Project 1

Submitted By:
Rajeev Kumar (2011CS1031)

How to run the Code?

Follow the following steps to run the code

1) Change the directory to the folder which contains project1.cpp file.
2) Compile the code using the command g++ project1.cpp on any linux system.
3) Run the code using ./a.out command.
4) The program will ask the depth of solution you want. Put an appropriate number.
5) Now the program will ask you to choose the algorithm. Put an appropriate number.
6) Additional input may be required in DFS and Astar with evaluation Function.
7) The program will show the output of the following quantities
    1. Depth at which solution is found
    2. Number of nodes Expanded
    3. Number of nodes Generated
    4. Maximum depth of expanded nodes
    5. Memory used in bytes
    6. Time taken in microseconds

The following tables depict the performance of different algorithms at different solution depth

Table 1
BFS

Solution Depth	Nodes Expanded	Nodes Generated	Maximum depth expanded	Memory Used (in bytes)	Time in microseconds
3	12	21	3	1512	127
4	18	35	4	2520	147
5	48	77	5	5544	347
6	78	138	6	9936	625
7	126	218	7	15696	1007
8	230	364	8	26208	1848
9	380	606	9	43632	2688
10	508	824	10	59328	1972

Table 2
DFS Depth Limit=20

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Reached	Memory Used (in bytes)	Time in microseconds
3	2585	2597	20	186984	14406

4	38990	38997	20	2807784	781975
5	18140	18152	20	1306944	114847
6	10122	10136	20	729792	52048
7	17406	17416	20	1253952	92050
8	1742	1751	20	126072	9395
9	17602	17615	20	1268280	92594
10	13803	13809	20	994248	69994

Table 3
GBEFS

Solution Depth	Nodes Expanded	Nodes Generated	Maximum depth explored	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	103
4	5	9	4	648	54
5	7	14	5	1008	82
6	93	159	28	11448	887
7	8	16	7	1152	106
8	883	1429	174	102888	9977
9	48	88	25	6336	477
10	1058	1746	208	125712	6470

Table 4
Astar with HeuristicsMisplacedTiles

Solution Depth	Nodes Expanded	Nodes Generated	Maximum depth explored	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	79
4	5	9	4	648	36
5	7	14	5	1008	57
6	9	19	6	1368	83
7	18	31	7	2232	148
8	18	32	8	2304	149
9	28	50	9	3600	327
10	25	42	10	3024	205

Table 5
IDAStar

Solution Depth	Nodes	Nodes	Maximum	Memory Used (	Time in

	Expanded	Generated	depth explored	in bytes)	microseconds
3	4	4	7	288	109
4	5	5	9	360	63
5	7	7	11	504	101
6	10	10	12	720	175
7	18	18	13	1296	244
8	18	18	15	1296	226
9	33	38	15	2736	651
10	28	34	18	2448	456

Table 6
Astar with HeuristicsSumManhatten

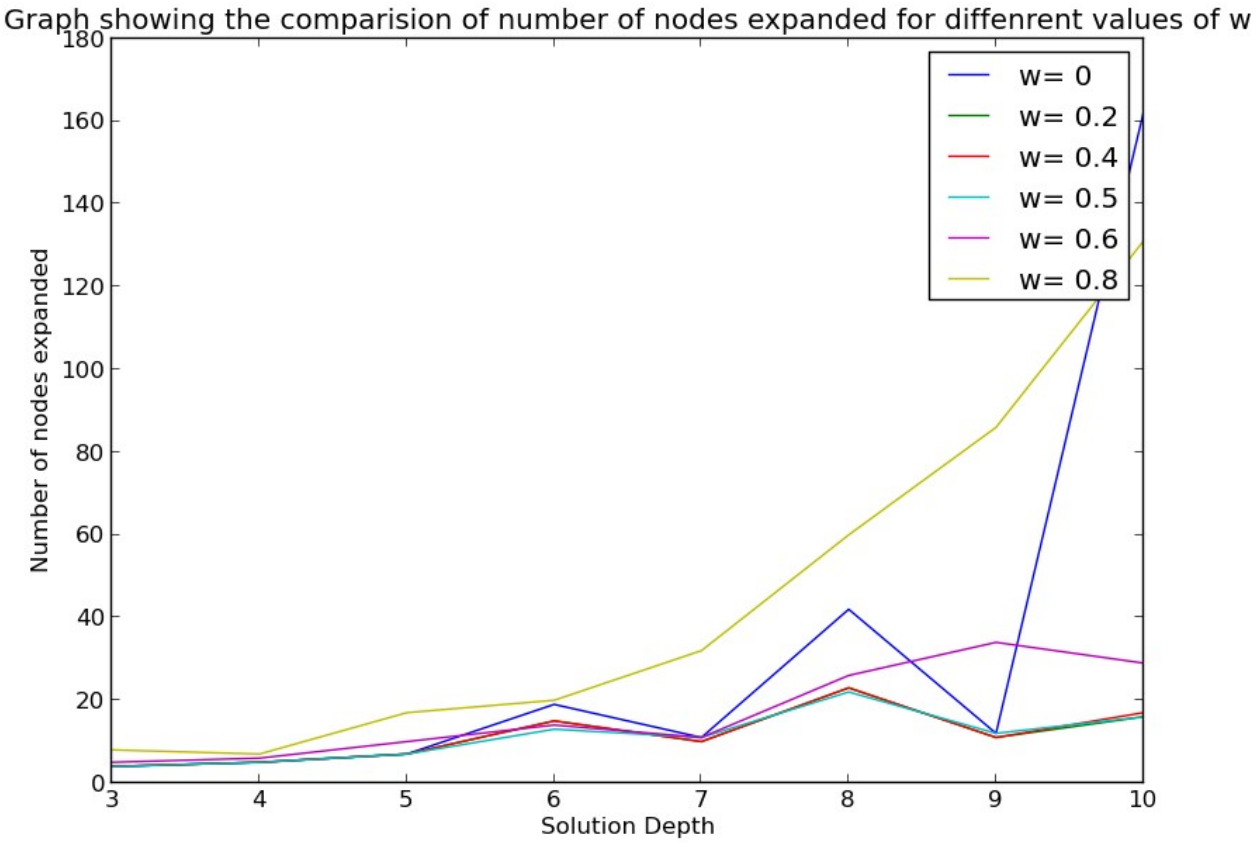
Solution Depth	Nodes Expanded	Nodes Generated	Maximum depth expanded	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	96
4	5	9	4	648	52
5	7	14	5	1008	80
6	14	25	6	1800	163
7	10	19	7	1368	116
8	18	32	8	2304	197
9	12	23	9	1656	141
10	16	31	10	2232	194

