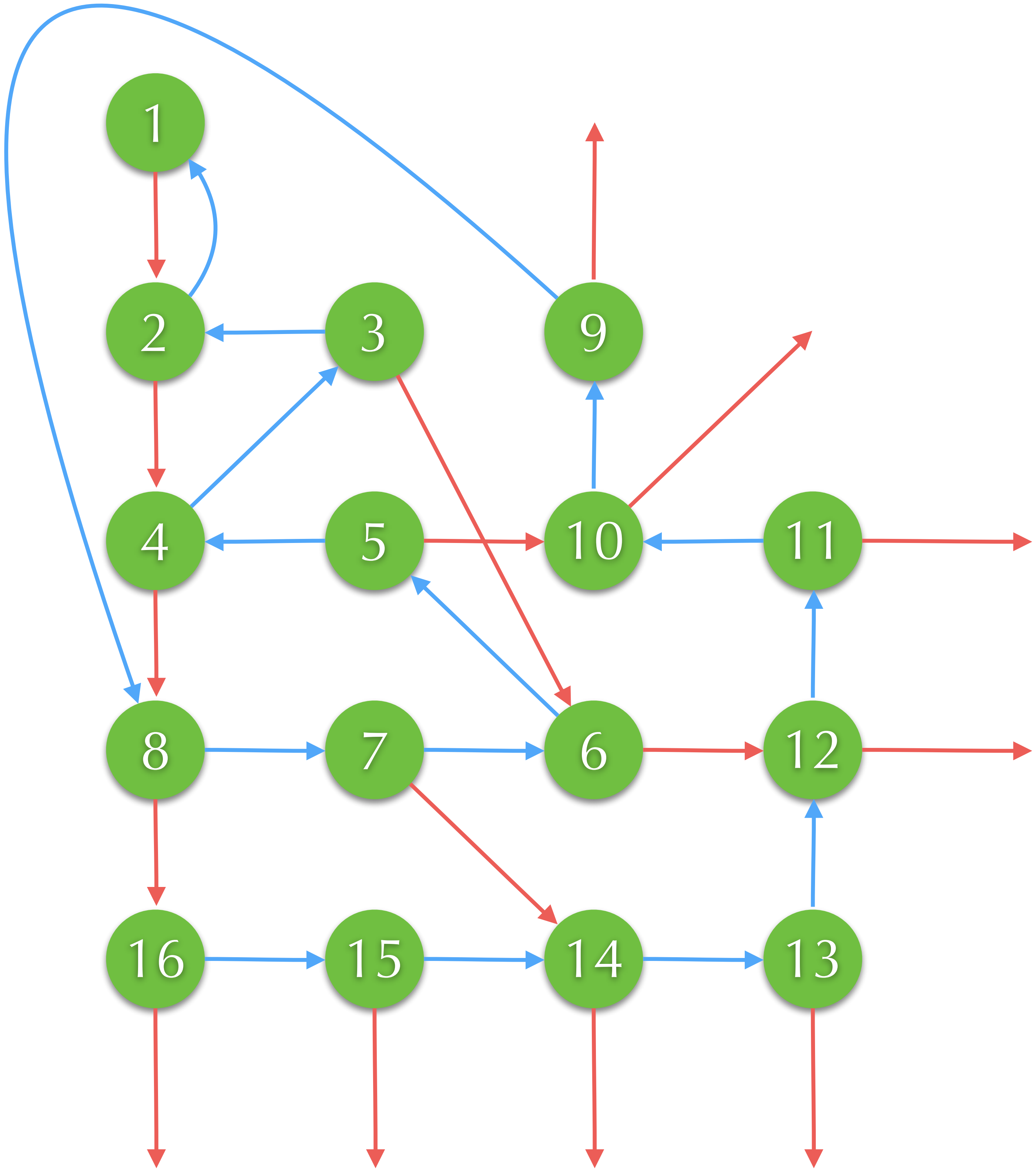
State Space Search

Learn from Example: CodeForces 520B

- A device initially displays an integer n>0.
- Red button doubles the displayed number
- Blue button decrease the number by 1
  - Cannot be clicked when the device displays 1.
- Question: given another integer m, how many clicks are required to change the displayed number into m?

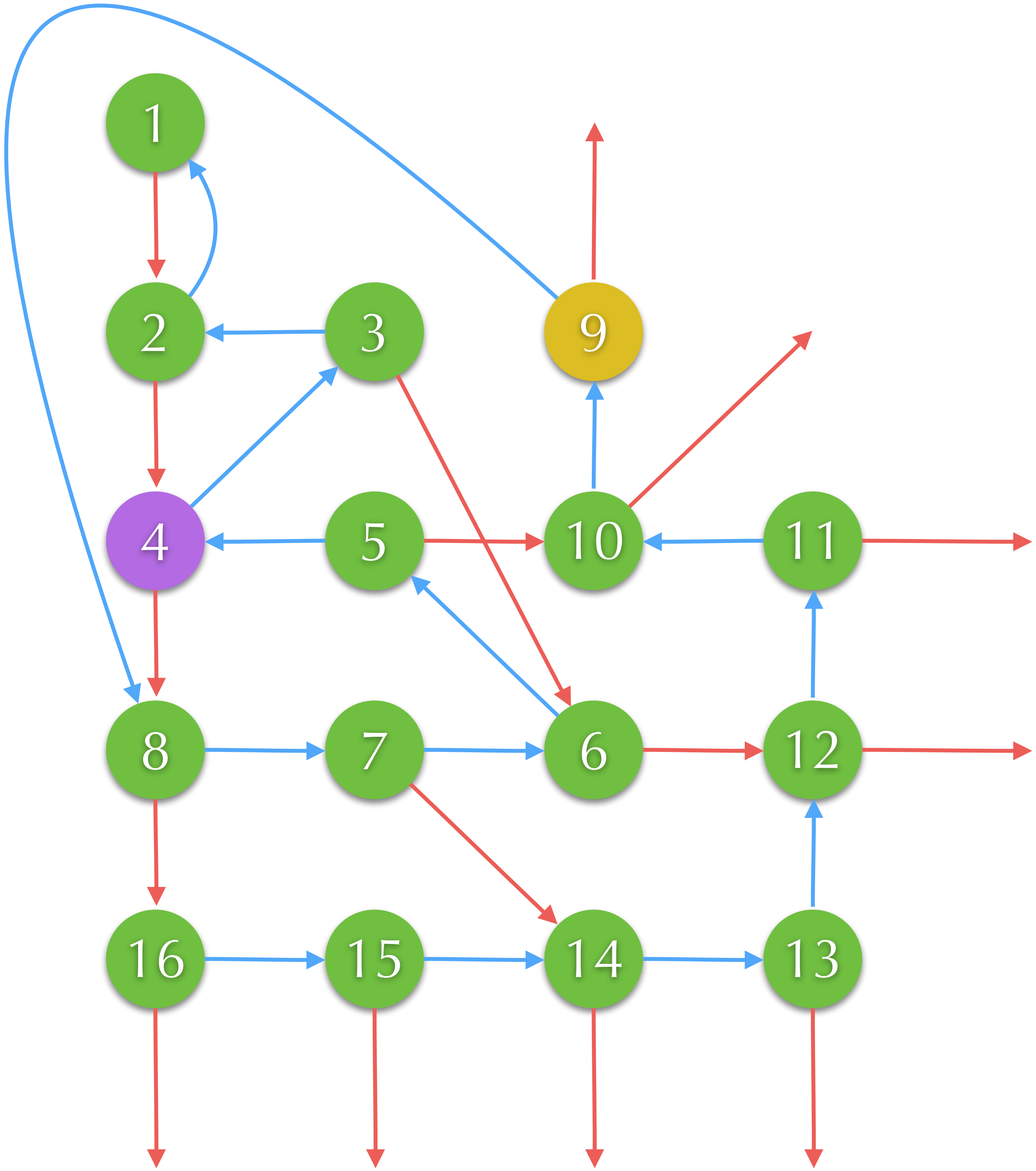
State Space

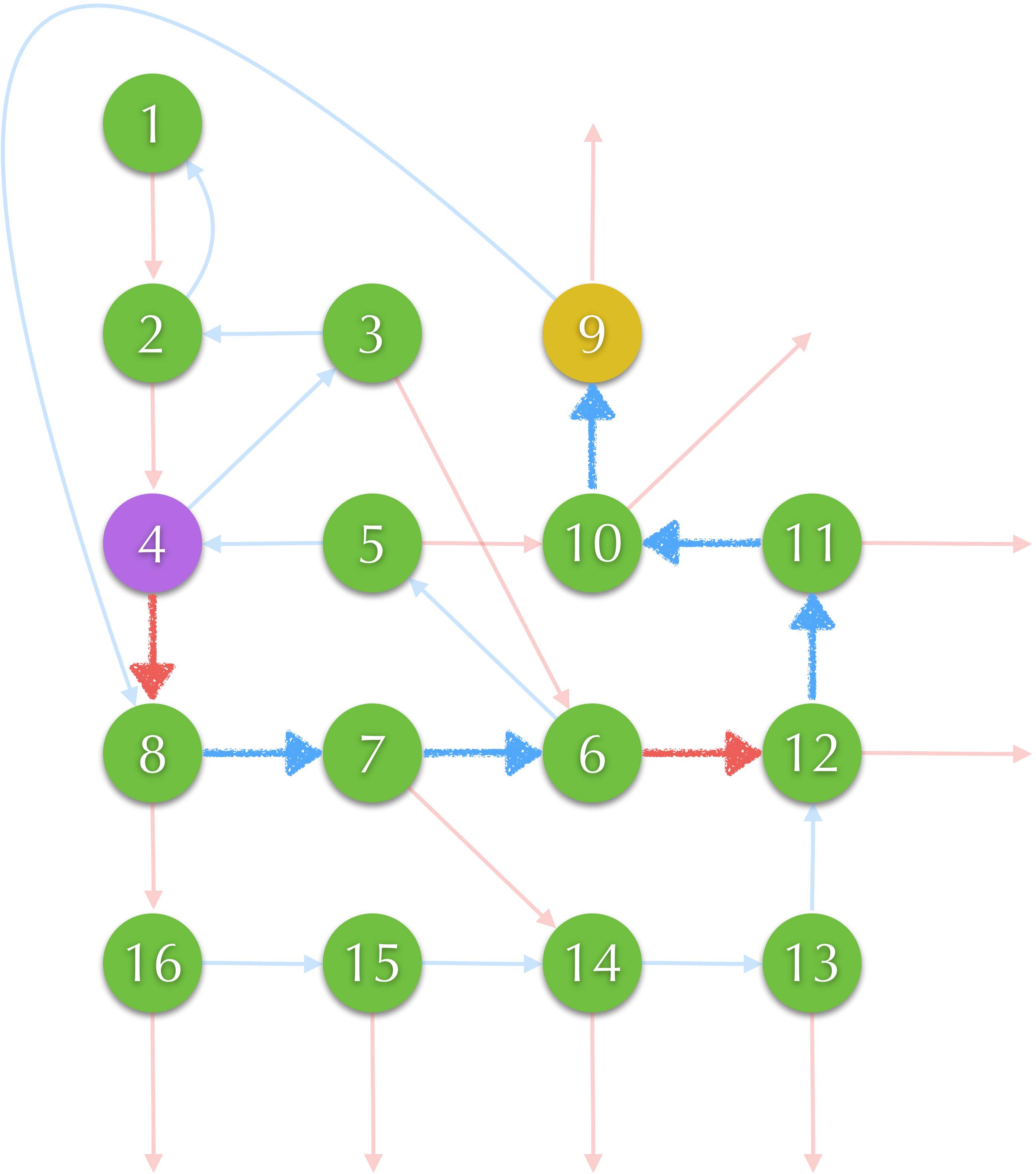
- State: the displayed number
- State space: set of all possible displayed numbers
    - Note: not necessarily finite!
- Transition
    - Action: clicking a button
    - Result: the new number
    - Cost: 1 (not necessarily uniform)

