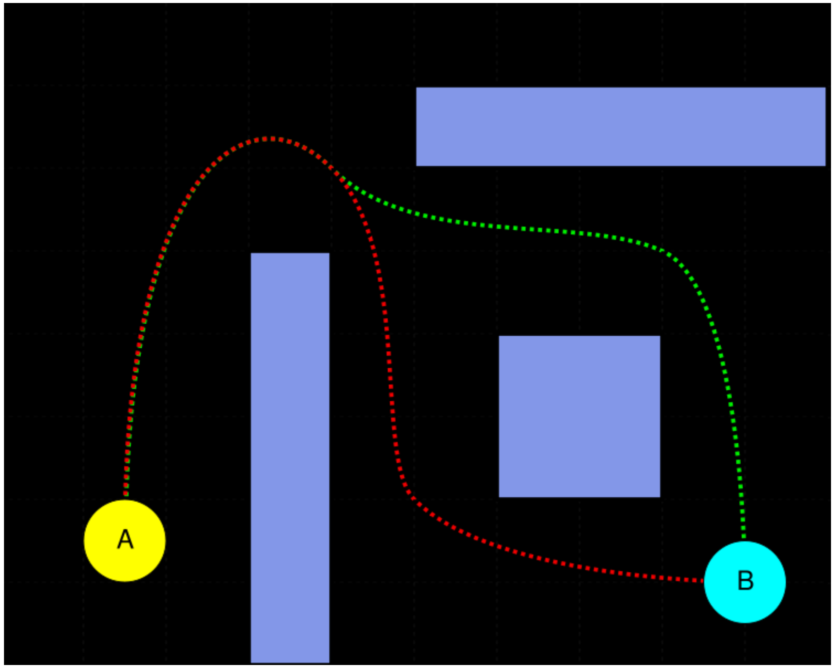
VGP332 – Artificial Intelligence

Instructor: Peter Chan



Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

Assignment 1 Redux

- Questions?

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

Review

- What is pathfinding?
- What can graphs be used for?
- How can you implement graphs?

Graph Searches

- Open list:
  - list of nodes you need to consider in the steps ahead
- Closed list:
  - list of nodes you’ve already visited and don’t need to consider again

Graph Searches

```
put start node in open list
while end node not reached && open list isn’t empty:
    move node N from open list to closed list
    expand node N:
      if expanded node isn’t in open or closed lists:
             add expanded node to open list
```

DFS, BFS

- **DFS:**
- Choose node from front of open list
- Expanded nodes get added to front of open list

- **BFS:**
- Choose node from front of open list
- Expanded nodes get added to back of open list

Agenda

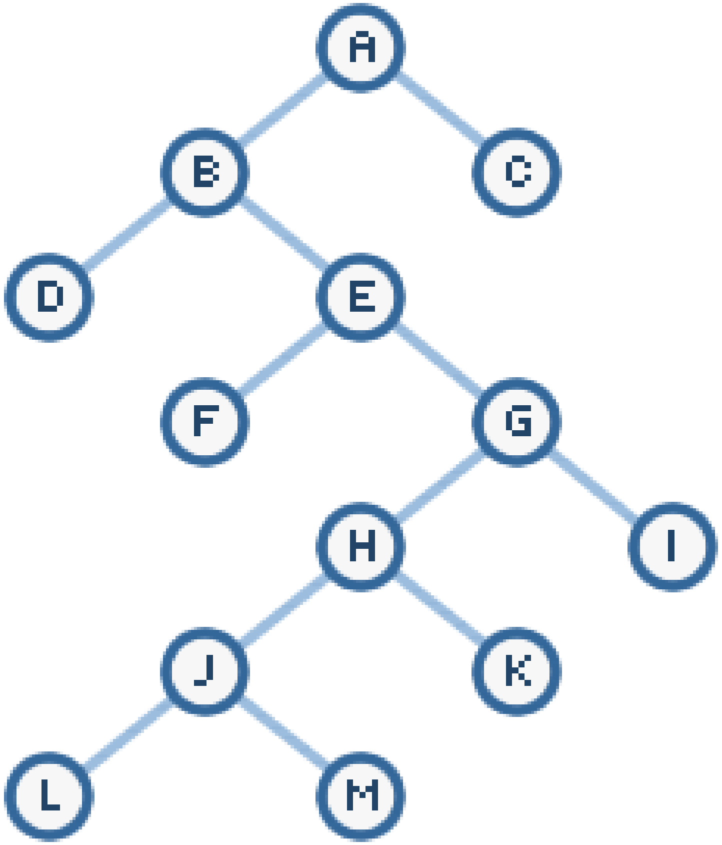
- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

Shortest Paths

- What is the shortest path between two nodes on a graph?

