APSP

Matthew Guzdial

guzdial@ualberta.ca

Announcements

- Quiz 1 released on Friday @ 11am (available for 48 hours), Quiz 2 two Fridays away.
- No Lecture Friday, but I'll be in the room to help (will also be checking Discord/email)
- We'll cover Quiz 1 answers on Monday.
- HW1 grades up this weekend. This will include feedback on where you lost points (if you did).
  - Waiting for a few students who had extensions.

Affordances of/Problems with A*

- Comparison to greedy path search
- Computational cost in a dynamic environment
- Computational cost in a static environment

Participation Question

https://forms.gle/EWdd7YMeXqfKzwfi9

https://tinyurl.com/guz-pq8

What could we do instead of running A* every time we need to calculate a path in a static environment?

Answer

All Pairs Shortest Path! (Don’t put this as an answer to the question, as you probably wouldn’t know it)

Generally, precompute shortest paths between nodes and save that info.

All pairs shortest path (APSP)

- A look-up table of the form table[node1,node2]-> node 3
  - Where node3 is the next node to go to if you want to go from node1 to node2
- Intuition: Find the shortest distance/path between all pairs of nodes
  - Use this to construct the look-up table

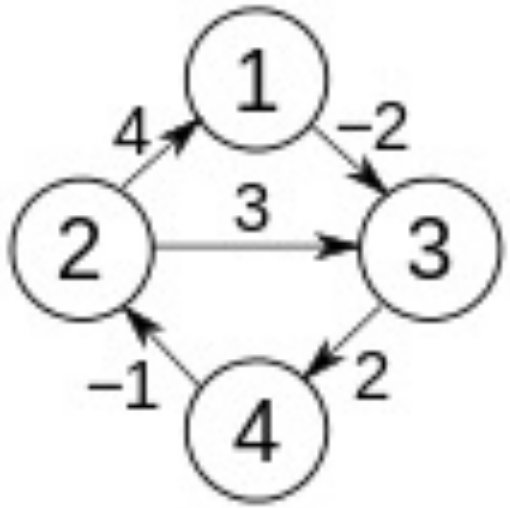
Floyd –Warshall algorithm part 1

let **dist** be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each vertex v:
    dist[v][v] = 0
for each edge (u,v):
    dist[u][v] = w(u,v) // the weight of the edge (u,v)
for k from 1 to |V|:
    for i from 1 to |V|:
        for j from 1 to |V|:
            if **dist**[i][j] > **dist**[i][k] + **dist**[k][j]:
                dist[i][j] = **dist**[i][k] + **dist**[k][j]:

Floyd –Warshall algorithm part 2

```
for each edge (u,v):
      dist[u][v] = w(u,v)
      next[u][v] = v
for k from 1 to |V|:
     for i from 1 to |V|:
          for j from 1 to |V|:
               if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]
                    next[i][j] = next[i][k]
```

Example



$k=0$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	3	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0

$k=1$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	**2**	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0

$k=2$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	2	∞
	3	∞	∞	0	2
	4	**3**	−1	**1**	0

$k=3$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	**0**
	2	4	0	2	**4**
	3	∞	∞	0	2
	4	3	−1	1	0

$k=4$		j			
		1	**2**	**3**	**4**
i	**1**	0	**−1**	−2	0
	2	4	0	2	4
	3	**5**	**1**	0	2
	4	3	−1	1	0

When to use A* and APSP?

Static: Paths, once computed, will never be invalid

Dynamic: Paths may become invalid while pathing

When the environment is small and static?
Nintendogs (2005) Walks

APSP

When the the environment is small and dynamic?
Into the Breach (2018)

When the environment is large and static?

Majora's Mask (2000)

Depends on runtime memory!

When the map is large and dynamic?
Age of Empires IV (2021)

Sometimes A*, but often more complex approaches!

Quiz Review

Topics/Lectures

- Intro to Game AI
- Grids
- Path Networks
- Nav Meshes
- Path smoothing + Steering
- A*
- High level concepts (path weighting, new steering techniques, targeting approaches, dynamic path nodes) from Sunset Overdrive Video
- APSP (concept and A* comparison)