Best-First Search and Algorithm A* Revisited (More Informed)

Note Title

Recall: "Algorithm A"
is actually a class of heuristics for Best-First-Search.
Best-First Search is the GraphSearch algorithm, using a heuristic, $f(n)$
to order the nodes on the OPEN list.
Using Algorithm A means using a heuristic of the form

$$f(n) = depth(n) + \underline{\underline{h(n)}}$$

estimate of distance from n to goal –

$f(n)$ is an estimate of the total distance (i.e, "path length") from start to goal if you go through node n.

Algorithm A* is Algorithm A,

using a heuristic $h(n)$ that is guaranteed not to over-estimate the distance from n to goal.

Use of Algorithm A* is guaranteed to find a shortest-path solution.

More informed heuristics for Algorithm A*:

if $h_1(n)$ and $h_2(n)$ are both A* heuristics ("admissible" heuristics)

then if $h_2(n) \geq h_1(n)$ for all n

$h_2(n)$ is more informed than $h_1(n)$

and $h_2(n)$ will result in no more total node expansions than $h_1(n)$.

New heuristic for 8-puzzle: $h_2(\text{node})$ = Total # of spaces removed from the goal for each tile out of place

initial state

2	8	3
1	6	4
7		5

1	2	3	4	5	6	7	8
1	1	OK	OK	OK	1	OK	2

We can see $h_2(\text{state}) \geq h_1(\text{state})$

$h_2(\text{state}) \leq \underbrace{h^*(\text{state})}_{\text{actual \# of moves needed.}}$

$h_2(\text{state}) = 1+1+1+2 = 5$

$h_1(\text{state}) = 4$ (# tiles out of place)

So let's re-evaluate with $h_2(\text{node})$ also.

