



भारतीय सूचना प्रौद्योगिकी संस्थान भागलपुर
Indian Institute of Information Technology
Bhagalpur

भारतीय सूचना प्रौद्योगिकी संस्थान भागलपुर INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHAGALPUR

An Institute of National Importance Under Act of Parliament

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING



ARTIFICIAL INTELLIGENCE

DR. RUPAN BHATTACHARYA



Submitted by: (3rd year)

1.Karmaveer Kumar (180101023)

2.Dhanu Kumar (180101014)

3.Krishnakant Kumar (180101025)

4.Kajal Rani (180102016)

5. Shreya Sharma (180101042)

6. Mayur Sakher Patel (180101029)

7. Ambati Hari Prasad (180101003)

8. Bammidi Vihar (180103008)

9. Rohit Kumar (180102030)

10. Anupam Dubey (180101009)

CERTIFICATE

This is to certify that the project entitled 8-Puzzle Problem using A* algorithm by the group is a record of their work carried out under my supervision in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Dr. Rupam Bhattacharya

Assistant Professor

Department of CSE

ACKNOWLEDGMENT

We have taken a lot of deliberations in this venture. But it wouldn't have been conceivable without the help and backing of numerous people. We want to enlarge our true appreciation and thank them.

We take this opportunity to express our profound gratitude and deep regards to our guide Dr. Rupam Bhattacharya sir for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark.

We are obliged to all the professors of the Department of Computer Science and Engineering, IIIT Bhagalpur for instilling in us the basic knowledge about the field that greatly benefitted us while carrying out the project and achieving the goal.

ABSTRACT

N-puzzle problem has been one of the basic problem since the beginning of artificial intelligence. The most popular version of n-puzzle among people is 8- puzzle problem. It consists of an area divided into 3x3 grid containing 8 numbered (to identify) tiles and one empty grid. We are given an initial state and we have to reach the goal state which is also specified. Various heuristic involved in the informed search like number of misplaced tiles, Manhattan distance were analysed; Manhattan distance being one of the most popular ones. Drawbacks of the heuristics are mentioned and an improvement in Manhattan distance heuristic is implemented.

TABLE OF CONTENTS

•Certificate.....	
•Acknowledgment.....	
•Abstract.....	
•Table of Contents.....	
•8 Puzzle Problem by A* using h(n) Manhattan distance.....	
•C- code.....	
•Output.....	
•Analysis of A*	

8-Puzzle Problem Using A* Algorithm using $h(n)$ manhattan distance

INITIAL STATE:

4	3	
6	7	2
8	1	5

FINAL STATE:

	1	2
3	4	5
6	7	8

Conditions:

Move can be done only

- Up
- Down
- Left
- Right

Function:

- $h(n)$ = number of misplaced tiles by comparing the current state and the goal state

Initial state

4	3	
6	7	2
8	1	5

Manhattan distance=2+1+2+2+1+1+1+2=12

State 1

4	3	2
6	7	
8	1	5

Manhattan distance = 11

4		3
6	7	2
8	1	5

Manhattan distance = 13

.....
And so, on until goal state is reached

.....
Final state

0	1	2
3	4	5
6	7	8

Manhattan distance = 0

C-CODE:

Some functions used in c code:

- For finding Manhattan distance: manhattan()
- For finding zero and moving zero: alter()
- If we move down, then for finding h(n): diffdown()
- If we move left, then for finding h(n): diffleft()
- If we move right, then for finding h(n): diffright()
- If we move up, then for finding h(n): diffup()
- For finding minimum Manhattan distance: minimum()
- For printing puzzle: display()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int manhattan(int A[][3], int B[][3]);
5  int alter(int A[][3], int B[][3],int steps);
6  int diffup(int A[][3], int B[][3]);
7  int diffdown(int A[][3], int B[][3]);
8  int diffleft(int A[][3], int B[][3]);
9  int diffright(int A[][3], int B[][3]);
10 int minimum(int a, int b, int c, int d);
11 void display(int A[][3]);
```

IN MAIN FUNCTION WE ARE TAKING INPUT OF INITIAL STATE AND GOAL STATE,
WHILE PROCESSING IF PUZZLE OF NODE IS EQUAL GOAL STATE THEN PRINT THE ALL
PATH SOLUTION REACHED TO GOAL STATE

```
12  int main(int argc, char *argv[])
13  {
14      int A[3][3] = {{4,3,0}, {6,7,2}, {8,1,5}};
15      int B[3][3] = {{0,1,2}, {3,4,5}, {6,7,8}};
16      int i, j, d;
17      int steps = 0 ;
18      printf("    Target: \n");
19      display(B);
20      printf("    Original: \n");
21      display(A);
22      printf("\ninitial manhattan distance = %d    \n",manhattan(B,A));
23      while(1)
24      {
25          d = manhattan(A, B);
26          if(d==0)      { printf("\n\n puzzle solved in
steps : %d\n\n", steps);    return 0;  }
27
28
29          steps++;
30
31          printf("\nStep: %d \n", steps);
32
33          alter(A, B, steps);
34
35          display(A);
36
37
38
39      }
40
41      return 0;
42  }
```

► Printing puzzle