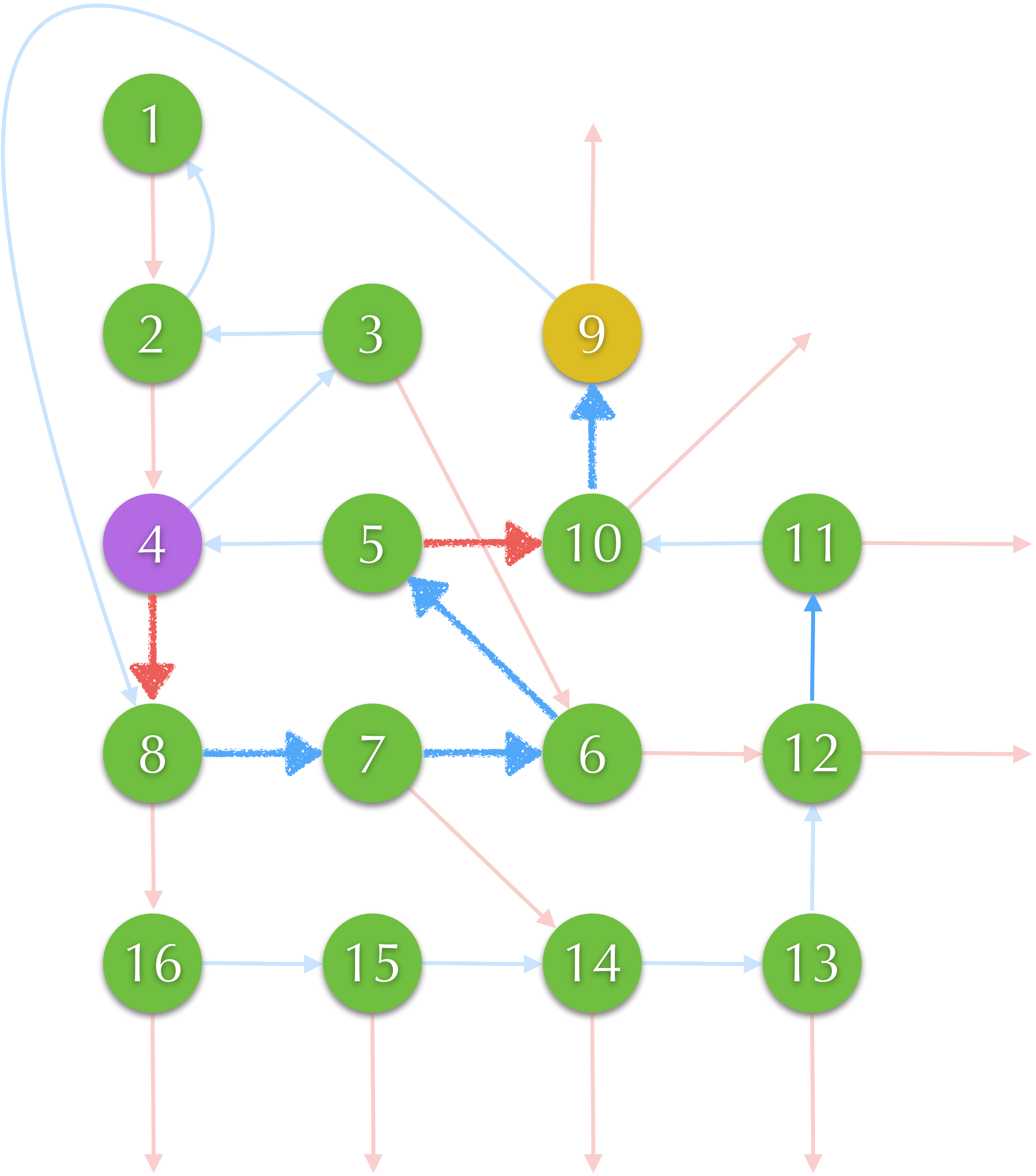

1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13

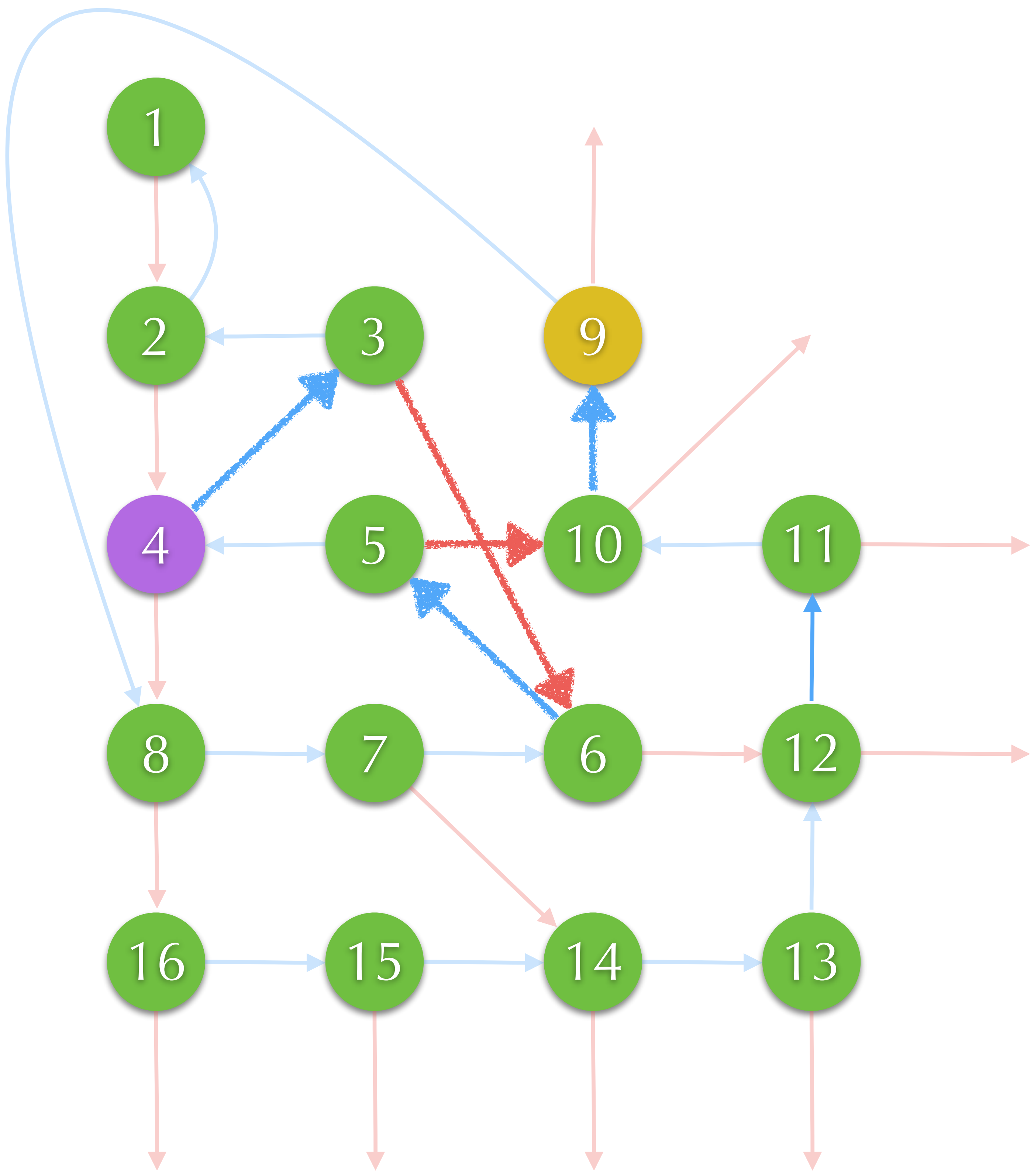
State Space Search

- Initial state
  - The initially displayed number n
- Goal state
  - The desired number m
- Find a sequence of transition from the initial state to the goal state.
- Sometimes there are many goal states
  - Ex: finding all states satisfying certain criteria.