depth = 0
$h = 4$
$f = 4$

2	8	3
1	6	4
7		5

A

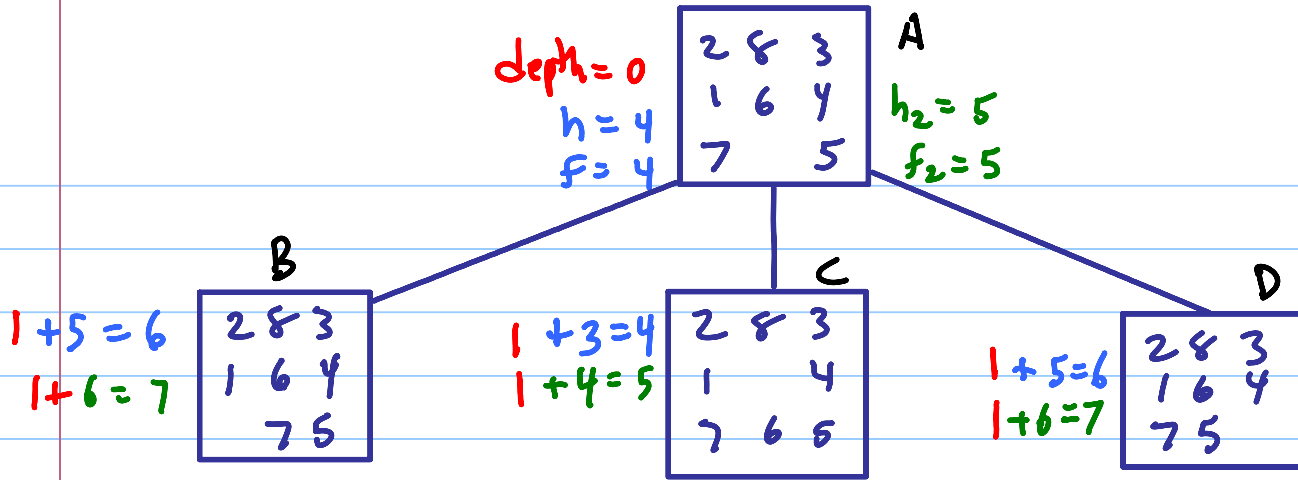
$h_2 = 5$
$f_2 = 5$

OPEN	A
CLOSED	

$h_2(n)$ = total distance between tiles out of place and the place they belong -

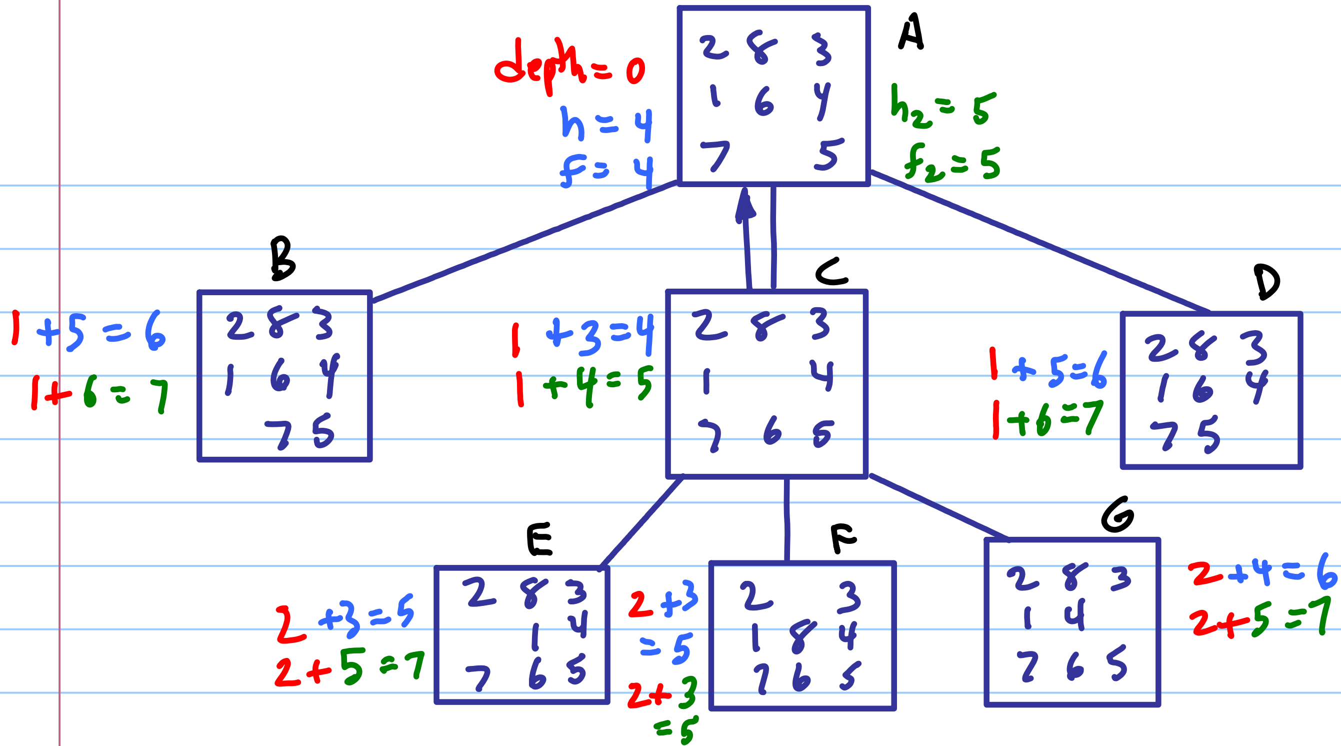
1	2	3	4	5	6	7	8	
↓	↓	-	-	-	↓	-	↓	
1	1				1		2	⇒ 5

This is admissible because each tile has to move at least that many times to reach goal state



OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A



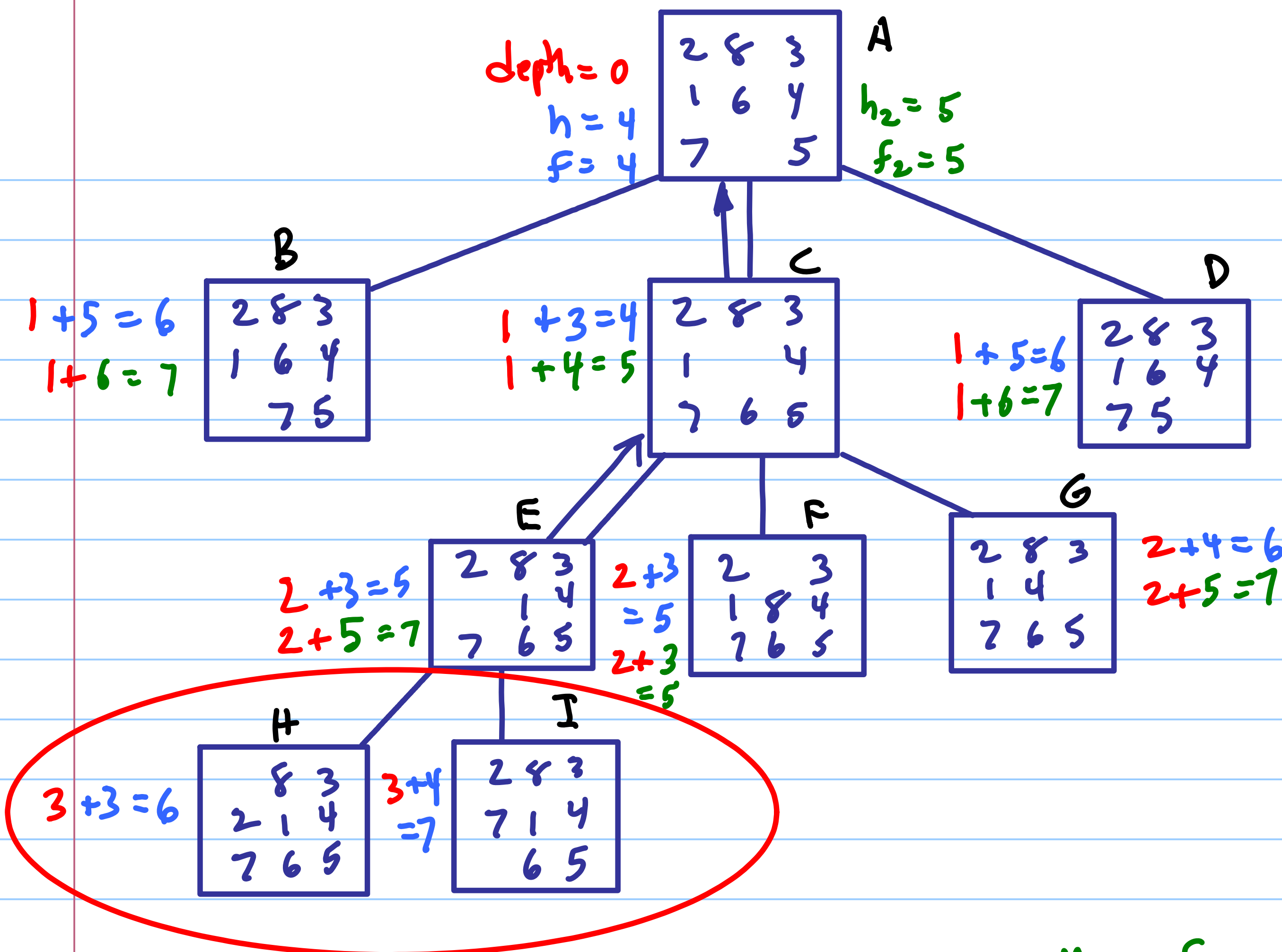
OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

OPEN	E F G B D E
CLOSED	A C

since E has a higher heuristic value with h_2 than it did with h_1, it is placed farther back in the queue, rather than the beginning.

