Timofey Malko (862311452)

Your challenges in this project.
Performance. In the end any 8 puzzle runs in seconds, but a 15 puzzle takes a while.
Design. Making the algorithm generic, determining relationships between data took multiple rewrites.

Your design.
Search is just a function. Used a graph search.

Search work with instances of “Interfaces”
Node { get_state, get_cost, get_depth, get_parent_state, print, print_line, print_expand },
Problem { solve, expand, is_goal_node, get_init_node }

Npuzzle and NpNode implement the above.

As heuristic function depends on the problem state representation, it is stored inside Npuzzle.

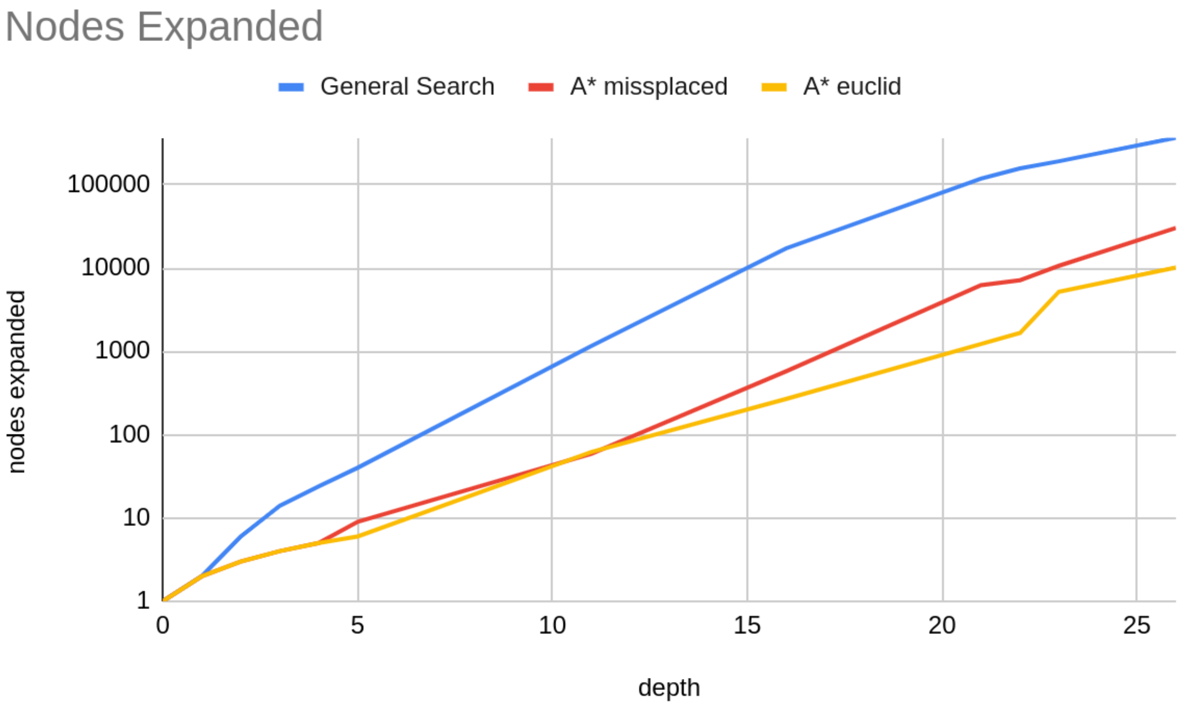
Optimizing
The notable optimization is using a hashset for visited, and use of binary search for look up and insertion for frontier.