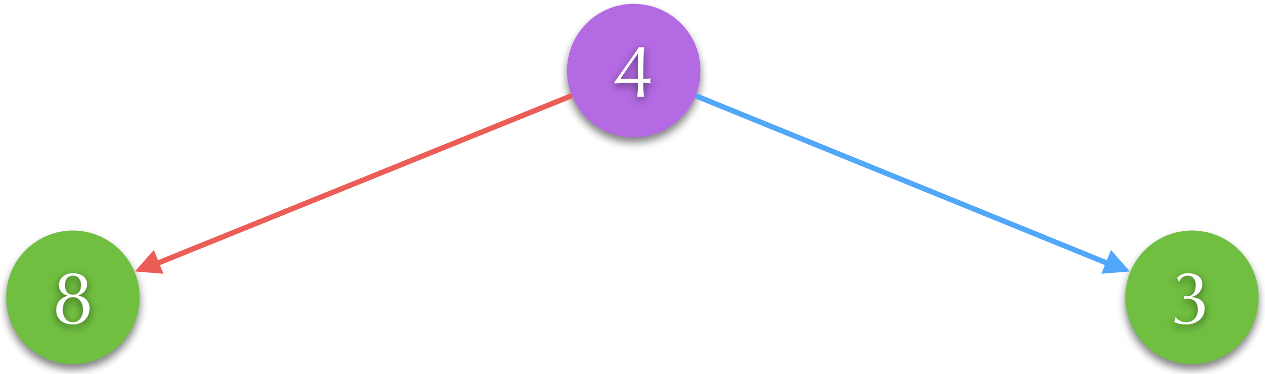

1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13


1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13

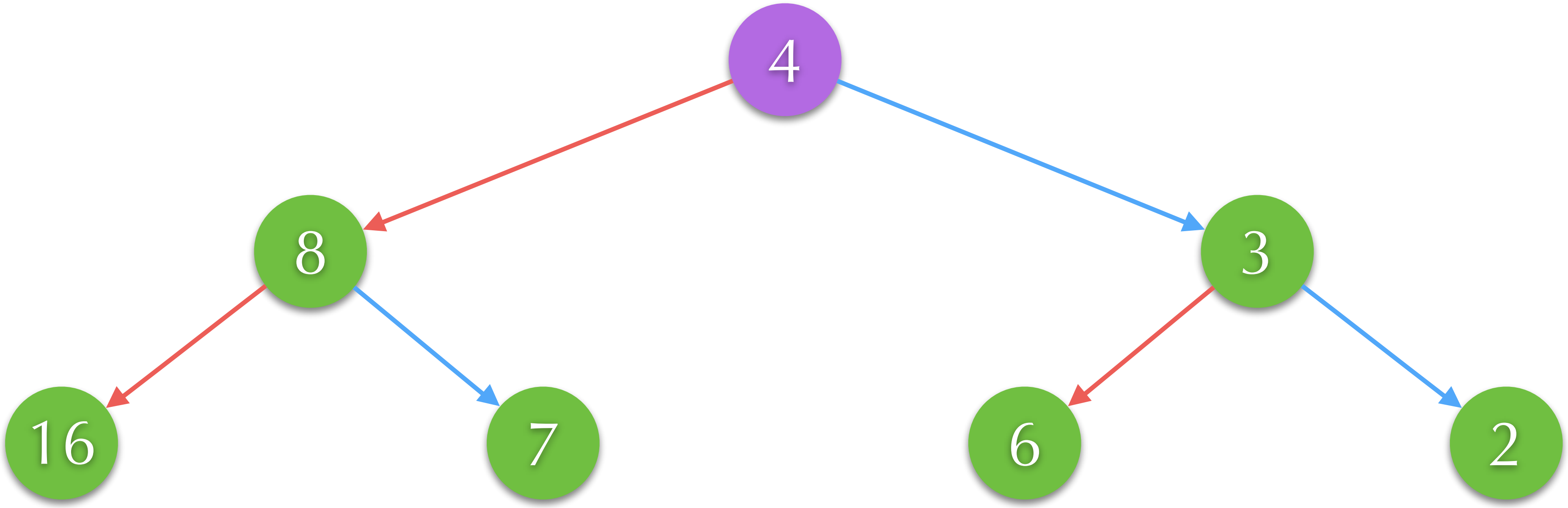

1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13


1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13

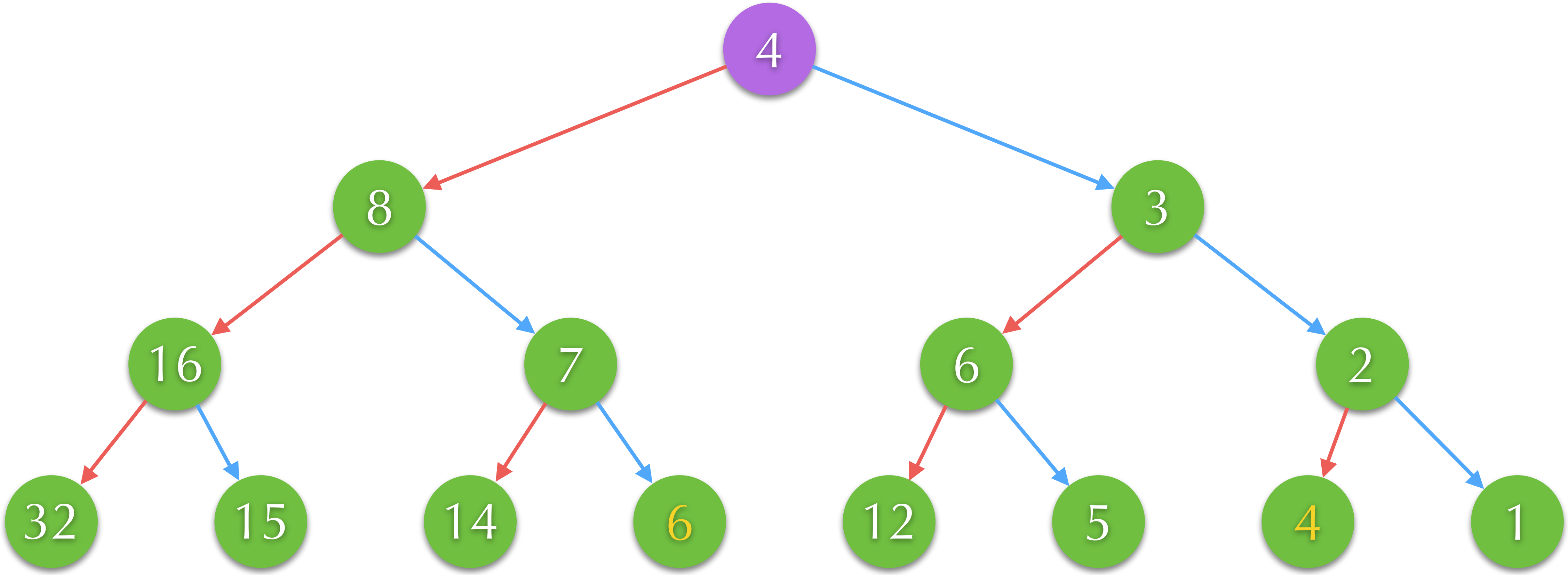
Search Tree



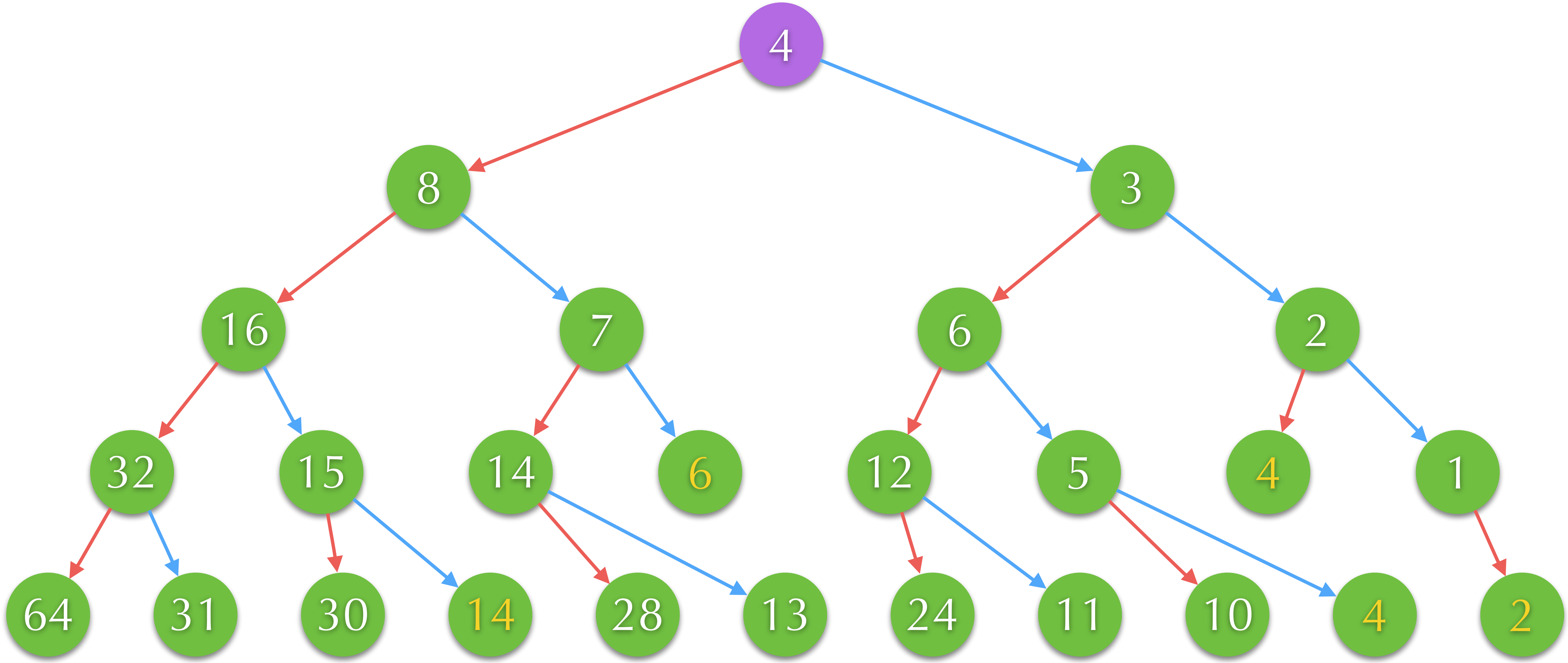
Search Tree



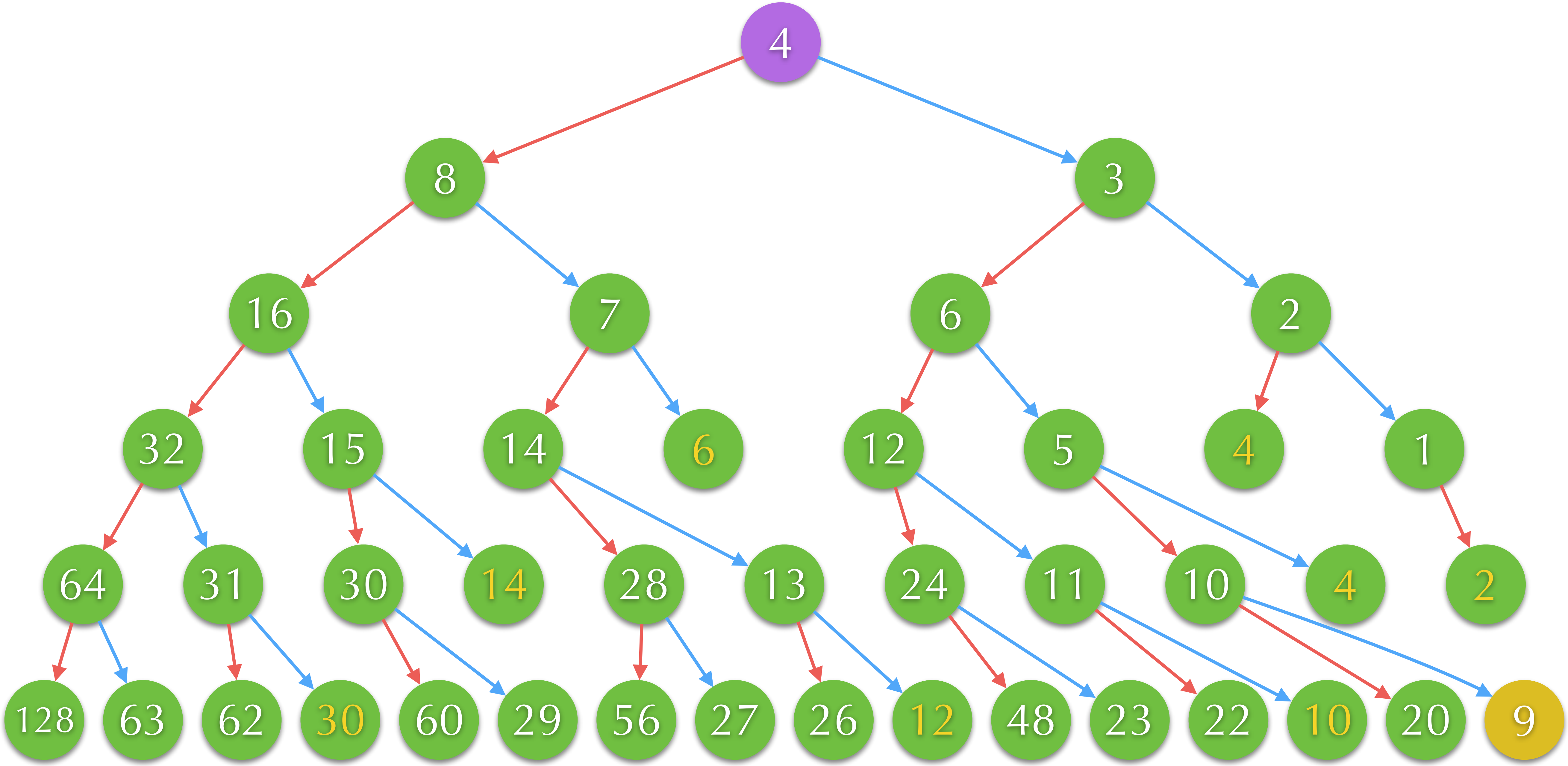
Search Tree



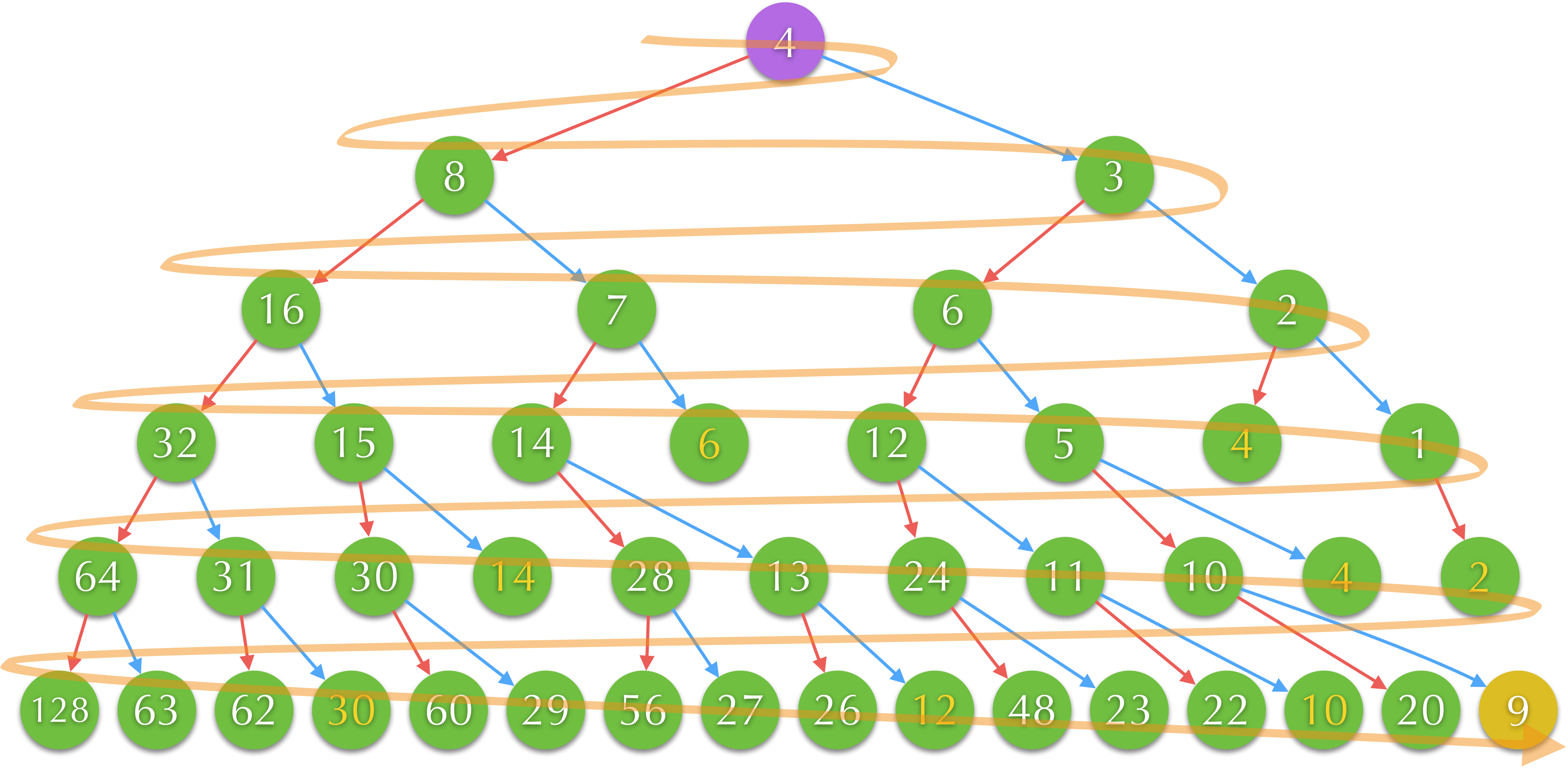
Search Tree



Search Tree


4
8
3
16
7
6
2
32
15
14
6
12
5
4
1
64
31
30
14
28
13
24
11
10
4
2
128
63
62
30
60
29
56
27
26
12
48
23
22
10
20
9

Breadth First Search

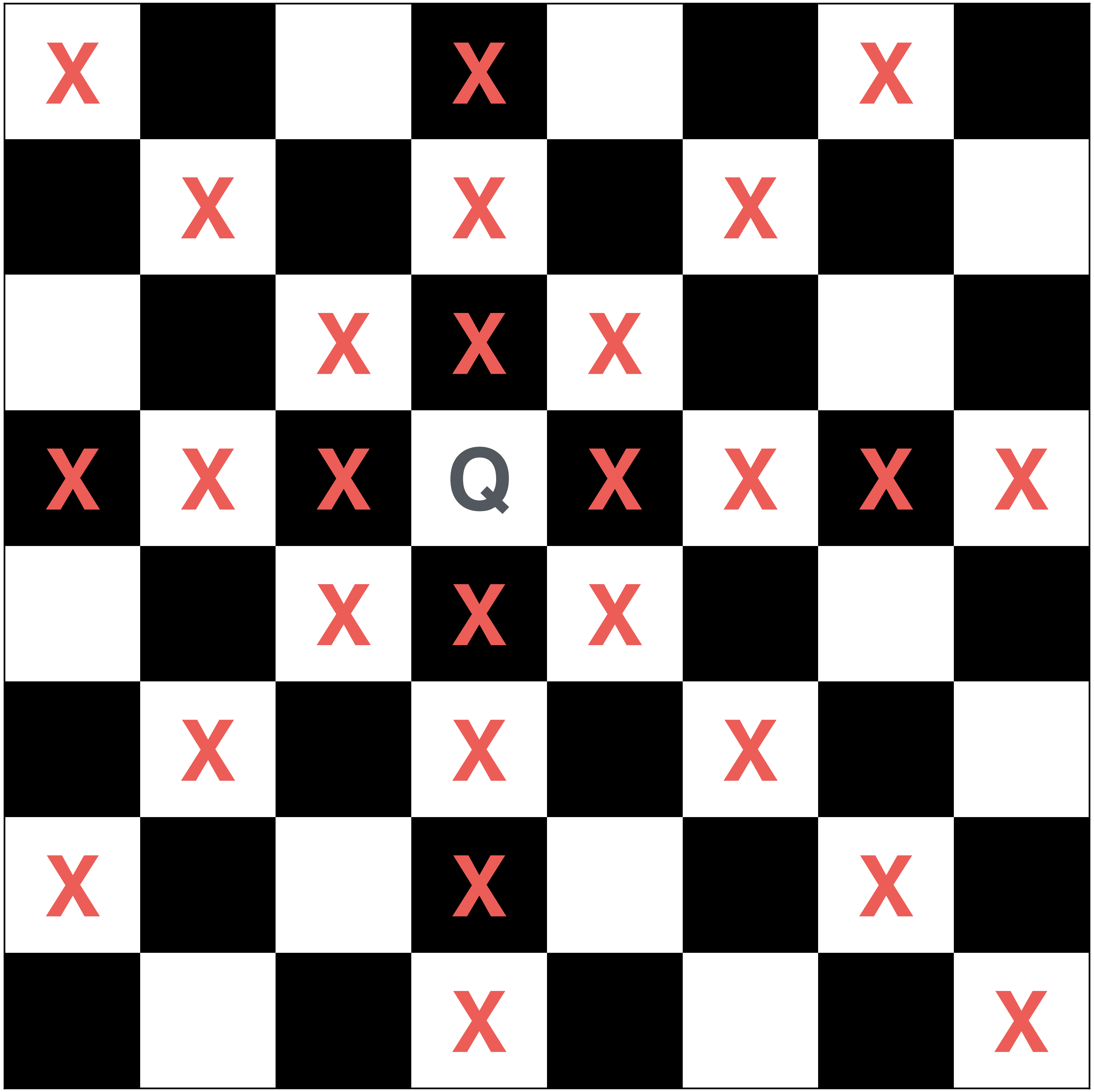


Breadth First Search

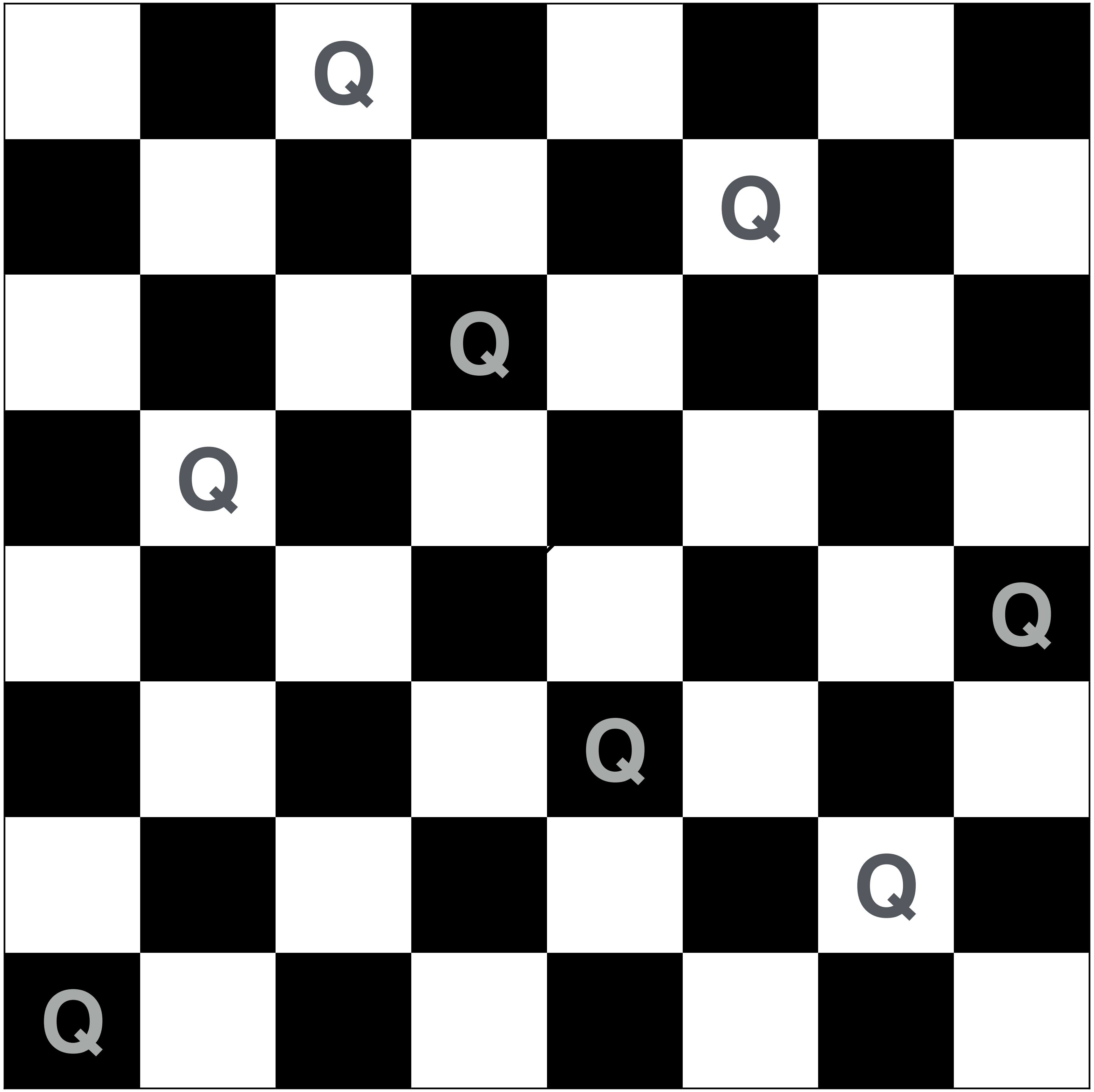
- Pros
  - Easy to implement
  - Can reach the goal state with minimum transitions
- Cons
  - May consume a lot of memory
  - May visit too many states

Learn from Example: n Queens Problem

- Queen can move any number of squares vertically, horizontally, or diagonally.
- Place queens on an n-by-n chess board. No two queens can take one another.
- Various questions:
  - How many queens can be placed?
  - How many kinds of valid placements?