More informed heuristic moves E to end.



These nodes are not expanded when using h_2(node)

OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

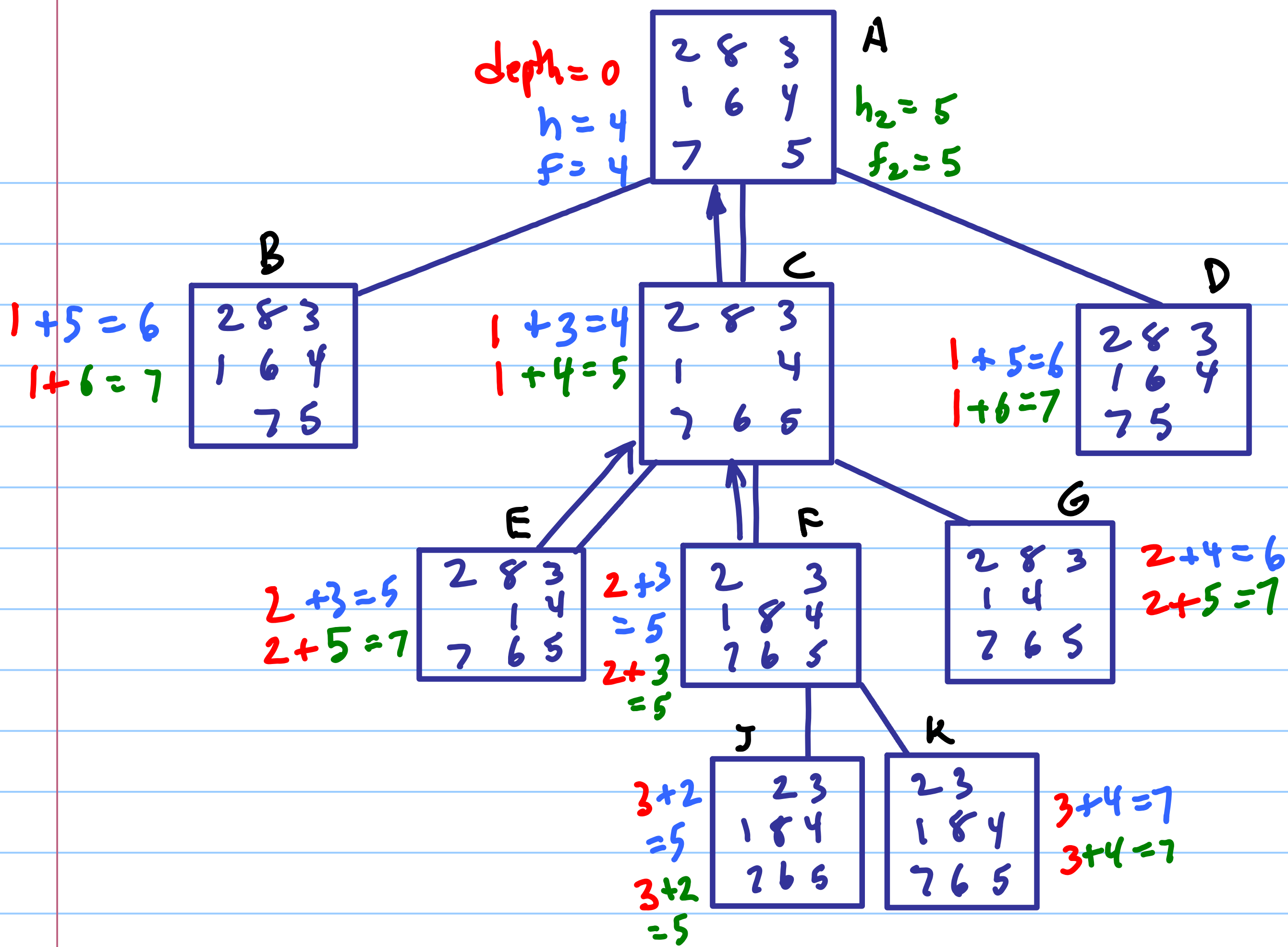
OPEN	E F G B D E
CLOSED	A C

OPEN	F G B D E H I
CLOSED	A C

H and I are not placed in the queue, since E hasn't been expanded.