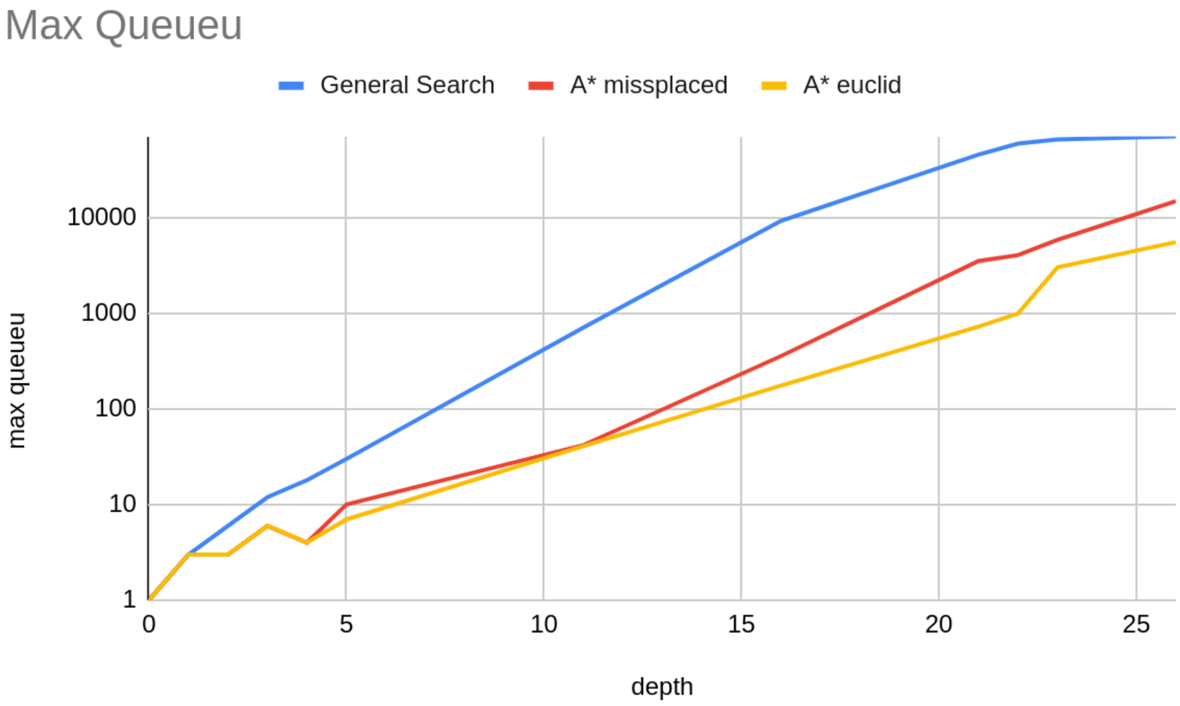
Comparing Heuristic Functions

As seen in graphs, when the depth of the goal is small there is small difference between having and not having a heuristic, and there is almost no difference between implemented heuristics.
There is small to no difference between implemented heuristics
For goals with medium depths, the improvement of use of heuristic is around a magnitude.
For goals with bigger depths, The heuristic outperforms uniform cost by more than a magnitude.
The difference between heuristic fluctuates but there is a clear resource usage decrease with better heuristic.

The euclidean heuristic is a couple times faster than the misplaced tile one.
The heuristic is a hundred times faster than the uniform cost.

Data:


Nodes Expanded
General Search
A* missplaced
A* euclid
nodes expanded
100000
10000
1000
100
10
1
0
5
10
15
20
25
depth


Max Queueu
General Search
A* missplaced
A* euclid
max queueu
10000
1000
100
10
1
0
5
10
15
20
25
depth

Heuristic	General Search	A* missplaced	A* euclid
Total Time	12.54s	0.15s	0.03s

General Search	Nodes Expanded	Max Queue	Depth
Trival	1	1	0
Very Easy	2	3	1
Easy	6	6	2
Ex 2	14	12	3
Doable	24	18	4
Ex 1	40	30	5
My test 11	1162	713	11
My test 16	17289	9315	16
My test 21	118531	45954	21
Oh Boy	157377	59801	22
My test 23	191869	66349	23
My test 26	366615	71268	26

A* missplaced	Nodes Expanded	Max Queue	Depth
Trival	1	1	0
Very Easy	2	3	1
Easy	3	3	2
Ex 2	4	6	3
Doable	5	4	4
Ex 1	9	10	5
My test 11	59	42	11
My test 16	577	358	16
My test 21	6243	3544	21
Oh Boy	7148	4072	22
My test 23	10681	5901	23
My test 26	30338	14949	26

A* euclid	Nodes Expanded	Max Queue	Depth
Trival	1	1	0
Very Easy	2	3	1
Easy	3	3	2
Ex 2	4	6	3
Doable	5	4	4
Ex 1	6	7	5
My test 11	62	41	11
My test 16	269	176	16
My test 21	1227	730	21
Oh Boy	1666	998	22
My test 23	5217	3036	23
My test 26	10141	5576	26