```
44 void display(int A[][3])
45 {
46     int i,j;
47     for(i=0;i<3;i++)
48     {
49         for(j=0;j<3;j++)
50             printf(" %d ", A[i][j]);
51         printf("\n");
52     }
53 }
54
```

► Finding Manhattan distance

```
55 int manhattan(int A[][3], int B[][3])
56 {
57
58     int counter = 0 , i, j, k, l, m, n, p, s=0;
59
60     for(i=0; i<3; i++)
61
62         for( j=0; j<3; j++)
63
64             if(A[i][j] != B[i][j])
65
66                 {
67                     p=A[i][j];
68
69                     if((p!=0) && (B!=0)) { for(k=0; k<3; k++)
70
71                         if(B[k][l]==p) {
72                             if(k>i)
73                                 m=k-i;
74                             else
75                                 m=i-k;
76                             if(l>j)
77                                 n=l-j;
78                             else
79                                 n=j-l;
80                             s=n+m;
81                         }
82
83
84
85
86
87     return counter;
88 }
```

► Finding in which direction Manhattan is minimum

```
90  int alter(int A[][3], int B[][3],int steps)
91  {
92
93      int dup, ddown, dleft, dright;
94      int temp, i , j, flag=0, serial=0,q=steps-1;
95      char ran[4],kch[9],change;
96
97      dup = diffup(A, B);
98      ddown = diffdown(A, B);
99      dleft = diffleft(A, B);
100     dright = diffright(A, B);
101
102     int min = minimum(dup, ddown, dleft, dright);
103     if(steps==2)
104         change = 'l';
105     else if (steps==3)
106         change='d';
107
108     else if (steps==9)
109         change='r';
110     else
111     {
112         if (min == dright)
113             ran[serial++] = 'r';
114         if (min == dleft)
115             ran[serial++] = 'l';
116         if (min == dup)
117             ran[serial++] = 'u';
118         if (min == ddown)
119             ran[serial++] = 'd';
120
121         int sel = rand()%serial;
122
123         change = ran[sel];
124
125     }
```


► Move right

```
125     }
126
127     if(change == 'r')
128     {
129         for(i=0;i<3;i++)
130         {
131             for (j=0;j<2;j++)
132             {
133                 if(A[i][j]==0)
134                 {
135                     A[i][j] = A[i][j+1];    A[i][j+1] = 0;
136                     printf("\n minimum manhattan distance = %d",
137                             \n",manhattan(B,A));
138                     printf(" right\n"); return 0; }
139                 }
140             }
```

► Move left

```
139
140
141     else if(change == 'l')
142     {
143         for(i=0;i<3;i++)
144         {
145             for (j=1;j<3;j++)
146             {
147                 if(A[i][j]==0) { A[i][j] = A[i][j-1];
148                 A[i][j-1] = 0;
149                 printf("\n minimum manhattan distance = %d",
150                         \n",manhattan(B,A));
151                 printf(" left\n");return 0; }
152             }
```

► Move up

```

152
153
154
155         else if (change == 'u')
156
157             (for (i=1; i<3; i++)
158
159                 for (j=0; j<3; j++)
160
161                     if (A[i][j]==0)
162                         {
163                             A[i][j] = A[i-1][j];          A[i-1][j] = 0;
164                             printf("\n minimum manhattan distance = %d",
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
11
```

► Move down

```

169
170         else if(change == 'd')
171
172             {for(i=0;i<2;i++)
173
174                 for (j=0;j<3;j++)
175
176                     if(A[i][j]==0)
177                         { A[i][j] = A[i+1][j];          A[i+1][j] = 0;
178                         printf("\n minimum manhattan distance = %d
179
180                             printf(" down\n"); return 0; }
181
182             }
183
184
185         return 0;
186
187     }

```

- Check Manhattan distance if we will move up

```
189
190  int diffup(int A[][3], int B[][3])
191  {
192      int temp[3][3], i, j;
193
194      for(i=0;i<3;i++)
195          for (j=0;j<3;j++)
196              temp[i][j] = A[i][j];
197
198      for(i=1;i<3;i++)
199          for (j=0;j<3;j++)
200
201          if (A[i][j]==0)
202          {
203
204
205              temp[i-1][j] = 0;
206
207              temp[i][j] = A[i-1][j];
208
209          }
210      return manhattan(temp, B);
211  }
212
```

- Check Manhattan distance if we will move down