More informed heuristic moves E to end.

HI nodes are never generated.



OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

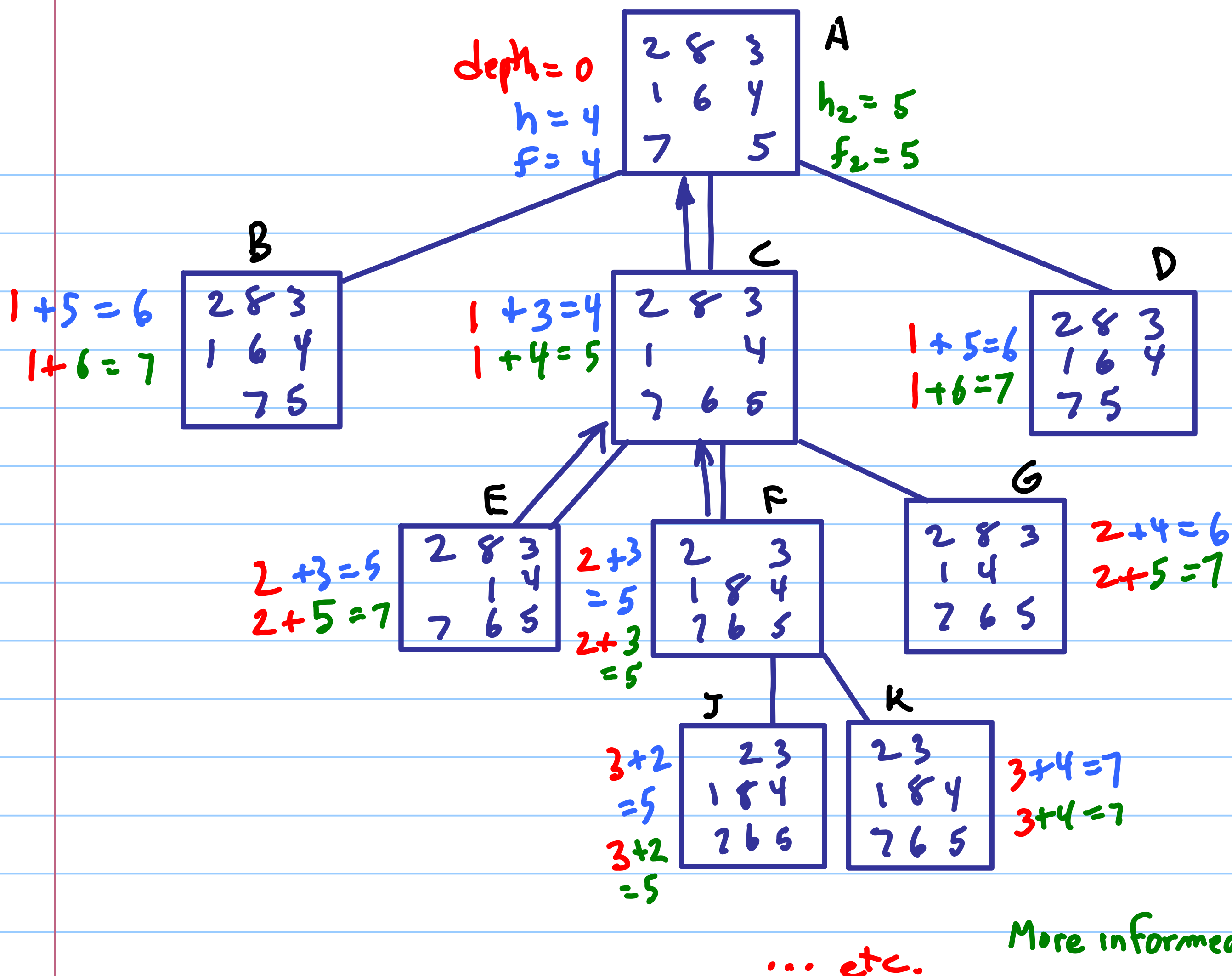
OPEN	E F G B D E
CLOSED	A C

OPEN	F G B D E H I
CLOSED	A C

OPEN	J G B D E H I K
CLOSED	A C F

More informed heuristic moves E to end.