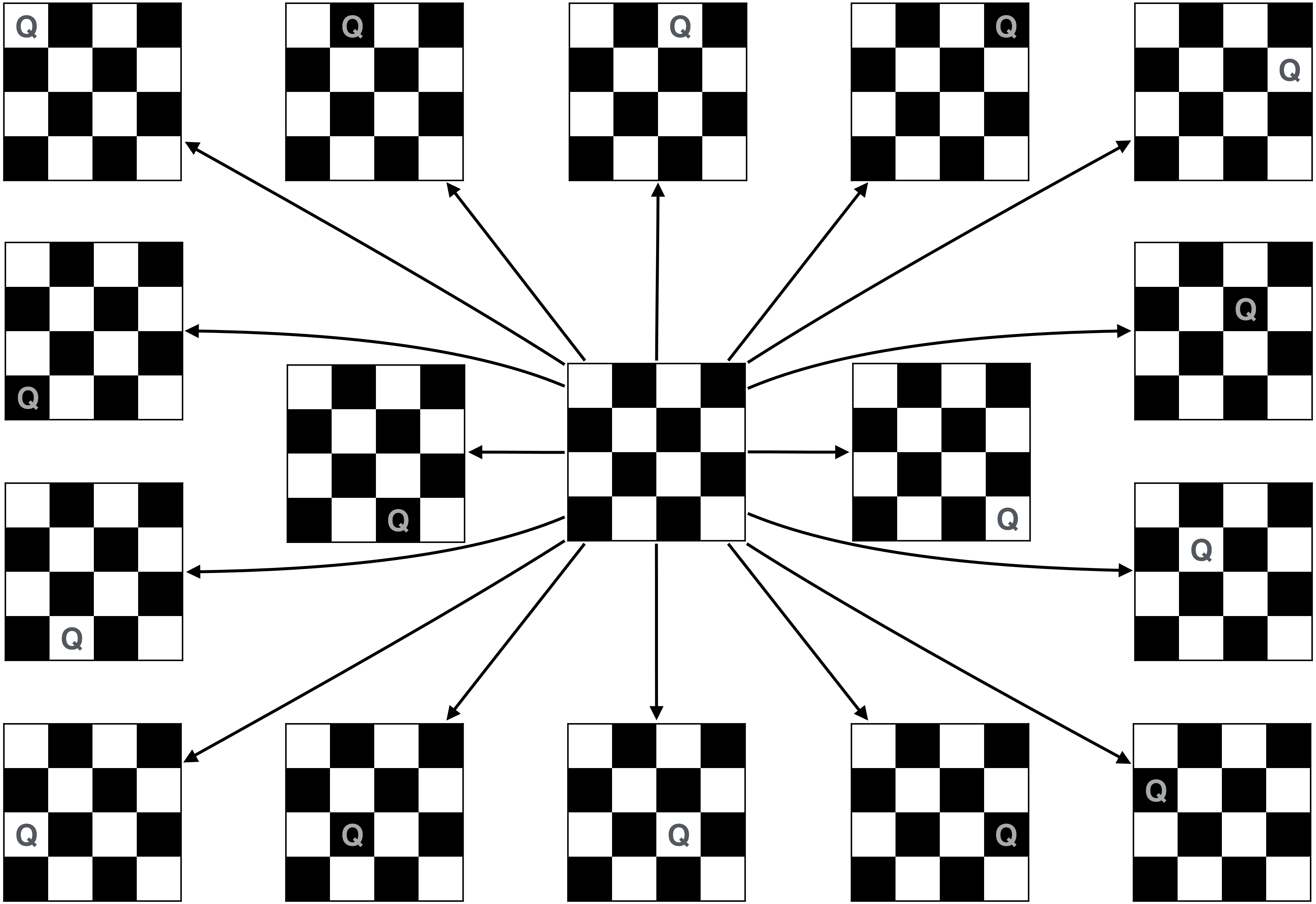


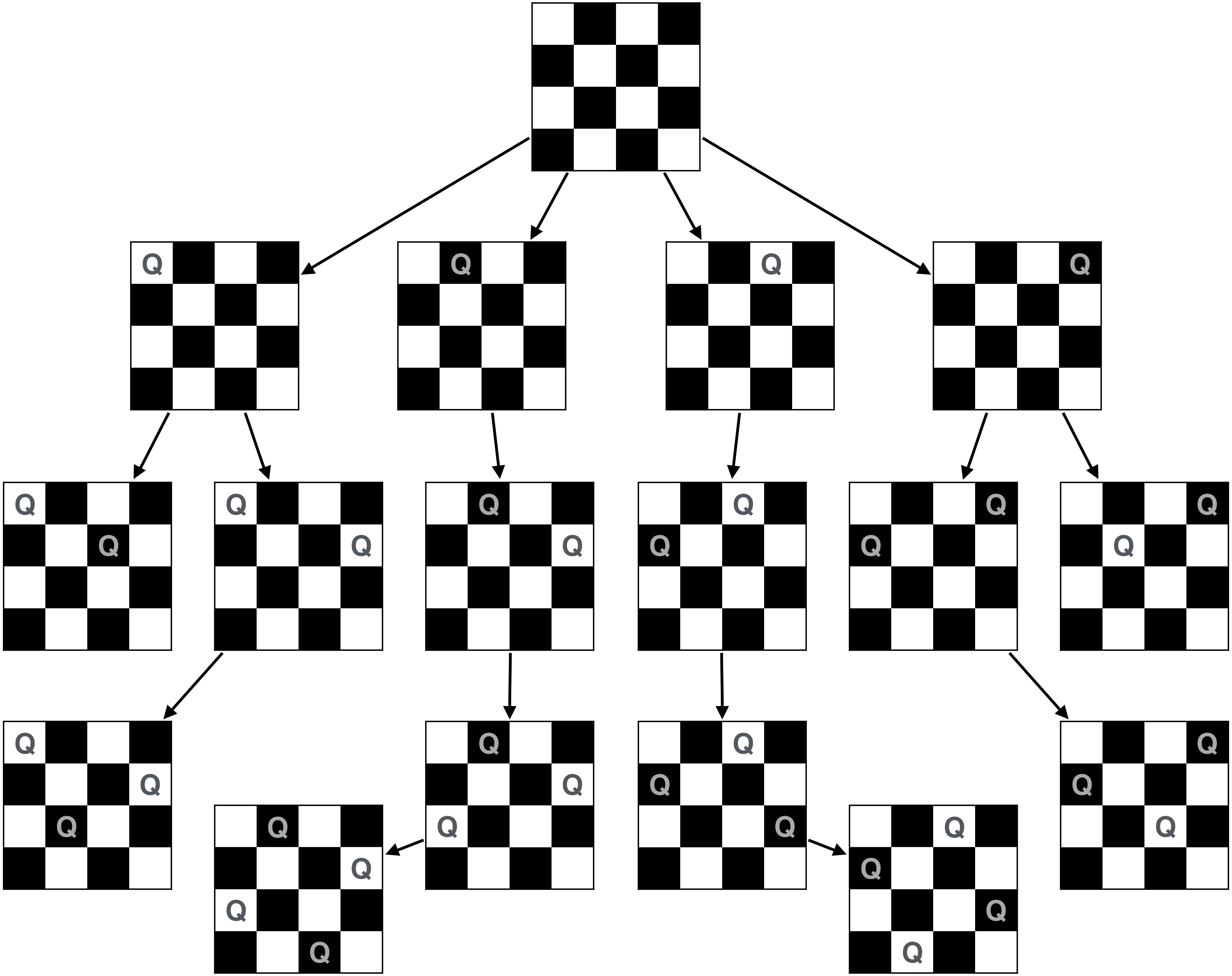
X			X			X	
	X		X		X		
		X	X	X			
X	X	X	Q	X	X	X	X
		X	X	X			
	X		X		X		
X			X			X	
			X				X


Q
Q
Q
Q
Q
Q
Q
Q

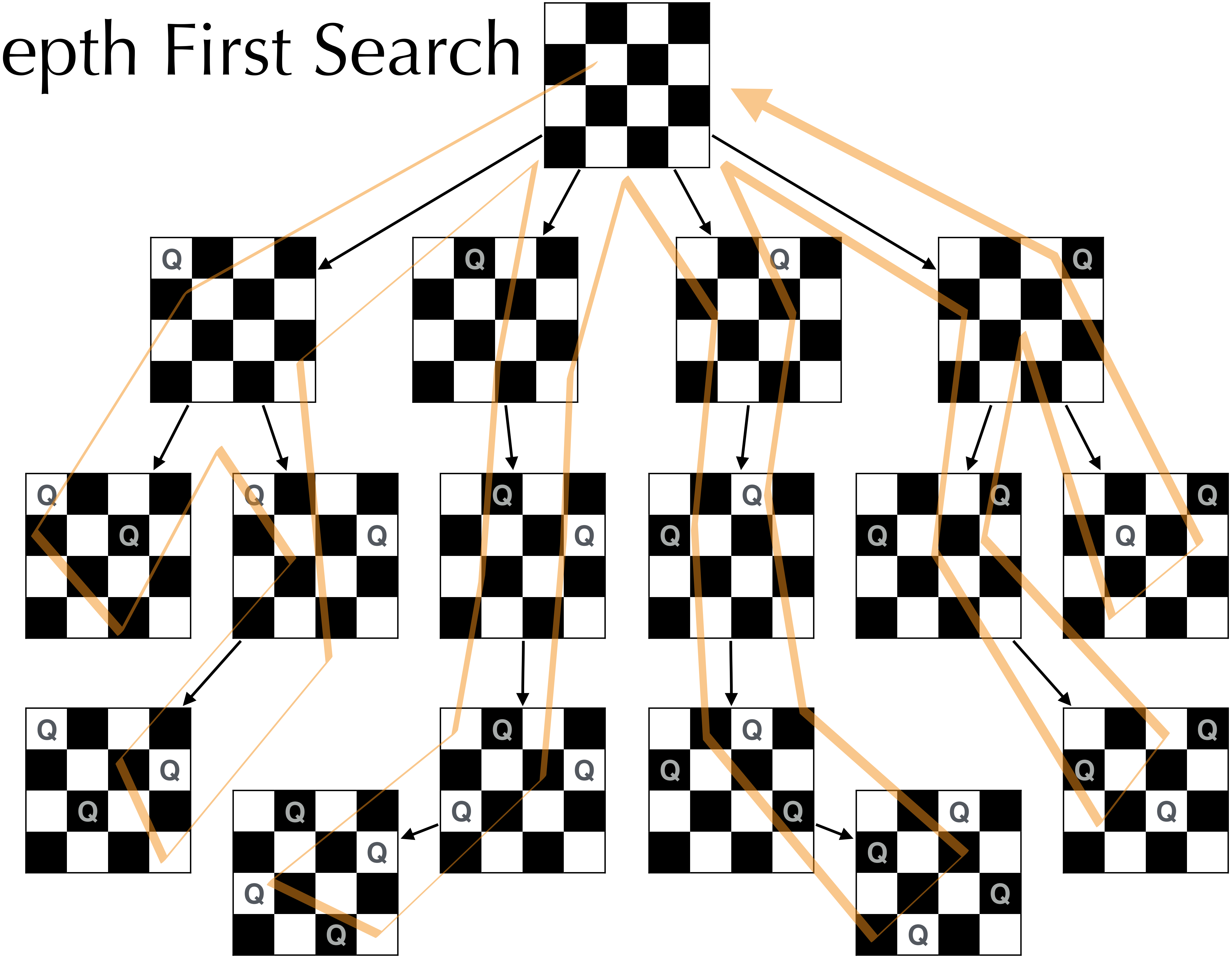
State Space

- State: a (valid) placement
- State space: set of all (valid) placement
  - This is finite but very large.
- Transition
  - Action: try to place a new queen on certain place (* add restriction)
  - Result: the new placement
  - Cost: 1 (* depends on the question)


Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q


Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q
Q

Depth First Search



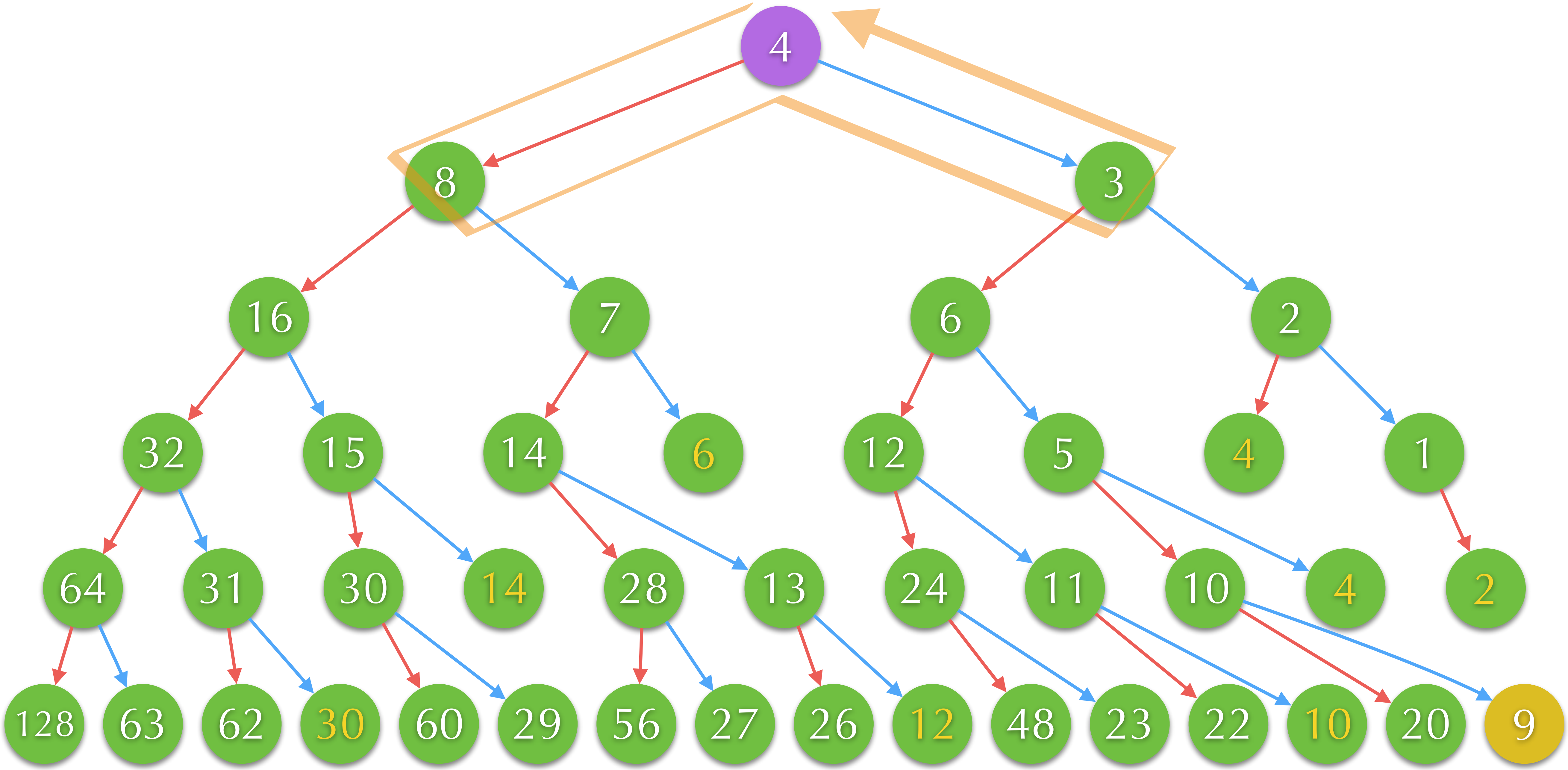
Depth First Search

- Pros
  - Easy to implement
  - Less memory consumption
  - Works for non-uniform costs
- Cons
  - Cannot reach the goal state with minimum transitions
  - May visit even more states than BFS

