Tianhua Zhao Project 1 Report

2.2.1 Question 1

Report the following information for each of the 18 provided input files:

- Number of A* searches before the car reached the goal.
- Total number of cells expanded by A* across all A* searches.
- Number of Adaptive A* searches before the car reached the goal. (Include the first A* search in this count.)
- Total number of cells expanded by Adaptive A* across all A* searches. (Include the number of cells expanded by the first A* search in this count.)

The first column is the number of A* searches.
The second column is the number of cells expanded by A*.
The third column is the number of Adaptive A* searches.
The fourth column is the number of cells expanded by Adaptive A*.

	A	B	C	D
1	1	2	1	2
2	2	8	2	8
3	1	7	1	7
4	4	21	4	21
5	4	24	4	23
6	8	155	4	41
7	1	14	1	14
8	23	730	26	613
9	5	69	5	69
10	637	49000	637	49000
11	9	150	9	150
12	30	707	30	617
13	1	12	1	12
14	57	1200	57	1038
15	22	543	22	469
16	40	928	40	853
17	8	155	4	41
18	1	9	1	9
19	in favor larger g			

2.2.2 Question 2

What data structure did you choose to use for your priority queue. Explain your reasoning.

I used a binary heap because I want to pop min element efficiently, it faster than a sorted array. Implementing the binary heap, I use an underlying vector of node pointers because the size of the array is unknown at compile time; g value and h value are used for prioritization.

2.2.3 Question 3

Determine experimentally which tie-breaking criteria (in favor of smaller g-values or larger g-values) results in a smaller number of cell expansions. Explain your observation in detail, that is, explain what you observed and give a reason for the observation.
Breaking ties in favor of smaller g-values means that if two cells s1 and s2 in the OPEN list both have the same f-value that is the smallest in the priority queue (i.e., f(s1) = f(s2) = mins∈OP EN {f(s)}) and s1 has a smaller g-value than s2 (i.e., g(s1) < g(s2)), then your algorithm will choose s1 over s2 to pop from the priority queue.

	A	B	C	D
1	1	2	1	2
2	2	8	2	8
3	1	7	1	7
4	4	21	4	21
5	4	24	4	23
6	8	155	4	41
7	1	14	1	14
8	23	730	26	613
9	5	69	5	69
10	637	49000	637	49000
11	9	150	9	150
12	30	707	30	617
13	1	12	1	12
14	57	1200	57	1038
15	22	543	22	469
16	40	928	40	853
17	8	155	4	41
18	1	9	1	9
19	in favor larger g			

	A	B	C	D
1	1	2	1	2
2	2	11	2	11
3	1	19	1	19
4	4	21	4	21
5	4	29	4	28
6	4	58	4	51
7	1	14	1	14
8	26	1100	26	816
9	5	267	5	267
10	637	49000	667	49000
11	9	542	9	542
12	30	719	30	640
13	1	47	1	47
14	51	3504	51	3465
15	22	557	22	494
16	40	4937	40	4931
17	4	58	4	51
18	1	11	1	11
19	in favor smaller g			

By comparison, in general, tie-breaking in favor of larger g value results in fewer planning, and fewer nodes expanded. The reason is that in most cases, heuristics are good estimates of cost to go, tie breaking in favor of larger g would first expand the frontier farthest from the start if there are ties; in contrast, tie breaking in favor of smaller g would first expand the frontier closest to the start, also larger h value is likely to result in larger estimate error from the actual cost to go. However, there are some exceptions. For example in 6.txt A* case. The reason could be that the heuristic of the larger g node is far from actual cost to go.

2.2.4 Question 4

Determine experimentally whether Adaptive A* expands fewer cells than A* and try to determine in which situations the savings are large versus small (or non-existing). Explain your reasoning for your observations.

By comparison, in general, adaptive A* expands fewer cells than A*. The reason is that adaptive A* uses information from the previous search to obtain more informed heuristics. Better heuristics generally results in fewer nodes expanded.