Best-First Search and Algorithm A*

Note Title

Graphsearch:

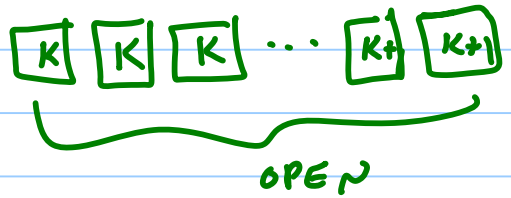
Insert (s', OPEN)

- insert at front: Depth-First Search
  - IDGA: heuristic = constant (just put at front, since all have same value)
- insert at rear: Breadth-First Search
  - heuristic = depth(node)
- insert in place where it probably belongs: Best-First Search

in general, don't know for sure — use a good heuristic

low → good
high → bad

OPEN ordered from low to high



IDEA ("Algorithm A"): [Let f(n) be an estimate of the total distance from start to goal if you go through node n.]

i.e., "path length"

Example:

start: Drexel

goal: Disney World

f(Alaska) > f(Kansas) > f(South Carolina)

depth(South Carolina) > depth(New York)

but

f(South Carolina) < f(New York)

Alg. A heuristic: $f(n) = depth(n) + h(n)$

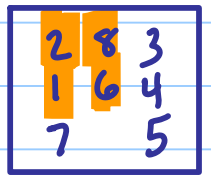
$h(n)$: estimate of distance from n to goal

- We compute f(n) when we put n on OPEN.
- We already have generated n — we know depth(n)

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

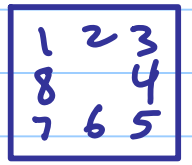
[Use of Algorithm A* is guaranteed to find a shortest-path solution !!!!!]

for 8-puzzle: heuristic $h(\text{node})$ = # tiles out of place (not including blank)



$h(\text{start}) = 4$ { 1, 2, 6, 8 out of position ; 3 4 5 7 in position }

goal:



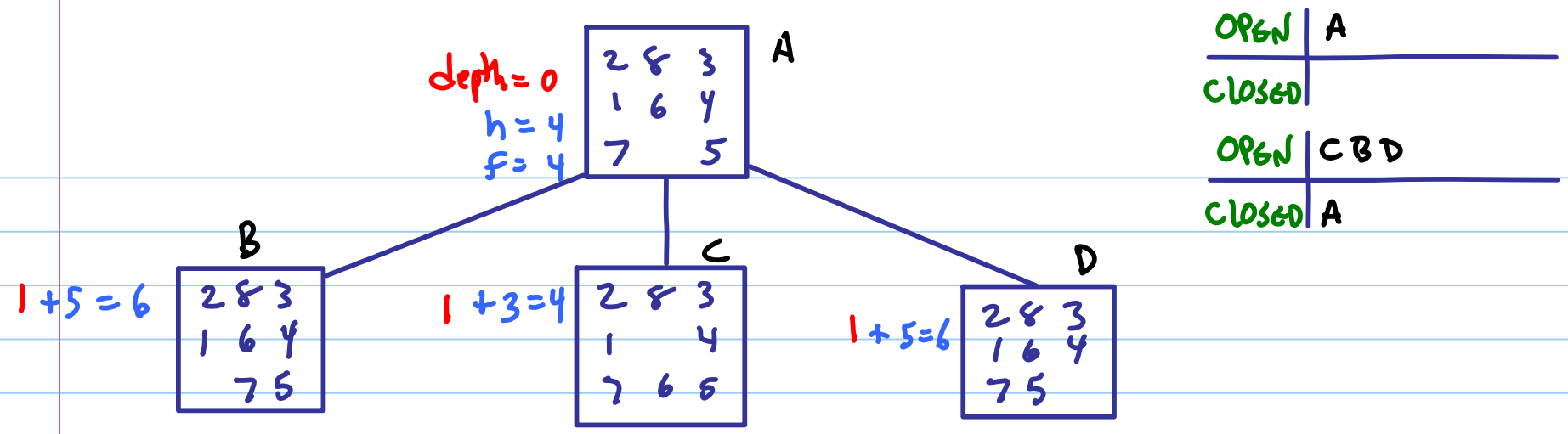
Note: it will take at least 4 moves to get all tiles in position (goal!)
because on a single move, at most 1 tile can move into position