HI nodes are never generated.



OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

OPEN	E F G B D E
CLOSED	A C

OPEN	F G B D E H I
CLOSED	A C

OPEN	J G B D E H I K
CLOSED	A C F

...

continue to goal as before.

More informed heuristic moves E to end.

HI nodes are never generated.

$h_2(n)$ is more informed than $h_1(n)$.

Since $h_2(n)$ and $h_1(n)$ are both admissible heuristics, each guarantees it will find a shortest-path solution to goal when used with Algorithm A and Best-First Search.

Since $h_2(n)$ is more informed than $h_1(n)$, $h_2(n)$ will use no more total node expansions than $h_1(n)$ to find a shortest-path solution.

$h_2(n)$ is more informed than $h_1(n)$.

Since $h_2(n)$ and $h_1(n)$ are both admissible heuristics, each guarantees it will find a shortest-path solution to goal when used with Algorithm A and Best-First Search.

Since $h_2(n)$ is more informed than $h_1(n)$, $h_2(n)$ will use no more total node expansions than $h_1(n)$ to find a shortest-path solution.

- Note: Can I trick Algorithm A* by using $h(n) = 0$ for all n?

  [clearly, $h(n)$ never exceeds actual distance from n to goal — So I will find shortest path to goal, guaranteed!

$h_2(n)$ is more informed than $h_1(n)$.

Since $h_2(n)$ and $h_1(n)$ are both admissible heuristics, each guarantees it will find a shortest-path solution to goal when used with Algorithm A and Best-First Search.

Since $h_2(n)$ is more informed than $h_1(n)$, $h_2(n)$ will use no more total node expansions than $h_1(n)$ to find a shortest-path solution.

- <u>Note</u>: Can I trick Algorithm A* by using $h(n) = 0$ for all n?

  [clearly, $h(n)$ never exceeds actual distance from n to goal—
  So I will find shortest path to goal, <u>guaranteed</u>!

  (so $f(n) = \text{depth}(n) + 0$)

  $= \text{depth}(n)$

This is breadth-first search!

(And of course, we will find shortest path to goal with breadth-first search)

Note that $h(n) = $ # tiles out of place is a more informed heuristic than $h(n) = 0$ —

So $h_2(n) \geq h_1(n) \geq h(n)$

breadth-first search

more informed search

even more informed search

Suppose $h_1(n) > h_2(n)$ sometimes

and $h_2(n) > h_1(n)$ sometimes

where $h_1(n)$, $h_2(n)$ are both admissible?

Suppose $h_1(n) > h_2(n)$ sometimes

and $h_2(n) > h_1(n)$ sometimes

where $h_1(n), h_2(n)$ are both admissible?

Let $h_3(n) = \max\{h_1(n), h_2(n)\}$

Note $h_3(n) \geq h_1(n)$

$h_3(n) \geq h_2(n)$

$h_3(n)$ is more informed than either heuristic

Even more informed?

We need to swap two adjacent tiles ~ can this be solved?