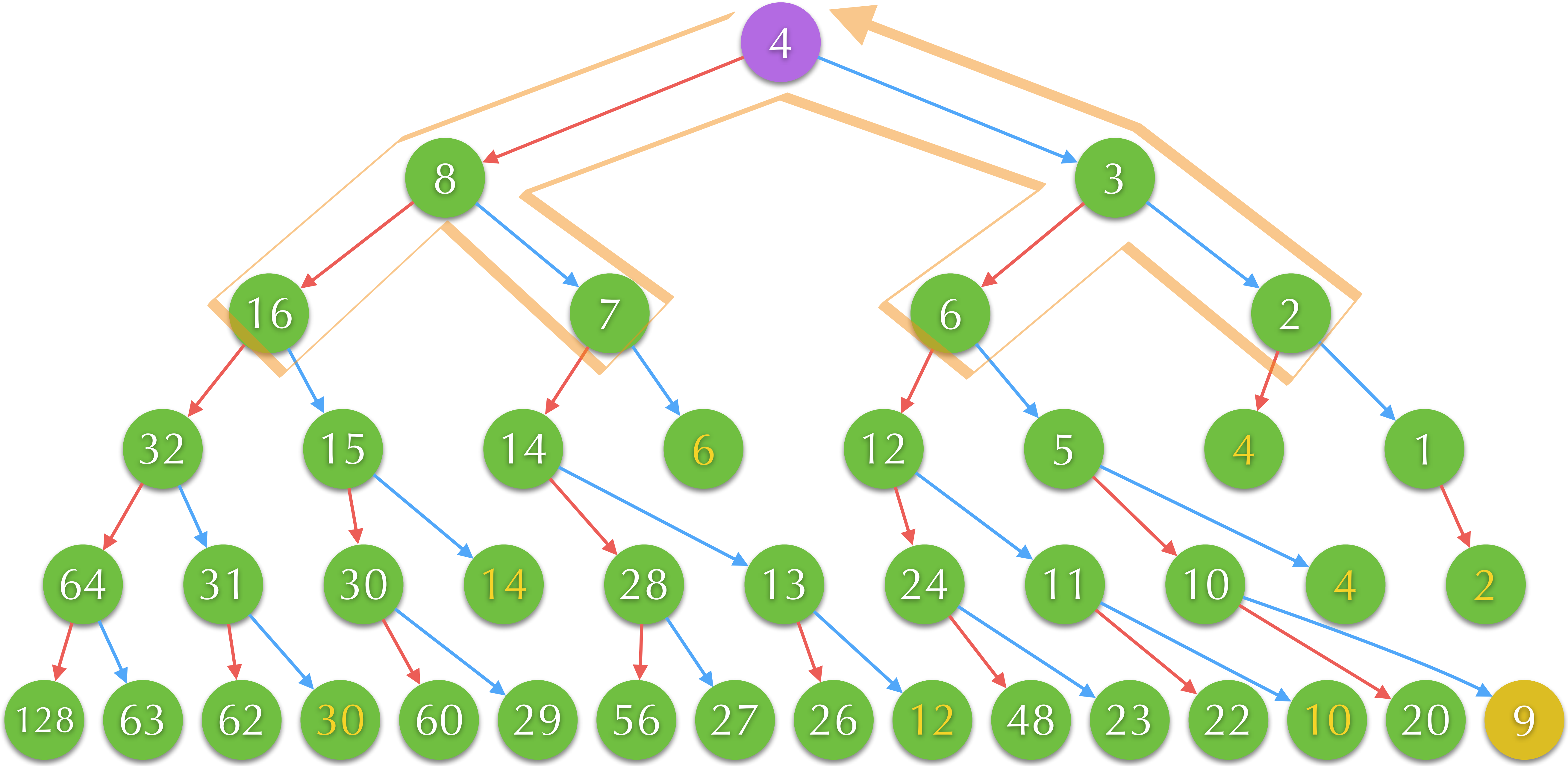
Iterative Deepening

- Simulate BFS by multiple modified DFSs
- Pros
  - Still easy to implement
  - Less memory consumption than BFS
  - Works for non-uniform costs
- Cons
  - Slower than BFS & more complex than DFS

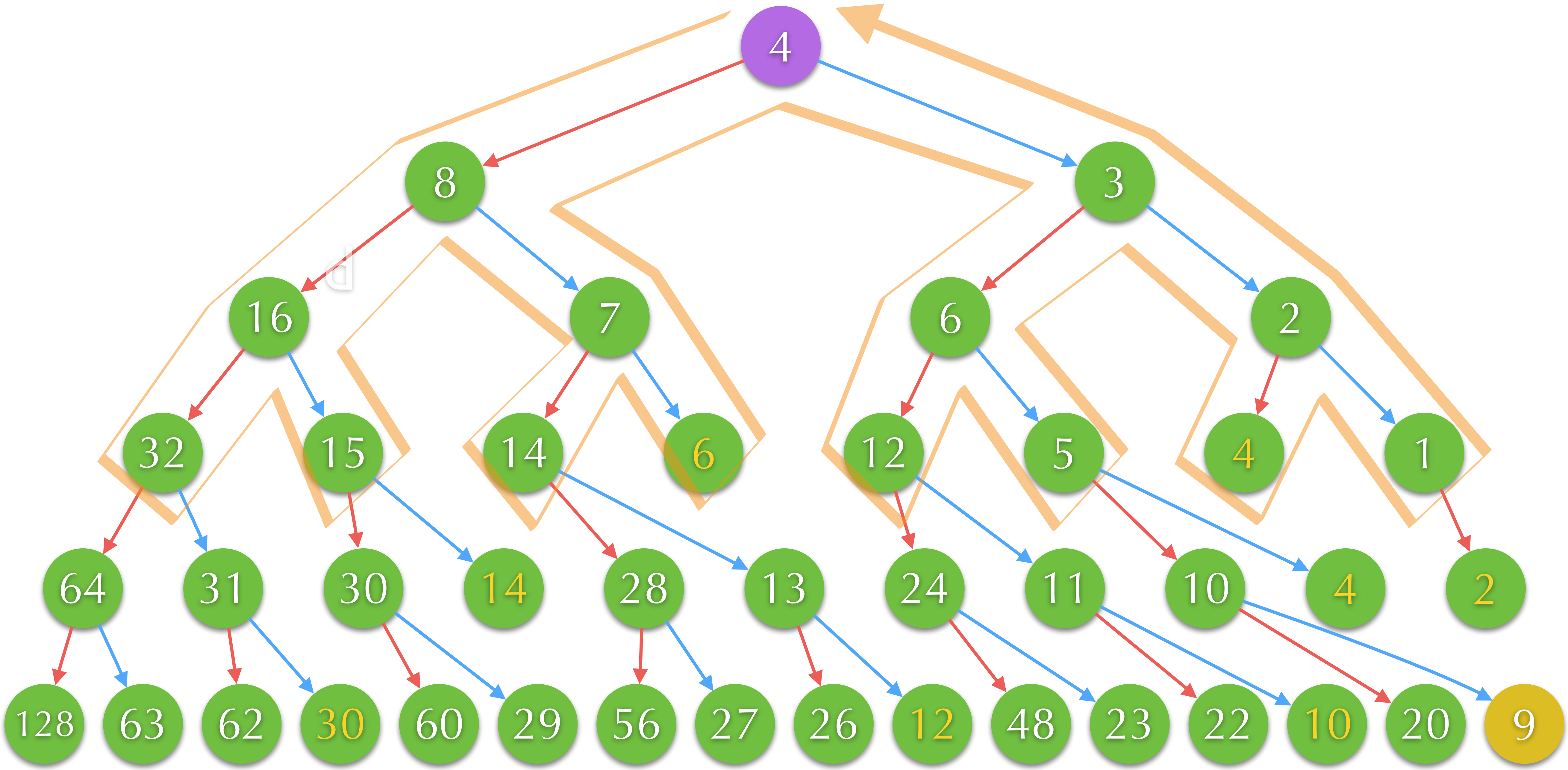
Iterative Deepening



Iterative Deepening

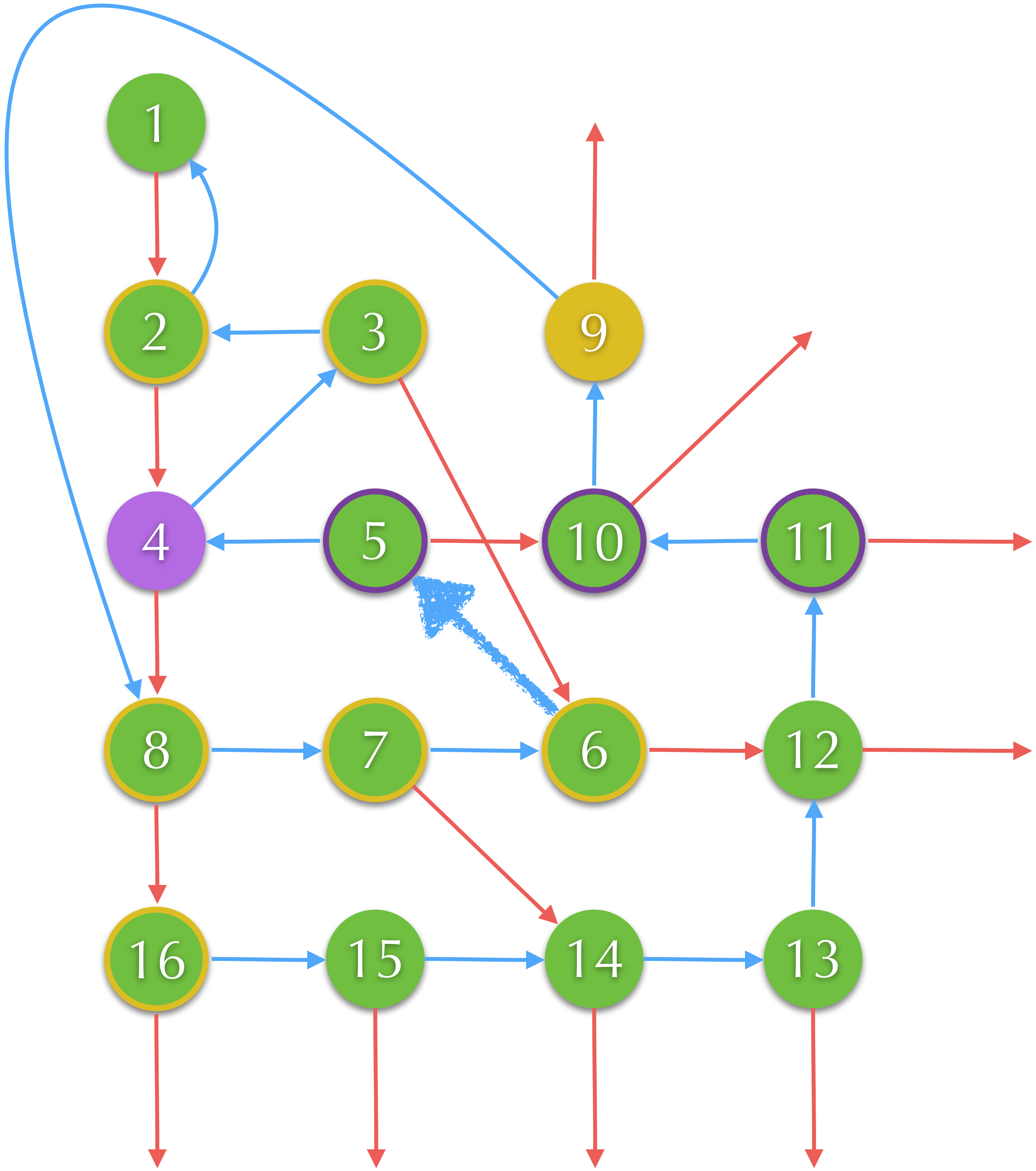


Iterative Deepening



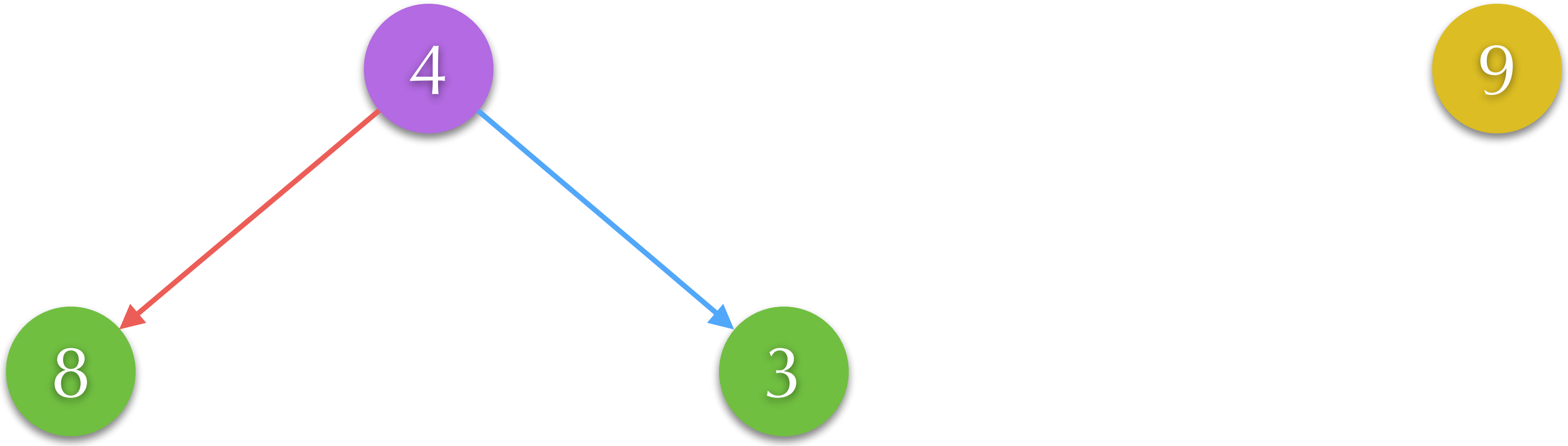
Assignment
Week 2

- UVa 11974
    - bitwise operators
    - stack overflow
        - Global variable (NG for SW development)
        - Static variable
        - vector (highly recommended)
        - Dynamic allocation
- UVa 167
    - format control: %5d