The comparision between table 4 and table 6 shows that the number of nodes expanded is lower when heuristics is manhatten distance . This is because manhatten distance is a tighter heuristics for the problem.

Table 7
Astar with Evaluation Fuction w=0

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	148
4	5	9	4	648	55
5	7	14	5	1008	82
6	19	34	6	2448	223
7	11	21	7	1512	131
8	42	70	14	5040	470

9	12	24	9	1728	146
10	162	270	46	19440	2004



The above figure shows the impact of using different values of w in the evaluation function of the Astar Algorithm

Table 8
Astar with Evaluation Function w=0.2

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in Bytes)	Time in microseconds
3	4	10	3	720	148
4	5	9	4	648	94

5	7	14	5	1008	84
6	15	26	6	1872	174
7	10	19	7	1368	119
8	23	40	8	2880	261
9	11	22	9	1584	131
10	16	32	12	2304	205

Table 9
Astar with Evaluation Function w=0.4

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	153
4	5	9	4	648	54
5	7	14	5	1008	82
6	15	26	6	1872	172
7	10	19	7	1368	118
8	23	41	8	2952	266
9	11	22	9	1584	132
10	17	32	10	2304	203

Table 10
Astar with Evaluation Function w=0.5

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in bytes)	Time in microseconds
3	4	10	3	720	190
4	5	9	4	648	56
5	7	14	5	1008	83
6	13	26	6	1872	165
7	11	19	7	1368	146
8	22	39	8	2808	300
9	12	23	9	1656	139
10	16	30	10	2160	253

Table 11
Astar with Evaluation Function w=0.6

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth	Memory Used (in bytes)	Time in microseconds

			Expanded		
3	5	12	3	864	162
4	6	11	4	792	61
5	10	20	5	1440	162
6	14	26	6	1872	191
7	11	19	7	1368	159
8	26	46	8	3312	334
9	34	61	9	4392	439
10	29	49	10	3528	356

Table 12
Astar with Evaluation Function w=0.8

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in bytes)	Time in microseconds
3	8	17	3	1224	195
4	7	14	4	1008	83
5	17	31	5	2232	199
6	20	34	6	2448	227
7	32	53	7	3816	351
8	60	100	8	7200	681
9	86	148	9	10656	1028
10	131	221	10	15912	1608

Table 13
Astar with Evaluation Function w=1.0

Solution Depth	Nodes Expanded	Nodes Generated	Maximum Depth Expanded	Memory Used (in bytes)	Time in microseconds
3	12	23	3	1656	197
4	17	32	4	2304	151
5	50	84	5	6048	412
6	100	162	6	11664	866
7	129	223	7	16056	1163
8	267	418	8	30096	2199
9	352	582	9	41904	1432
10	523	866	10	62352	2146

Observations:

1) As the solution depth increases, the number of nodes expanded and generated increase.
2) Uninformed search algorithms like BFS and DFS expand larger number of nodes in comparision to informed search algorithms
3) Manhatten distance is a tighter heuristics than the misplaced tiles heuristics for this problem
4) Evaluation functions performs optimally when w is around 0.5 in the evaluation function.
5) IDAstar usually uses lesser memory than Astar but takes more time.
6)