2	1	3
8		4
7	6	5

We need to swap two sets of adjacent tiles

2	1	3
8		4
7	5	6

$h_1(n) = 4$

$h_2(n) = 4$

$h_4(n) = 8$

$h(n) = 0$

$h_1(n) =$ # tiles out of place

$h_2(n) =$ # of spaces the out-of-place tiles must move

$h_2(n) \geq h_1(n) \geq h(n)$

let $h_4(n) = h_2(n) + 2 * ($# of pairs of adjacent out-of-place tiles$)$

probably admissible

definitely $h_4(n) \geq h_2(n)$

Sample puzzle with 2 pairs of tiles that need to be switched with their immediate neighbors:

Winning moves for the game: {2,1,3,8,0,4,7,5,6}

Move #1 : Blank Up
Move #2 : Blank Right
Move #3 : Blank Down
Move #4 : Blank Down
Move #5 : Blank Left
Move #6 : Blank Left
Move #7 : Blank Up
Move #8 : Blank Right
Move #9 : Blank Right
Move #10 : Blank Up
Move #11 : Blank Left
Move #12 : Blank Left
Move #13 : Blank Down
Move #14 : Blank Down
Move #15 : Blank Right
Move #16 : Blank Up

Solution from software provided at
www.8puzzle.com

SOLUTION
=====*INITIAL STATE*
2 1 3
8 0 4
7 5 6

=====*STEP 1*
2 0 3
8 1 4
7 5 6

=====*STEP 2*
2 3 0
8 1 4
7 5 6

=====*STEP 3*
2 3 4
8 1 0
7 5 6

=====*STEP 4*
2 3 4
8 1 6
7 5 0

=====*STEP 5*
2 3 4
8 1 6
7 0 5

=====*STEP 6*
2 3 4
8 1 6
0 7 5

=====*STEP 7*
2 3 4
0 1 6
8 7 5

=====*STEP 8*
2 3 4
1 0 6
8 7 5

=====*STEP 9*
2 3 4
1 6 0
8 7 5

=====*STEP 10*
2 3 0
1 6 4
8 7 5

=====*STEP 11*
2 0 3
1 6 4
8 7 5

=====*STEP 12*
0 2 3
1 6 4
8 7 5

=====*STEP 13*
1 2 3
0 6 4
8 7 5

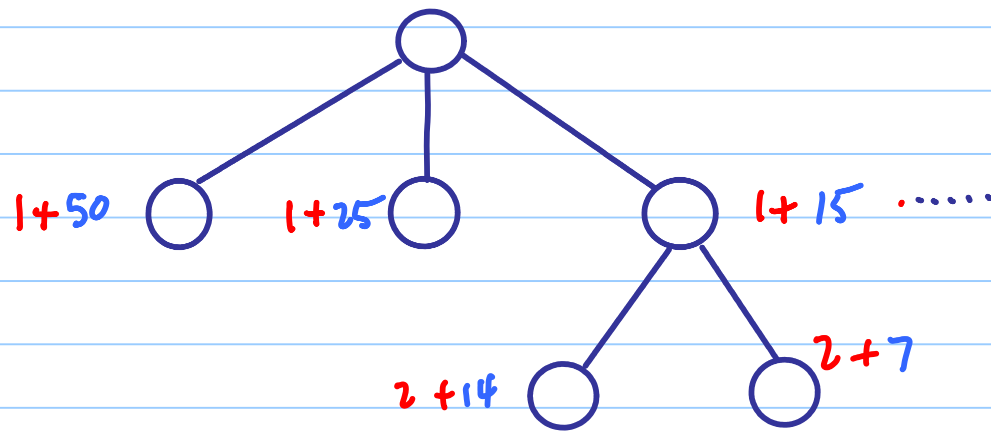
=====*STEP 14*
1 2 3
8 6 4
0 7 5

=====*STEP 15*
1 2 3
8 6 4
7 0 5

=====*STEP 16*
1 2 3
8 0 4
7 6 5

Algorithm A**

Example (Uniform Cost):



Since h() is admissible, this node is <u>at least</u> 15 from goal

this node is at least 7 from <u>goal</u>
[but we know it is farther than that because parent is at least 15 from goal]

Algorithm A**:

If Algorithm A* uses heuristic

$$f(node) = depth(node) + h(node)$$

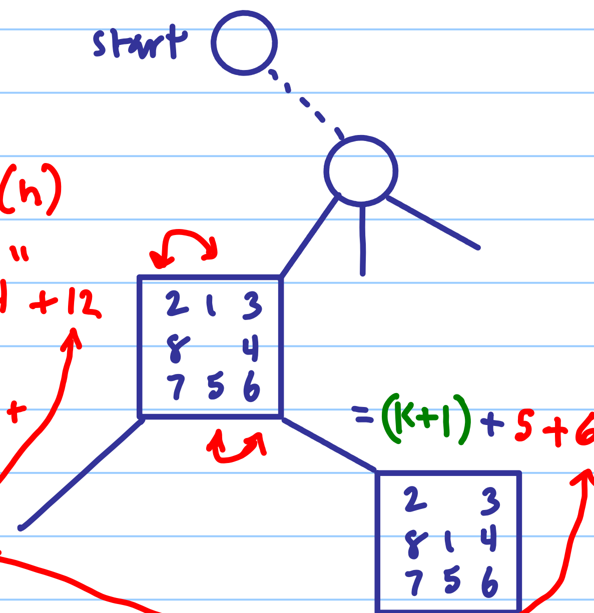
Then Algorithm A** uses the following heuristic:

$$\hat{f}(node)$$

where

$$\hat{f}(node) = \max_{\substack{n \in \text{path} \\ \text{from start} \\ \text{to node}}} \{ f(n) \}$$

for example, in 8-puzzle example, a heuristic that adds a suitably large number when two adjacent tiles need to be switched



$$f(n) = \underbrace{depth(n)}_{K} + h_4(n)$$

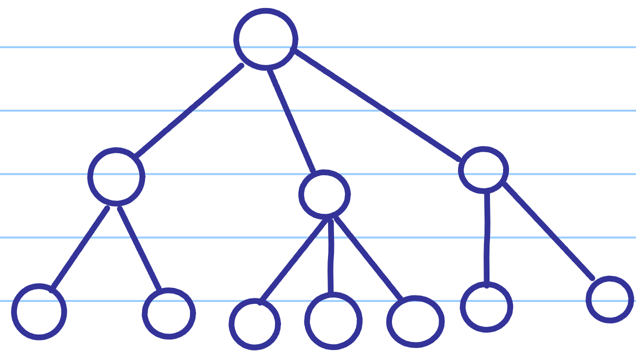
$$= K + 4 + 12$$

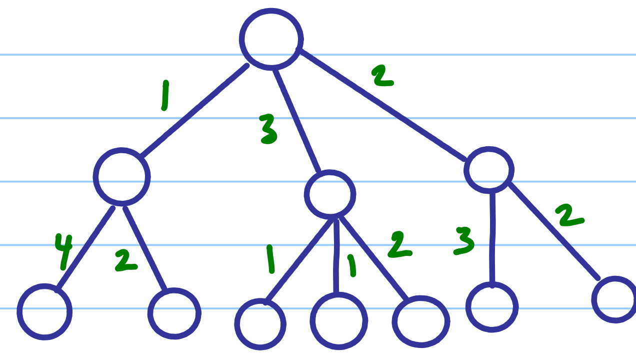
$h_2(n)$ + 3 * (# direct reversals)

$= (K+1) + 5 + 6$

[Estimate at level K is higher than estimate at level $K+1$]

Uniform vs. non-uniform costs:





Uniform:

every edge has the same value/weight (= 1)

non-uniform:

depth = sum of edge weights along path.

Total distance = sum of edge weights from start → goal

Graphsearch for non-uniform cost case

=

Shortest path algorithm for a graph

←

$h(n)$ = estimate of total distance to goal (sum of edge weights)

$f(n) = depth(n) + h(n)$

use $f(n)$ to order OPEN from least to most

$depth(s) = depth(parent(s)) + cost(edge(parent(s), s))$