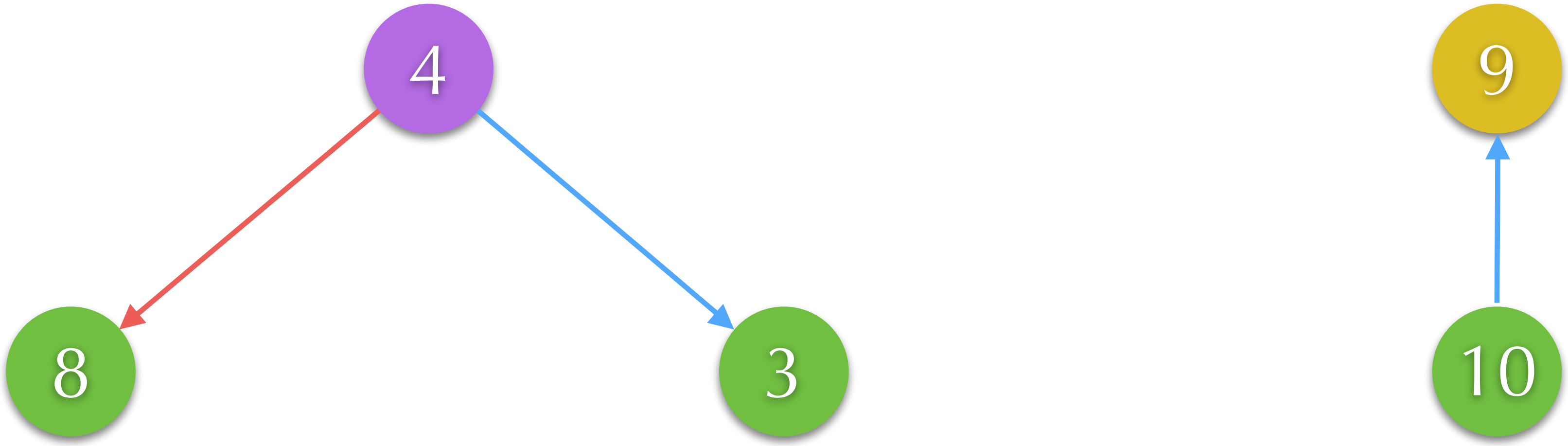

1
2
3
9
4
5
10
11
8
7
6
12
16
15
14
13

Meet in the Middle

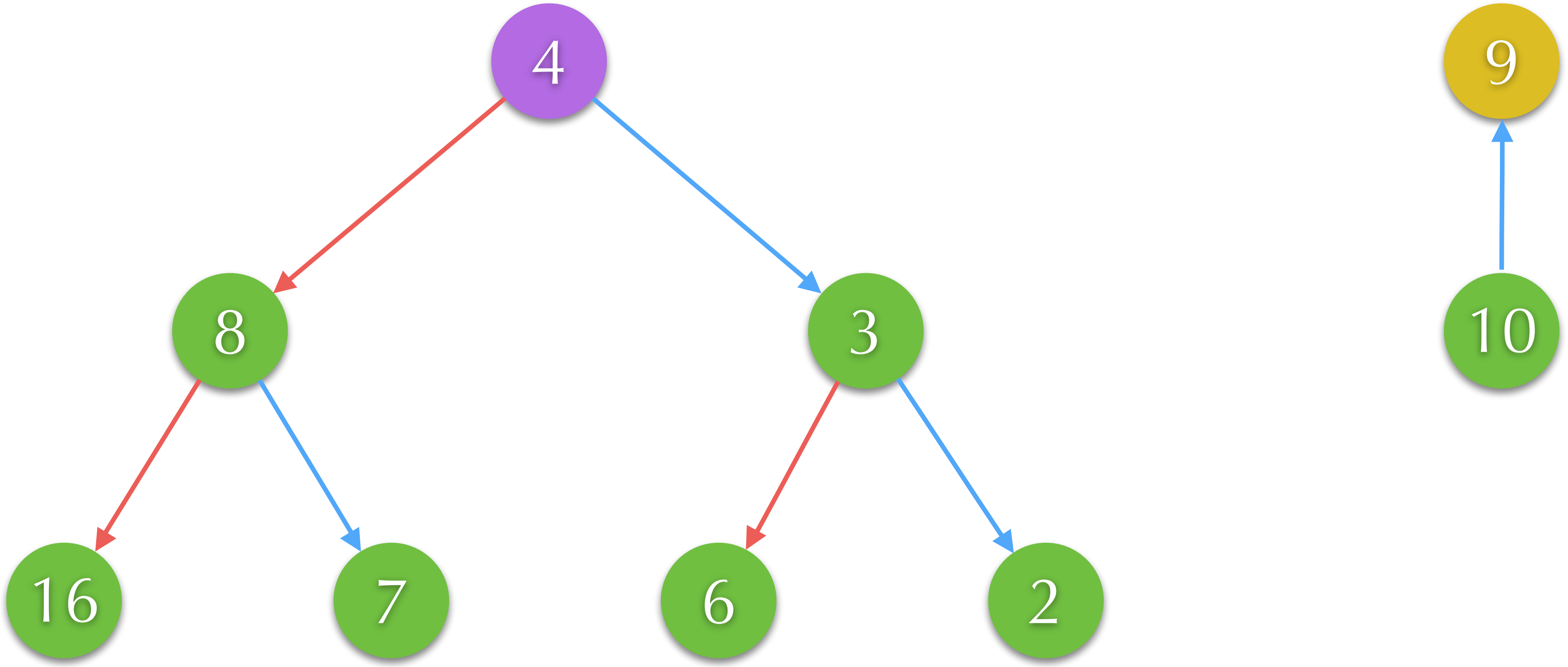
Meet in the Middle



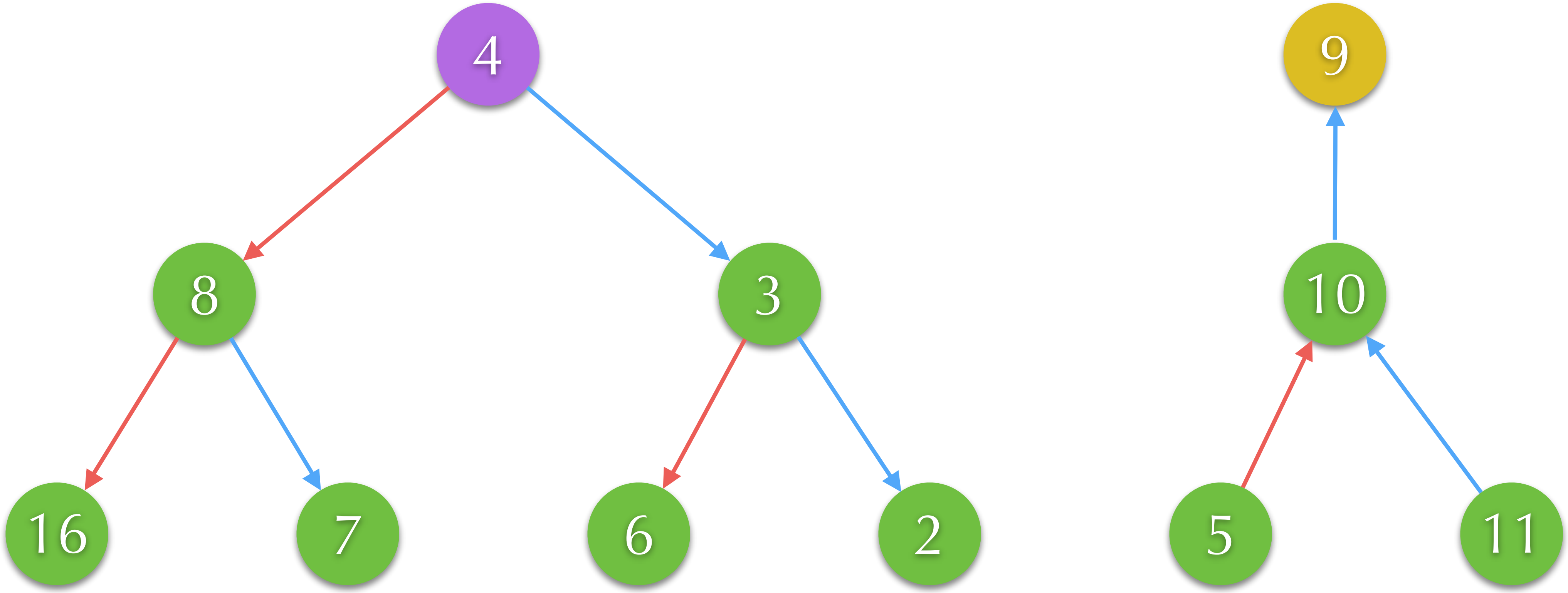
Meet in the Middle



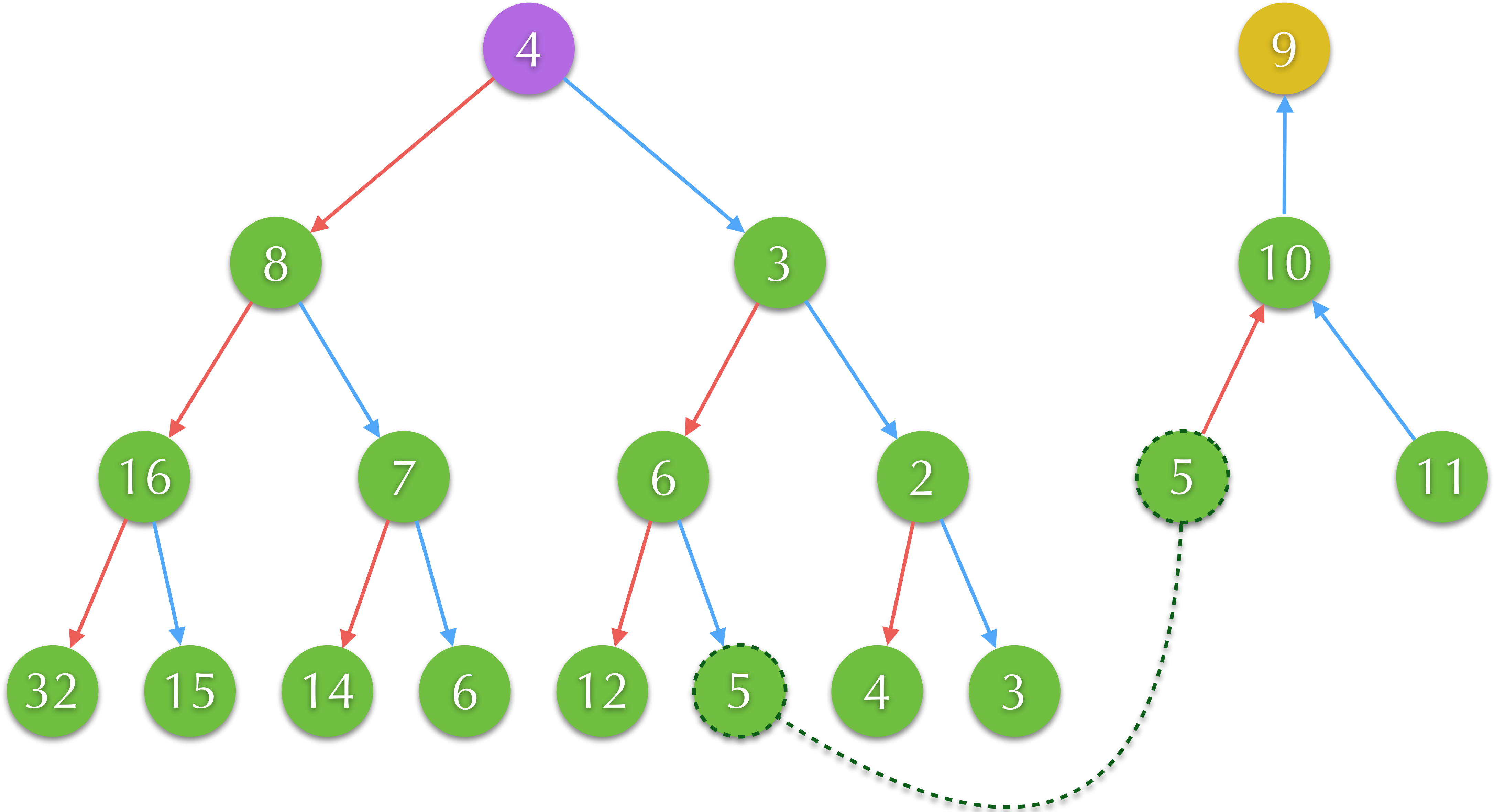
Meet in the Middle



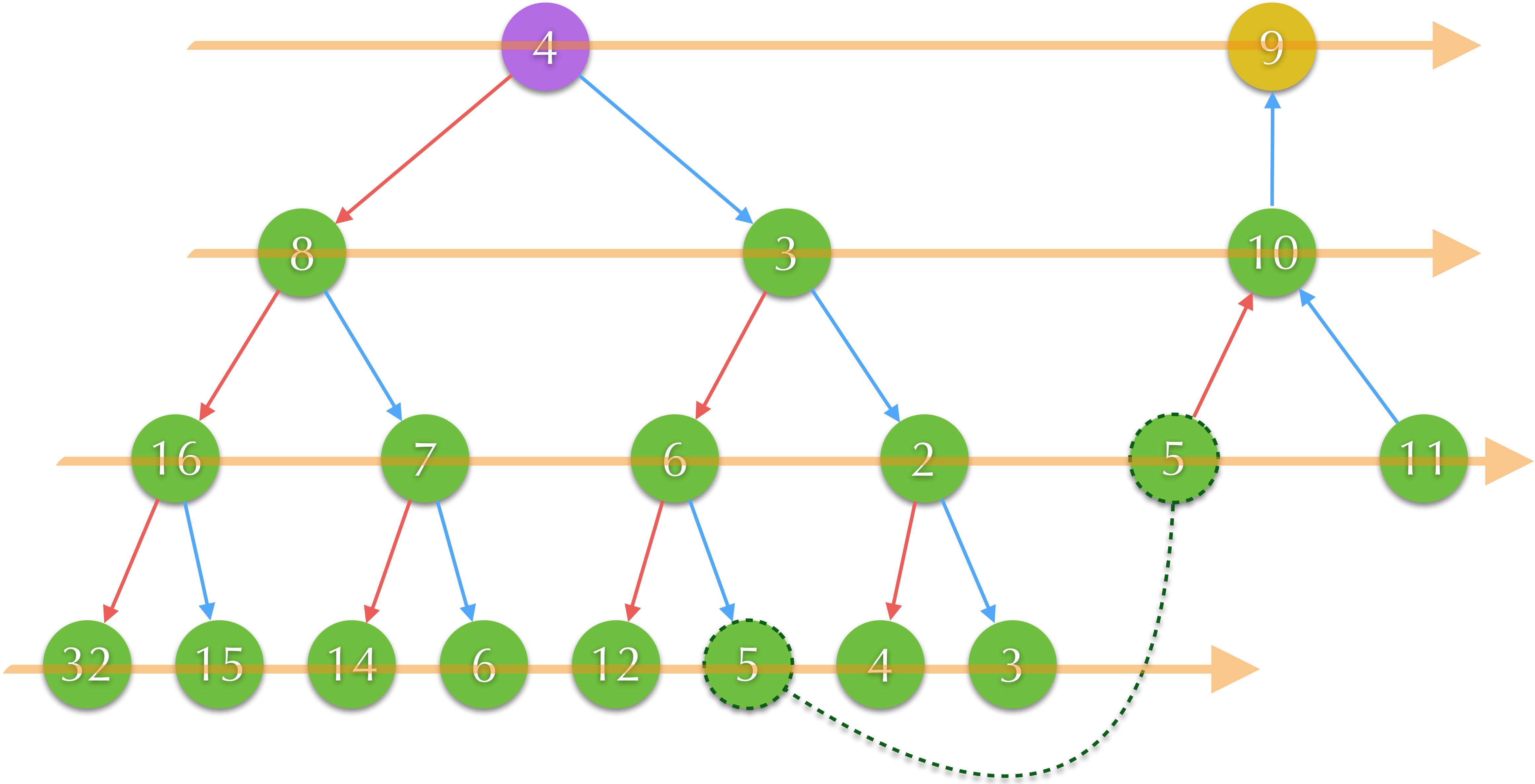
Meet in the Middle



Meet in the Middle



Meet in the Middle



Meet in the Middle

- Perform BFSs on both ends
- Pros
  - Still easy to implement
  - Less memory consumption than ordinary BFS
- Cons
  - More memory consumption than DFS
  - May visit too many states
  - Goal states must be specified

