Report First Project IAJ

João Vitór Sebastião Carvalho Tiago Antunes

2023-09-21

Contents

1 Introduction

The goal of this project was to create different levels of path finding algorithms, and compare their performance.
We compared 4 differente algorithms: Basic A*(unordered list for open set, unordered list for closed set), Basic A* but using tiebreaking (unordered list for open set, unordered list for closed set), NodeArray A* (NodeArray for open and closed set) and NodeArray A* with Goal Bounding.

2 Basic A*

2.1 Algorithm

2.2 Data

Table 1: Basic A* performance (Path 1)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	21432.88
GetBestAndRemove	2890	114.64
AddToOpen	3020	2.54
SearchInOpen	28291	759.53
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	2890	1.76
SearchInClosed	27990	20319.63
RemoveFromClosed	0	0

Table 2: Basic A* performance (Path 2)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	21432.88
GetBestAndRemove	2890	114.64
AddToOpen	3020	2.54
SearchInOpen	28291	759.53
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	2890	1.76
SearchInClosed	27990	20319.63
RemoveFromClosed	0	0

3 Basic A* with tiebraking

3.1 Algorithm

3.2 Data

Table 3: Basic A* with tiebraking performance (Path 1)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	10029.02
GetBestAndRemove	1904	112.83
AddToOpen	1954	1.56
SearchInOpen	18564	260.07
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	1904	1.27
SearchInClosed	18460	9524.91
RemoveFromClosed	0	0

Table 4: Basic A* with tiebraking performance (Path 2)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	21970.11
GetBestAndRemove	2890	282.13
AddToOpen	3021	1.94
SearchInOpen	28291	768.2
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	2890	1.61
SearchInClosed	27990	20761.34
RemoveFromClosed	0	0

4 NodeArray A*

4.1 Algorithm

4.2 Data

Table 5: NodeArray A* performance (Path 1)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	5.48
GetBestAndRemove	200	1.96
AddToOpen	216	1.13
SearchInOpen	235	0
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	100	0.01
SearchInClosed	1009	0.04
RemoveFromClosed	0	0

Table 6: NodeArray A* performance (Path 2)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	159.11
GetBestAndRemove	2890	38.36
AddToOpen	3019	9.31
SearchInOpen	28247	1.84
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	2885	0.58
SearchInClosed	27932	1.73
RemoveFromClosed	0	0

5 NodeArray A* with Goal Bounding

5.1 Algorithm

5.2 Data

Table 7: NodeArray A* with Goal Bounding performance (Path 1)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	10.62
GetBestAndRemove	200	0.43
AddToOpen	216	0.34
SearchInOpen	235	0
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	100	0.01
SearchInClosed	126	0
RemoveFromClosed	0	0

Table 8: NodeArray A* with Goal Bounding performance (Path 1)

Method	Calls	Execution Time (ms)
A*Pathfinding.Search	1	17.12
GetBestAndRemove	158	0.40
AddToOpen	165	0.35
SearchInOpen	388	0.01
RemoveFromOpen	0	0
Replace	0	0
AddToClosed	158	0.03
SearchInClosed	282	0.01
RemoveFromClosed	0	0

6 Bonus Level

7 Conclusions

We can infer that A* is pretty slow when compared to it's otimizations.
Also, we can notice that adding pre-processing to the algorithm can improve it's runtime by a lot, even though it takes some time to do it.