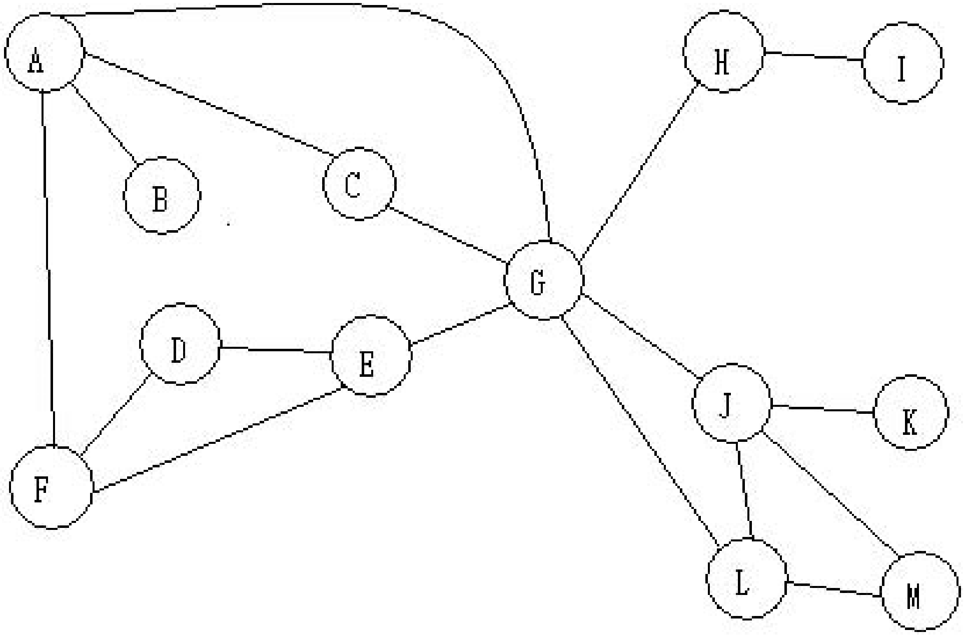
Shortest Paths

- Consider this graph:



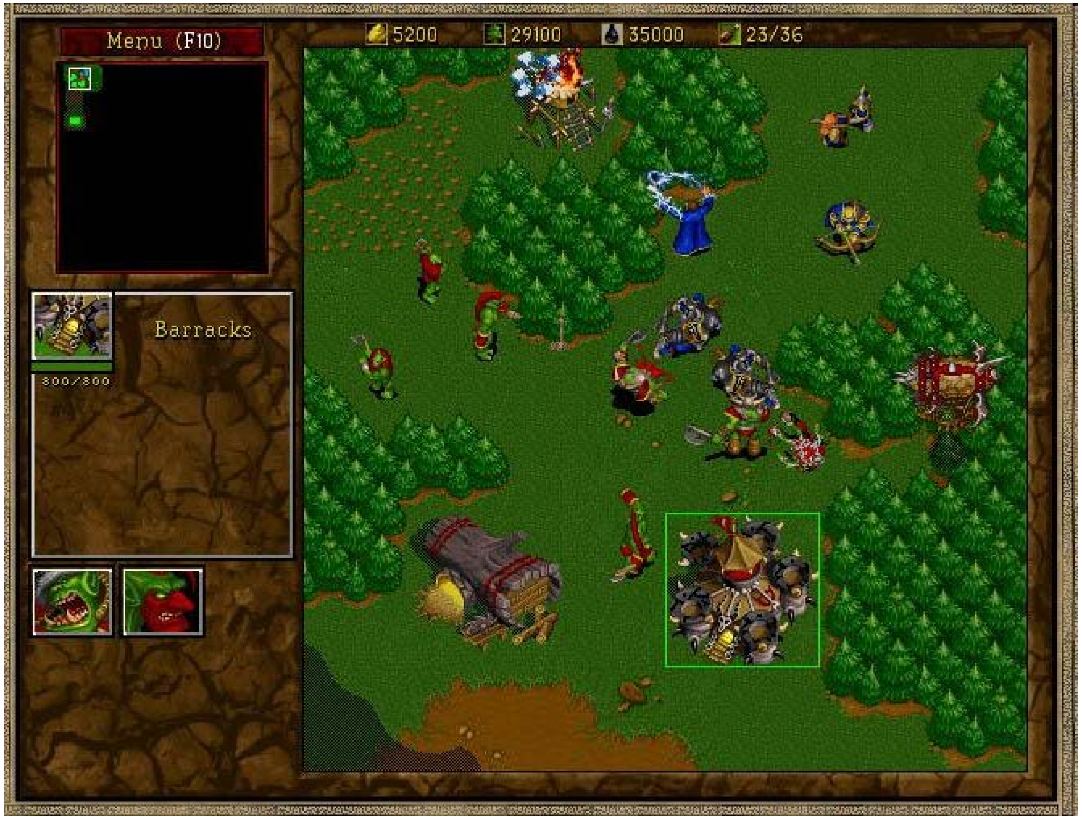
Shortest Paths

- Consider this graph:



Shortest Paths

- Consider this map:



Dijkstra’s Algorithm

- Choose node from front of open list that is the shortest cumulative distance from the starting node
- Expanded nodes get added to ~~front of~~ open list

Dijkstra’s Algorithm

- Choose node from front of open list that is the shortest cumulative distance from the starting node

This is called a **cost function**

- Expanded nodes get added to ~~front of~~ open list

Dijkstra’s Algorithm

- Cost function

$f(x) = g(x)$

where $g(x)$ is shortest distance traveled from initial node to current node

Dijkstra’s Algorithm

http://en.wikipedia.org/wiki/Dijkstra_algorithm

Dijkstra’s Search Applet

- Try this out:

http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/Dijkstra.shtml

Dijkstra’s Search on Grid-Based

	A	B	C	D	E
1					
2			▨		
3	S		▨		E
4			▨		
5					

Observations on Dijkstra’s Search

- Pros?
- Cons?
- Results?