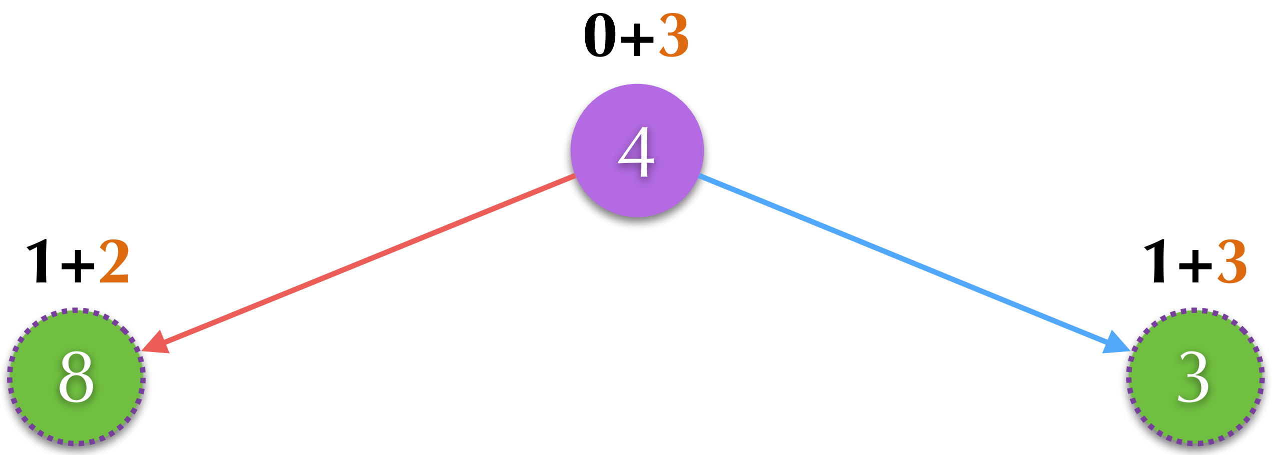
A-star

- Previous approaches do not exploit the COST.
- A-star is a greedy approach: pick the state x of minimum f(x)=g(x)+h(x) to branch
  - g(x): accumulated cost from the initial to x
  - h(x): estimated cost from x to the goal
    - Heuristic function
- If h(x) is admissible (never overestimates), then A-star must find the optimal solution.

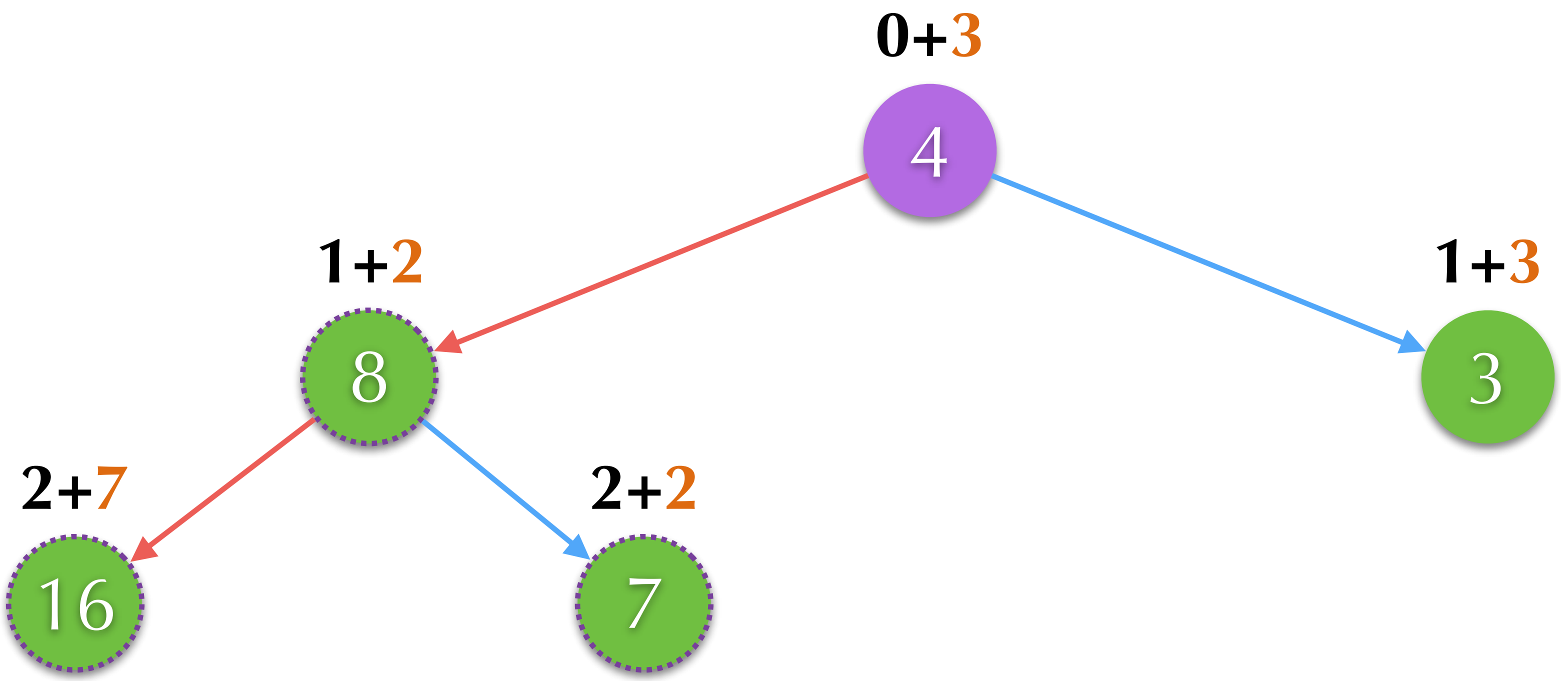
Heuristic for CF 520B

- x: the current displayed number
- m: the goal
- r: m/x
- $h(x)=x-m$ if $x \geq m$.
- $h(x)=\lceil \log_2 r \rceil + I[\log_2 r \notin Z] = 2\lceil \log_2 r \rceil - \lfloor \log_2 r \rfloor$ if $x<m$.
  - $\geq \lceil \log_2 r \rceil$ doubling clicks
  - ≥ 1 minus click if $\log_2 r$ is not integral
- Never overestimate!