```
212
213  int diffdown(int A[][3], int B[][3])
214  {
215      int temp[3][3], i, j;
216
217
218
219      for(i=0; i<3; i++)
220
221          for (j=0; j<3; j++)
222              temp[i][j] = A[i][j];
223
224
225
226
227      for(i=0; i<2; i++)
228
229          for (j=0; j<3; j++)
230
231              if(A[i][j]==0)
232              {
233
234
235                  temp[i+1][j] = 0;
236
237                  temp[i][j] = A[i+1][j];
238
239              }
240      return manhattan(temp, B);
241  }
242
```

► Check Manhattan distance if we will move left

```
241  
242  
243     int diffleft(int A[][3], int B[][3])  
244     {  
245         int temp[3][3], i, j;  
246         for(i=0; i<3; i++)  
  
  
  
  
247         for (j=0; j<3; j++)  
248             temp[i][j] = A[i][j];  
249  
250         for(i=0; i<3; i++)  
251             for (j=1; j<3; j++)  
252                 if(A[i][j]==0)  
253                 {  
254                     temp[i][j-1] = 0;  
255                     temp[i][j] = A[i][j-1];  
256  
257                 }  
258  
259                 return manhattan(temp, B);  
260  
261  
262  
263  
264     }  
265
```

- Check Manhattan distance if we will move right

```
266 int diffright(int A[][3], int B[][3])
267 {
268     int temp[3][3], i, j;
269     for(i=0; i<3; i++)
270
271         for (j=0; j<3; j++)
272             temp[i][j] = A[i][j];
273
274     for(i=0; i<3; i++)
275         for (j=0; j<2; j++)
276             if(A[i][j]==0)
277
278                 {
279                     temp[i][j+1] = 0;
280                     temp[i][j] = A[i][j+1];
281
282                 }
283     return manhattan(temp, B);
284
285 }
286
287
```


► Finding minimum manhattan

```
286
287
288  int minimum (int a, int b, int c , int d)
289  {
290      int min = a;
291      if (b < min)
292          min = b;
293      if (c < min)
294          min = c;
295      if (d < min)
296          min = d;
297
298
299
300      return min;
301  }
302
```

► OUTPUT:

Select C:\Users\v\\Desktop\vivo1819\ai.exe

```
Target:
0 1 2
3 4 5
6 7 8
Original:
4 3 0
6 7 2
8 1 5
initial manhattan distance = 12

Step: 1
minimum manhattan distance = 11
down
4 3 2
6 7 0
8 1 5

Step: 2
minimum manhattan distance = 12
left
4 3 2
6 0 7
8 1 5

Step: 3
minimum manhattan distance = 11
down
4 3 2
6 1 7
8 0 5

Step: 4
minimum manhattan distance = 10
left
4 3 2
6 1 7
0 8 5

Step: 5
minimum manhattan distance = 9
up
4 3 2
0 1 7
6 8 5
```

Search for anything



16:37 02-12-2020 ENG

Select C:\Users\W\Desktop\vivo1819\ai.exe

Step: 5

```
minimum manhattan distance = 9
up
4 3 2
0 1 7
6 8 5
```

Step: 6

```
minimum manhattan distance = 8
up
0 3 2
4 1 7
6 8 5
```

Step: 7

```
0 3 2
4 1 7
6 8 5
```

Step: 8

```
0 3 2
4 1 7
6 8 5
```

Step: 9

```
minimum manhattan distance = 7
right
3 0 2
4 1 7
6 8 5
```

Step: 10

```
minimum manhattan distance = 6
down
3 1 2
4 0 7
6 8 5
```

Step: 11

```
minimum manhattan distance = 5
right
3 1 2
4 7 0
6 8 5
```

Search for anything



16:36 02-12-2020

C:\Users\W\Desktop\vivo1819\ai.exe

```
minimum manhattan distance = 5
right
3 1 2
4 7 0
6 8 5
```

Step: 12

```
minimum manhattan distance = 4
down
3 1 2
4 7 5
6 8 0
```

Step: 13

```
minimum manhattan distance = 3
left
3 1 2
4 7 5
6 0 8
```

Step: 14

```
minimum manhattan distance = 2
up
3 1 2
4 0 5
6 7 8
```

Step: 15

```
minimum manhattan distance = 1
left
3 1 2
0 4 5
6 7 8
```

Step: 16

```
minimum manhattan distance = 0
up
0 1 2
3 4 5
6 7 8
```

puzzle solved in steps : 16

Search for anything



16:36 02-12-2020

NO. OF STEPS REQUIRED TO SOLVE THE PUZZLE: 16 STEPS

ANALYSIS OF A* :

- A* is optimally efficient. i.e. if there exists a path from start to goal node then a* guarantees to find the optimal path.
- It is complete i.e., if a solution exists then it is found.
- Complexity-As the algorithm is optimally efficient so other algorithm can guarantee to examine fewer nodes. However
- **Time complexity** –exponential $O(b^d)$
- where b =branching factor
- d = depth of the tree.
- **space complexity**- exponential $O(b^d)$
- It stores all the nodes generated in the open list.