depth = 0
h = 4
f = 4

2	8	3
1	6	4
7		5

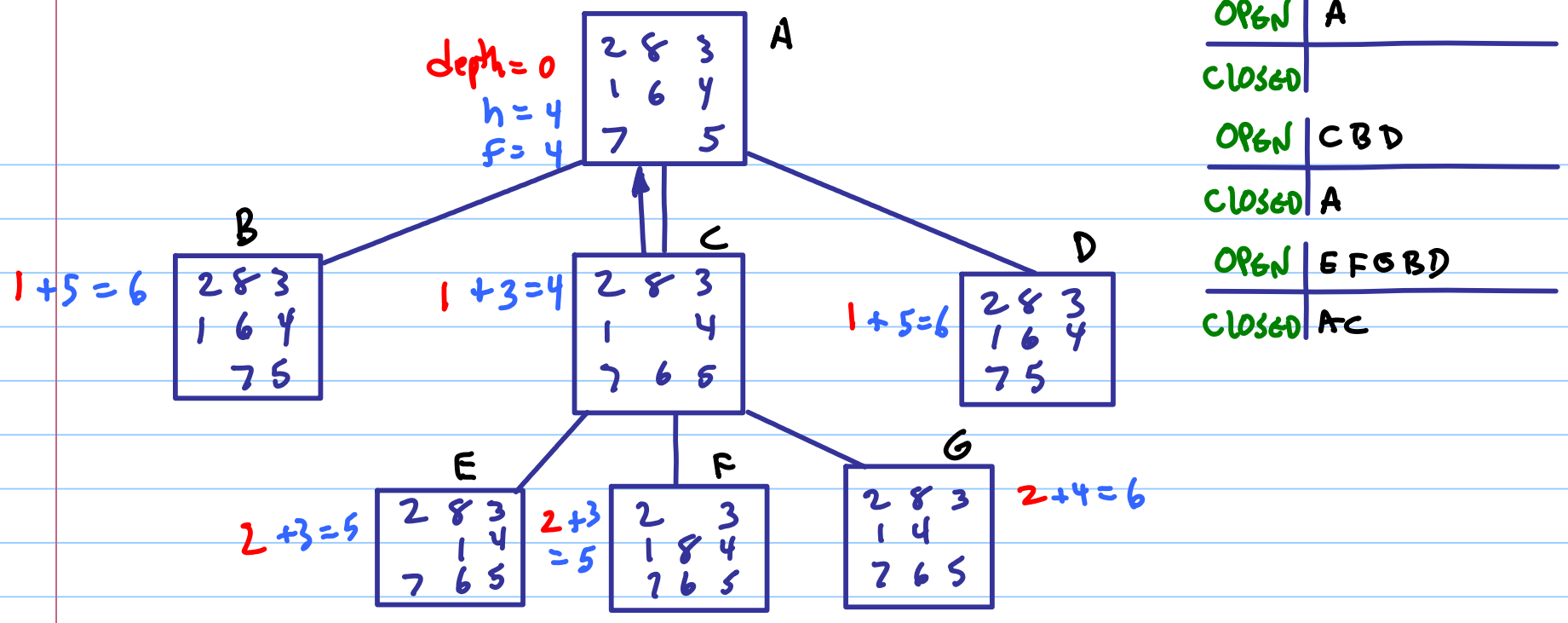
A

OPEN	A
CLOSED	



OPEN	A
CLOSED	

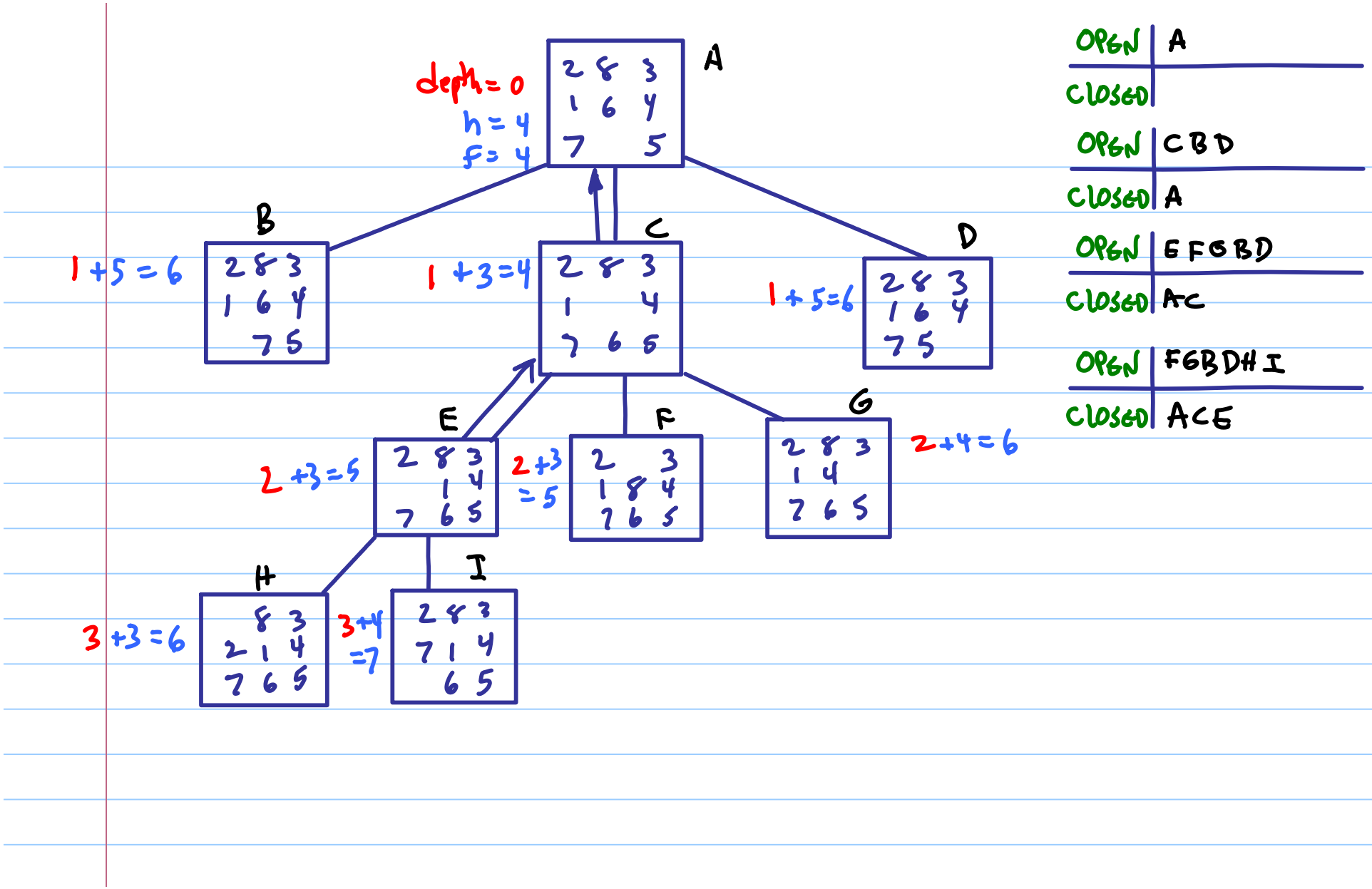
OPEN	C B D
CLOSED	A



OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

OPEN	E F G B D
CLOSED	A C

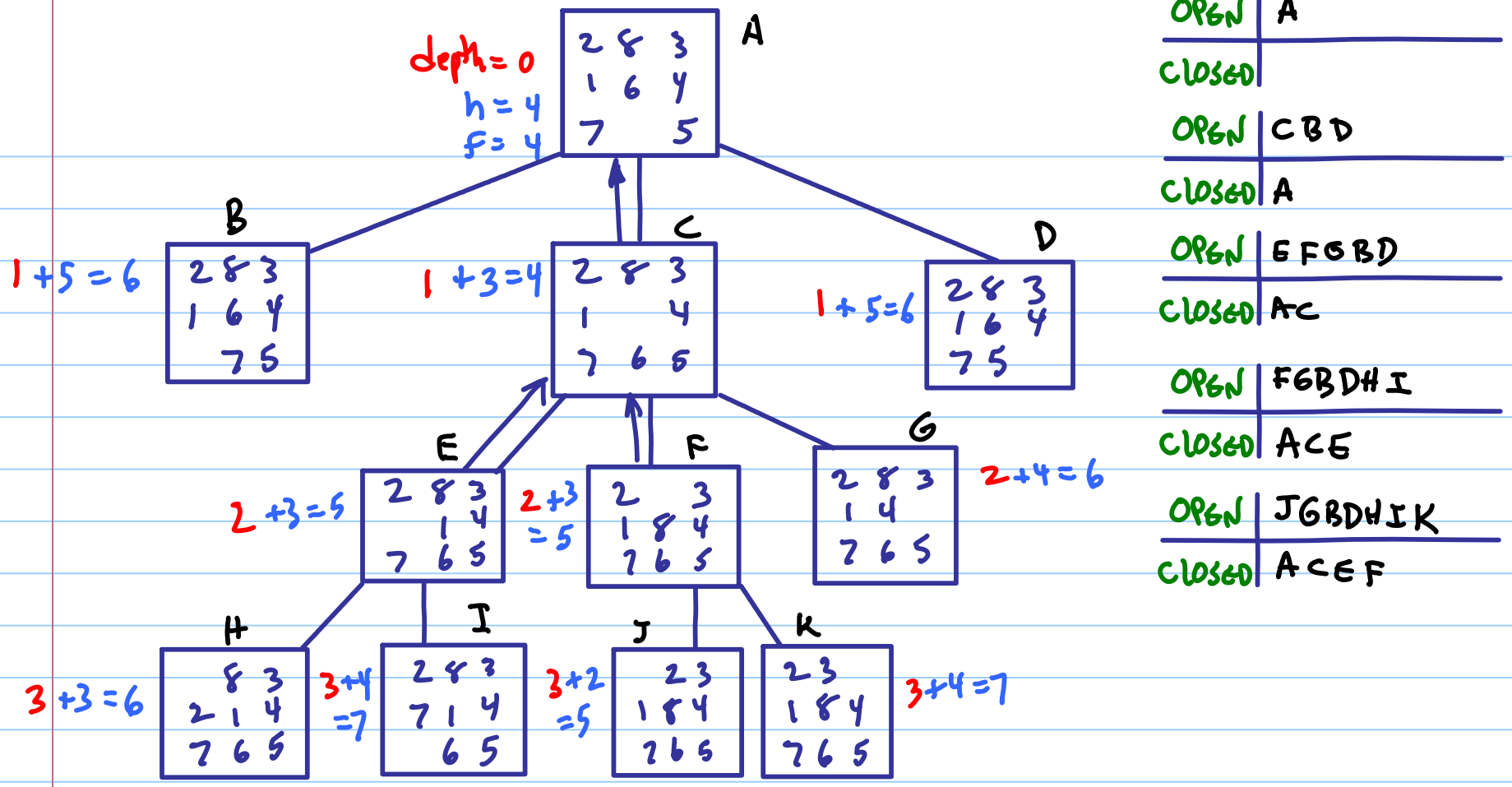


OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

OPEN	E F G B D
CLOSED	A C

OPEN	F G B D H I
CLOSED	A C E



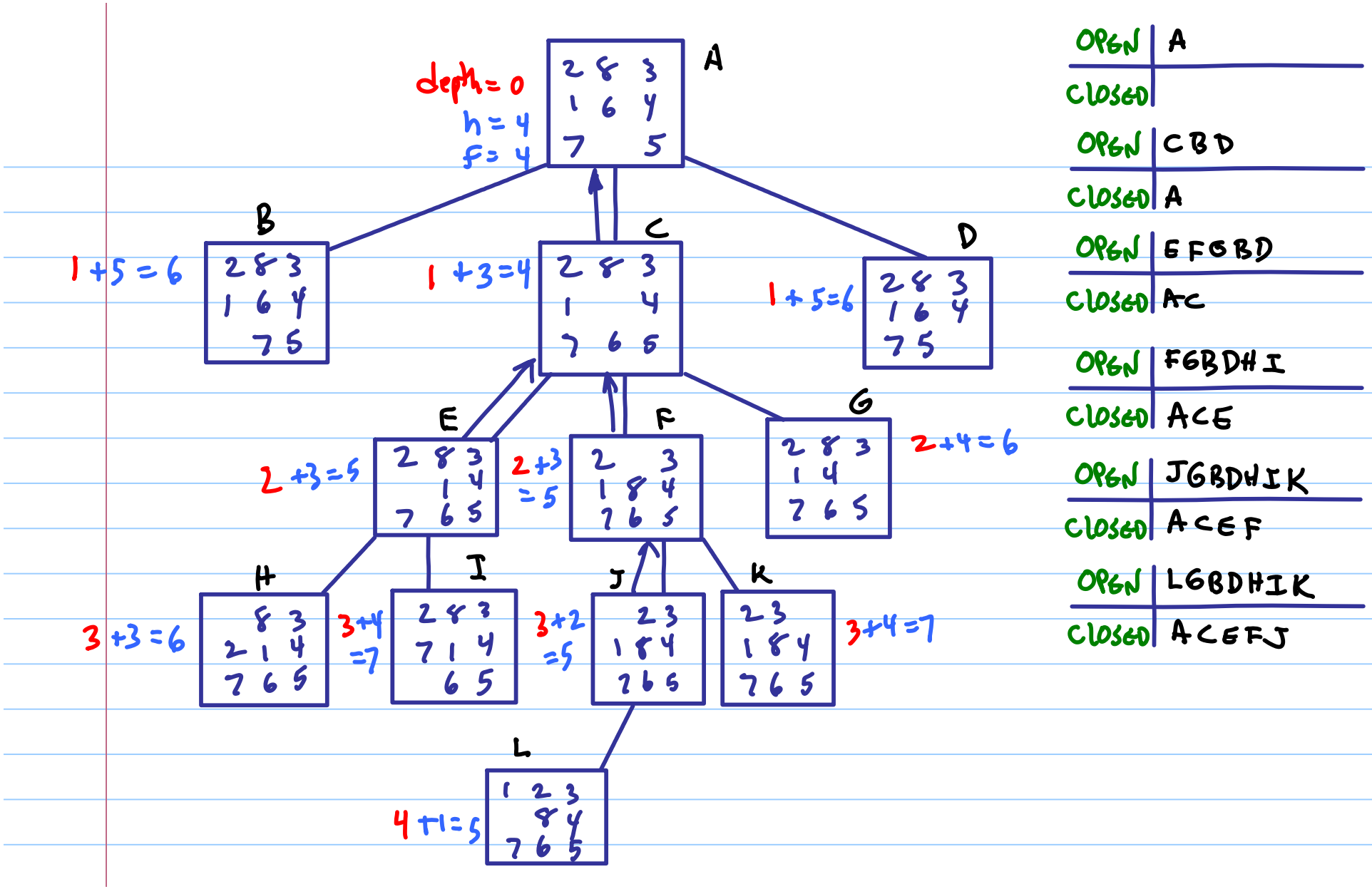
OPEN	A
CLOSED	

OPEN	C B D
CLOSED	A

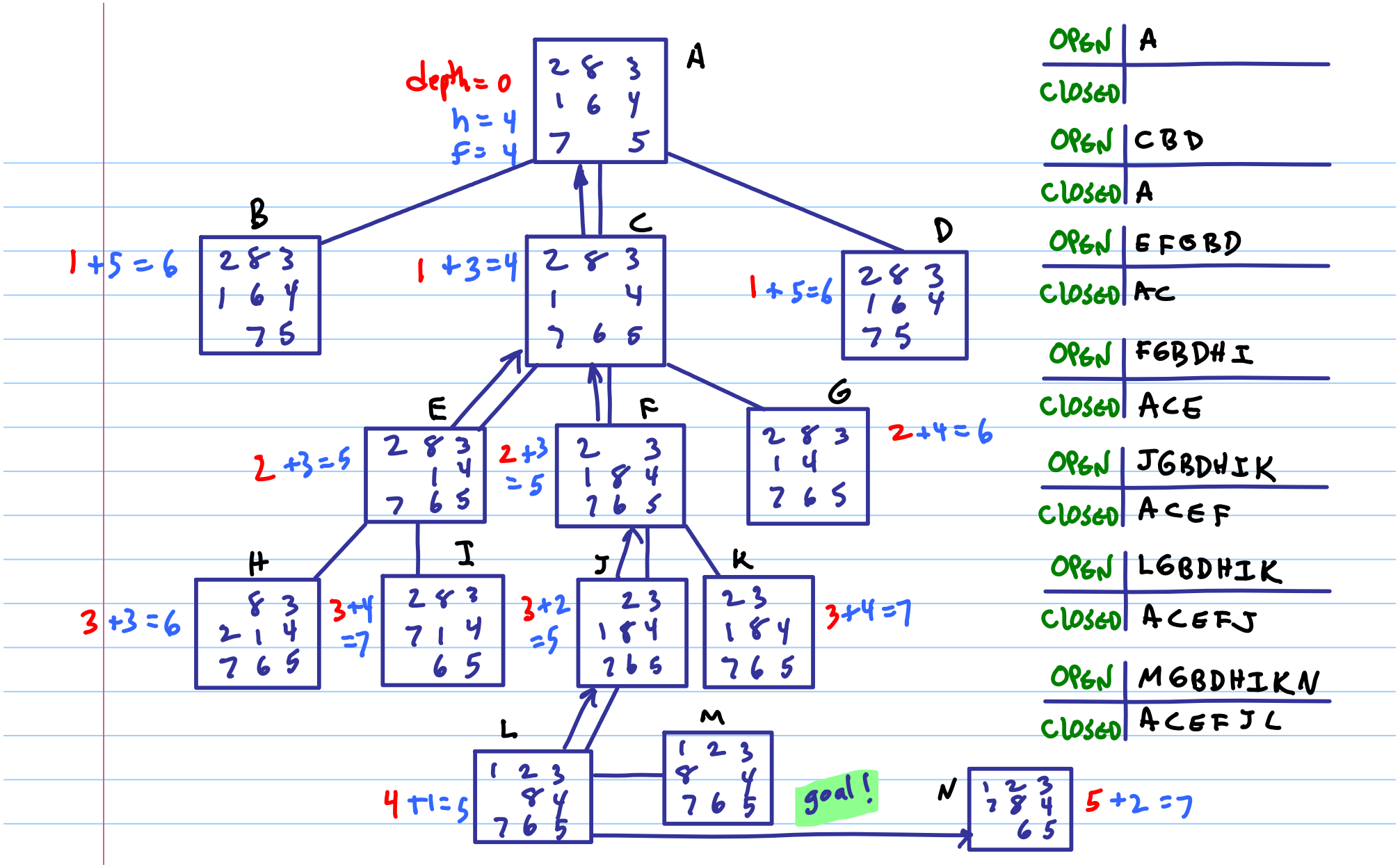
OPEN	E F G B D
CLOSED	A C

OPEN	F G B D H I
CLOSED	A C E

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



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


depth = 0
h = 4
f = 4
A
B
C
D
E
F
G
H
I
J
K
L
M
N
1 + 5 = 6
1 + 3 = 4
1 + 5 = 6
2 + 3 = 5
2 + 3 = 5
2 + 4 = 6
3 + 3 = 6
3 + 4 = 7
3 + 2 = 5
3 + 4 = 7
4 + 1 = 5
goal!
5 + 2 = 7
OPEN A
CLOSED
OPEN CBD
CLOSED A
OPEN EFGBD
CLOSED AC
OPEN FGBDHI
CLOSED ACE
OPEN JGBDHIK
CLOSED ACEF
OPEN LGBDHIK
CLOSED ACEFJ
OPEN MGBDHIKN
CLOSED ACEFJL

Best-First Search using Algorithm A*

- finds a shortest-path solution to a goal

- requires fewer node expansions than breadth-first search (an admissible heuristic guarantees you won't go deeper than a goal node, so you can't expand more nodes than breadth first search)

- requires fewer node expansions than depth-first search (unless you were lucky - a good heuristic will let you choose properly which nodes to examine next, rather than rely on luck)

- should have improved performance with an improved heuristic