Observations on Dijkstra’s Search

- Optimal
- Informed

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

A* Algorithm

- Choose node from front of open list that is on the shortest path from the starting node to the ending node
- Expanded nodes get added to ~~front of~~ open list

A* Algorithm

- Choose node from front of open list that is on the shortest path from the starting node to the ending node

Anything odd about this cost function?

- Expanded nodes get added to ~~front of~~ open list

A* Algorithm

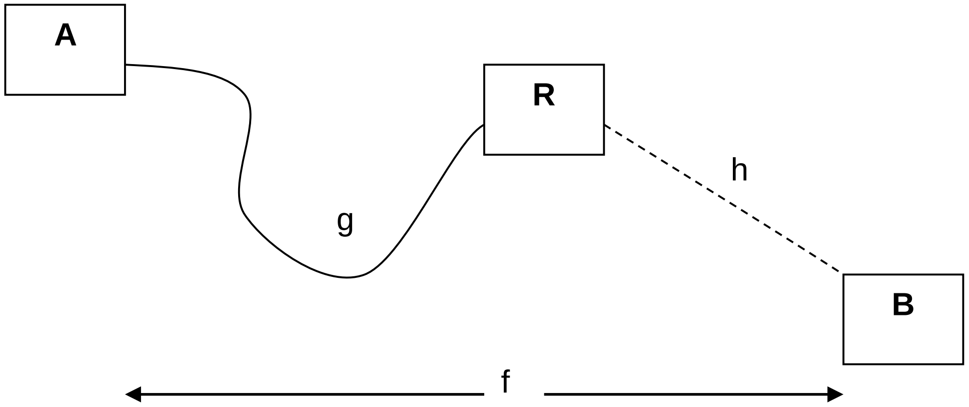
- Q: If you’re searching for a path but haven’t found one yet, how can you know how long the full path is?

A* Algorithm

- Q: If you’re searching for a path but haven’t found one yet, how can you know how long the full path is?

- A: Make an informed guess

A* Algorithm



- For any path you’re considering from A to B
  - Suppose you’re as far along as node R
  - You know how far you’ve come: distance = g
  - Best case from R to B: distance = h

A* Algorithm

- Cost function

$f(x) = g(x) + h(x)$

where $g(x)$ is shortest distance traveled from initial node to current node and $h(x)$ is the guessed distance from the current node to the end node

A* Algorithm

http://en.wikipedia.org/wiki/A-star_algorithm

A* Search on Grid-Based

	A	B	C	D	E
1					
2			▨		
3	S		▨		E
4			▨		
5					

A* Search Applet

- Try this out:

http://www.vision.ee.ethz.ch/~cvcourse/astar/AStar.html

Observations on A* Search

- Pros?
- Cons?
- Results?

Observations on A* Search

- Optimal
- Informed
- h(x) needs to be admissible

This means that the estimate must be less than or equal to the actual lowest cost

A* Search Heuristics

- Can you think of other heuristics to use?

Search Algorithm Comparison

- Try this out:

http://www.stefan-baur.de/cs.web.mashup.pathfinding.html

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

Graph Search Improvements

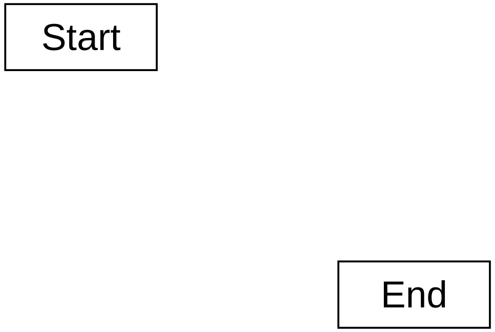
- What improvements can you think of for:
  - BFS
  - DFS
  - Dijkstra’s
  - A*

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview

“Best-first” search

- Favour paths that would lead towards the goal
- E.g.



Which search directions are preferable in this case?

- Can combine with A* search’s cost function

Minimax Search

- Usually used in turn-based two-player games
- Scoring function evaluates board/moves
- Create branching tree of move possibilities
- Maximize AI’s move and minimize opponent’s move

http://wolfey.110mb.com/GameVisual/launch.php

Alpha-beta pruning

- Optimization for minimax searches
- Prune (don’t generate/evaluate) unnecessary branches

Minimax & Alpha-Beta Example

- Petri
- Blob Wars

Agenda

- Assignment 1 Redux
- Review
- Dijkstra’s Algorithm
- A* Algorithm
- Graph Search Improvements
- Alternative Graph Searches
- Assignment 2 Overview