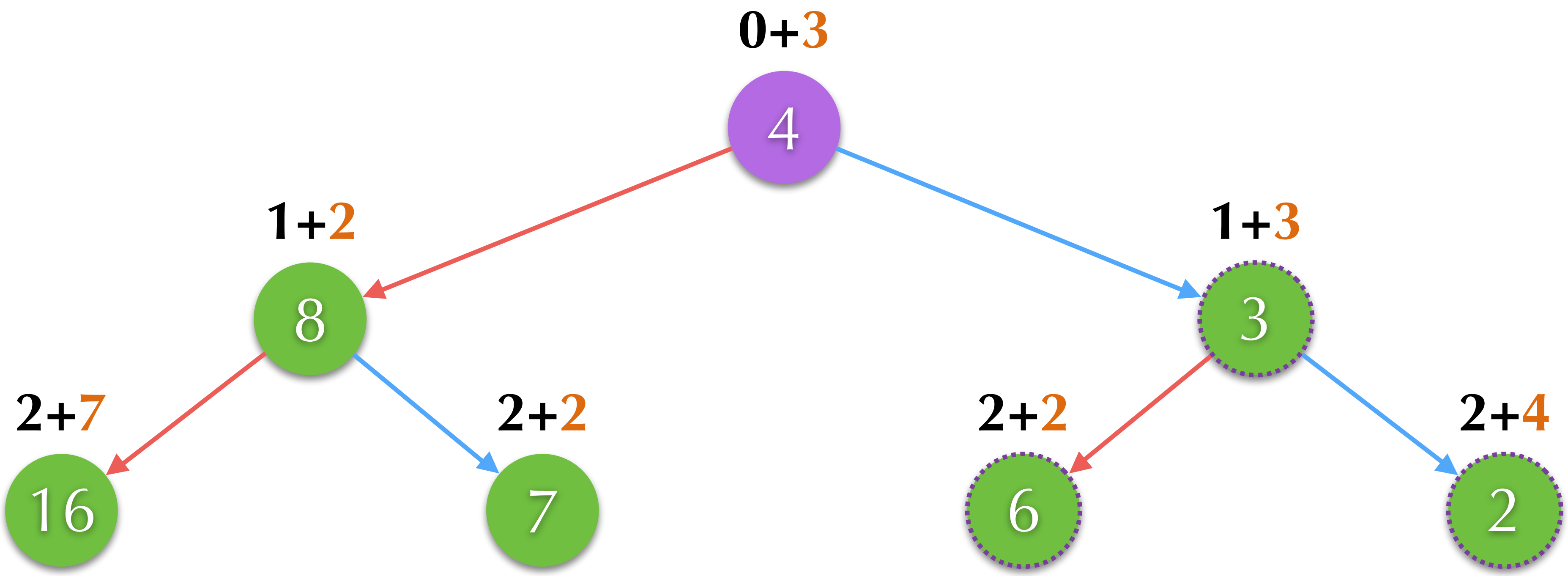
A-star



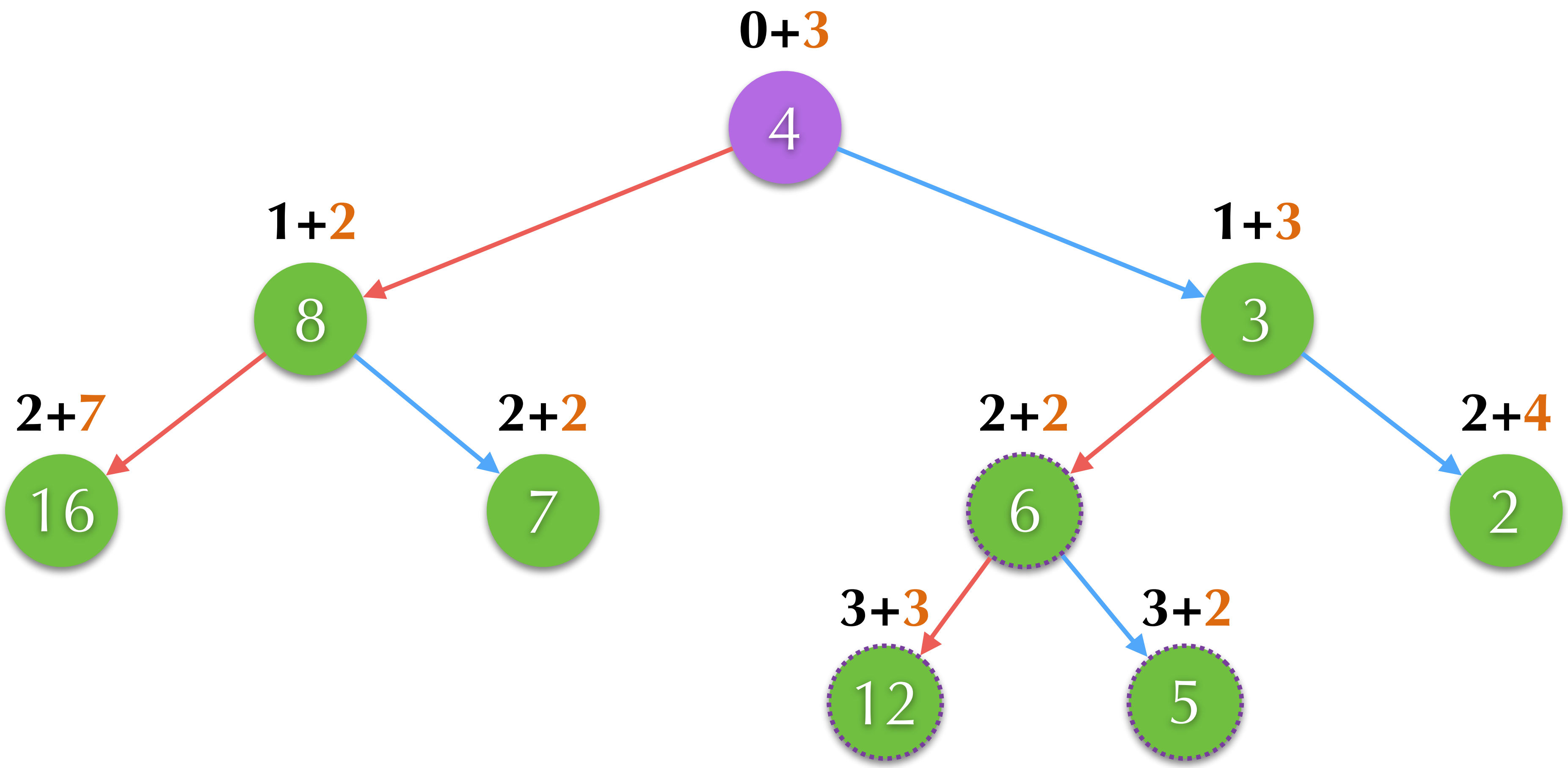
A-star



A-star

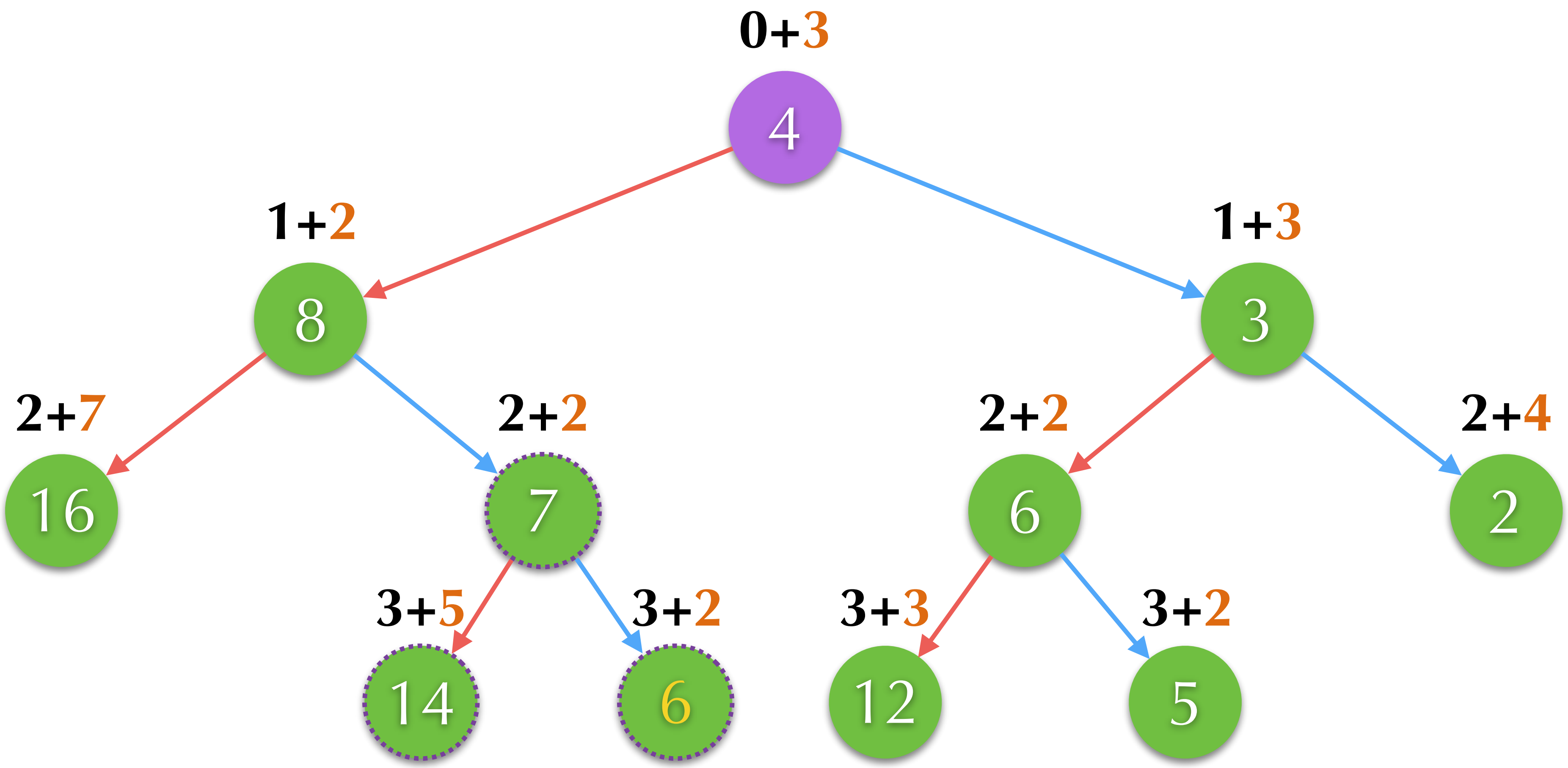


A-star

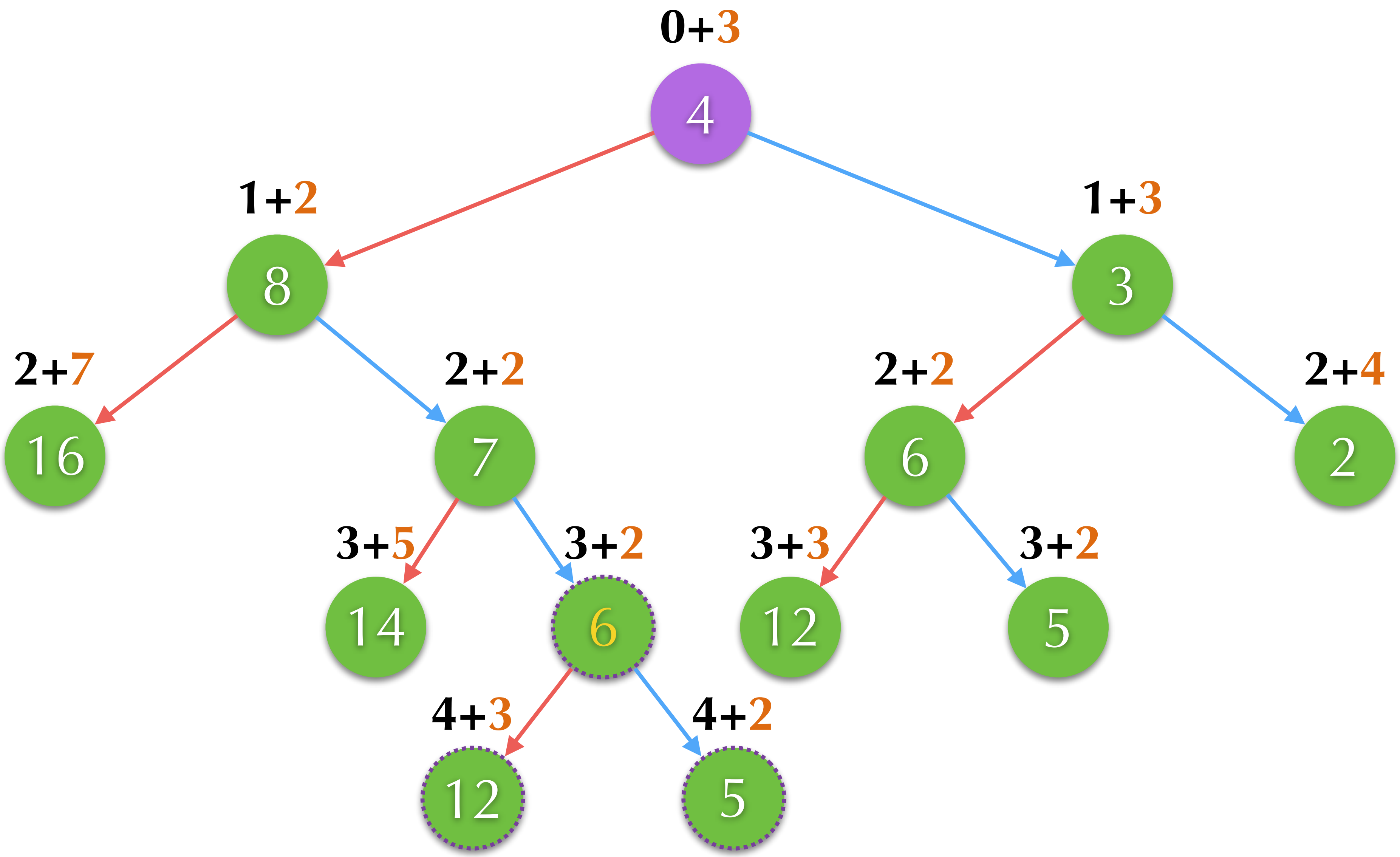


Competitive Programming

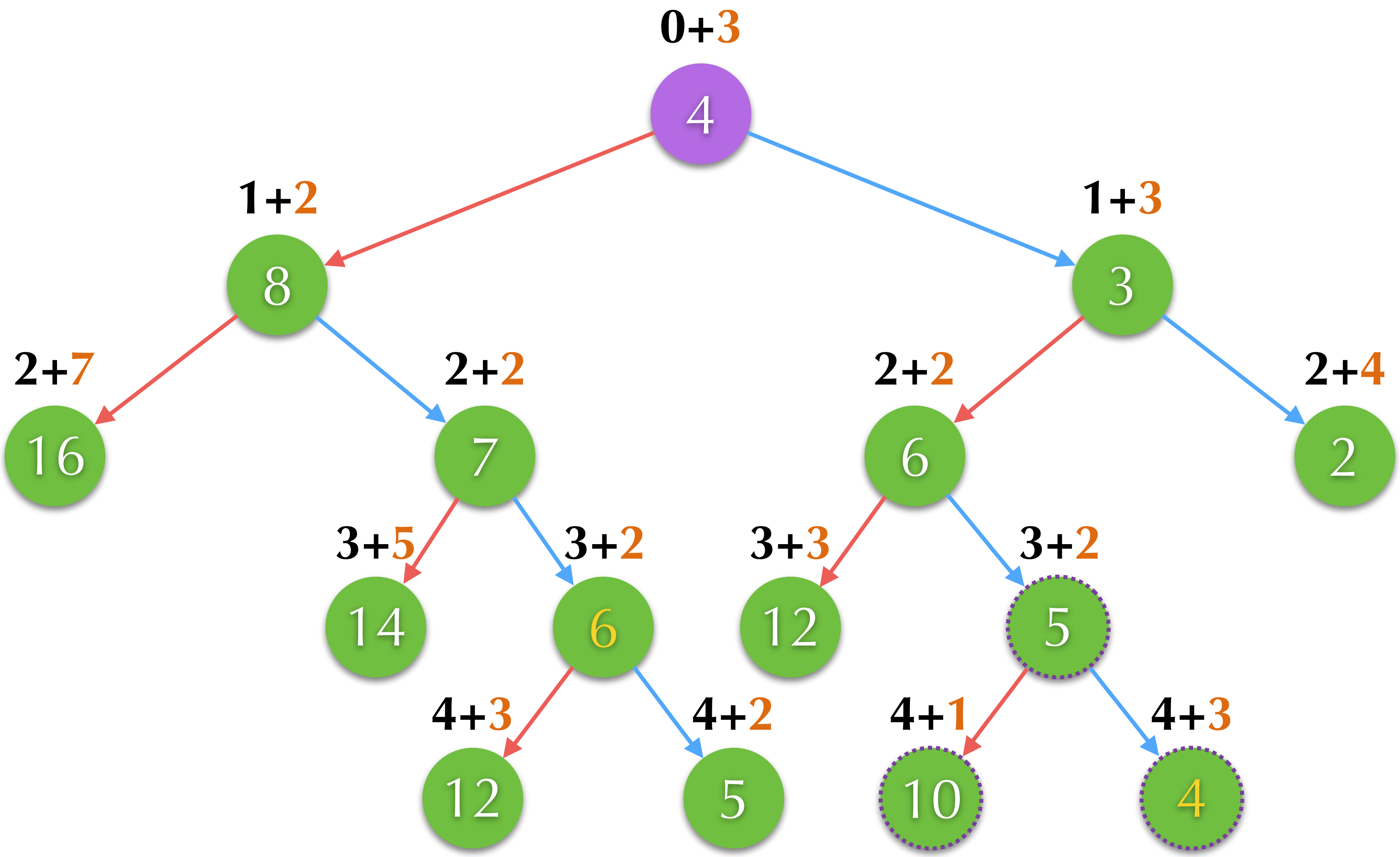
A-star



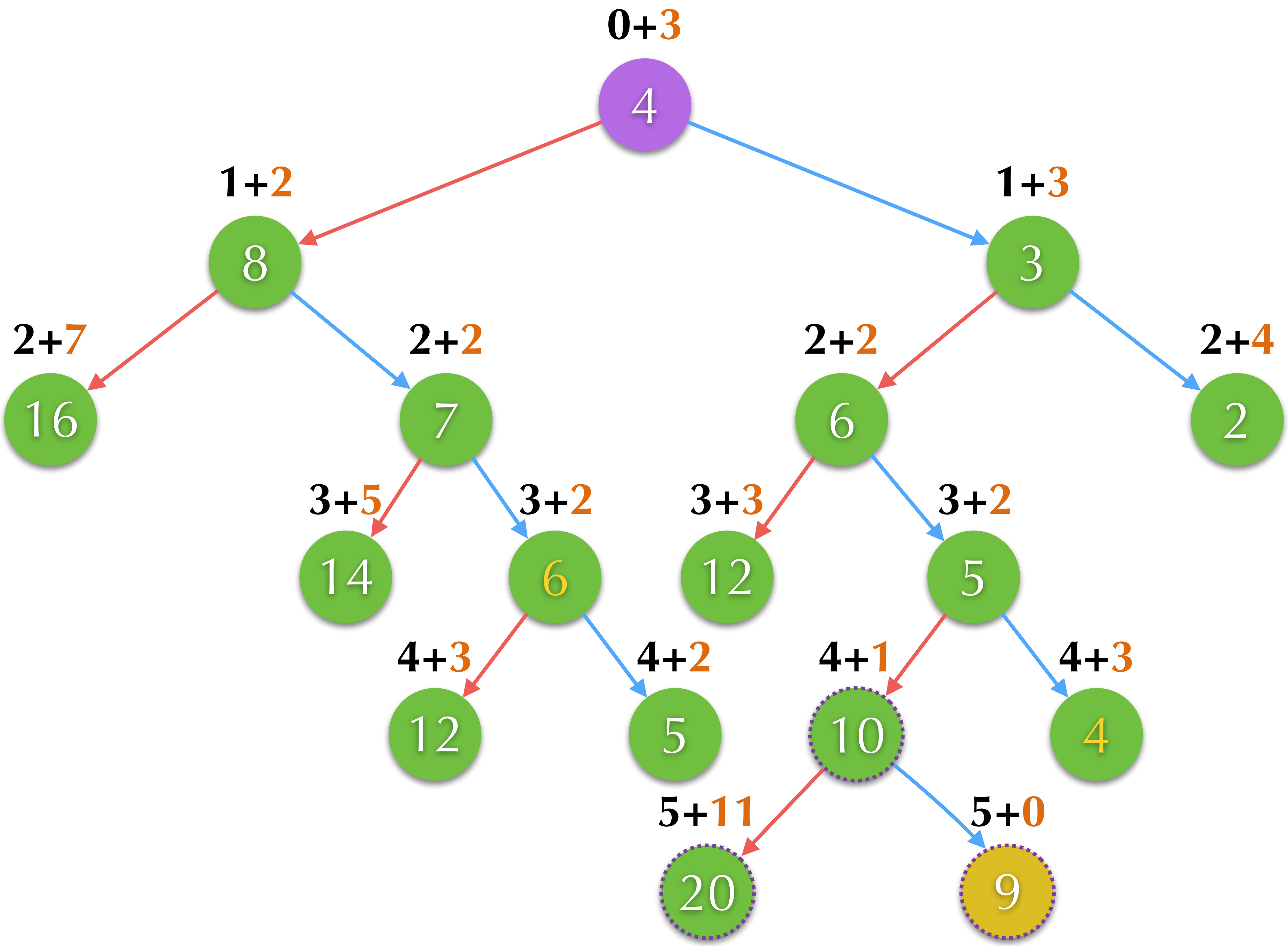
A-star



A-star


0+3
4
1+2
8
1+3
3
2+7
16
2+2
7
2+2
6
2+4
2
3+5
14
3+2
6
3+3
12
3+2
5
4+3
12
4+2
5
4+1
10
4+3
4

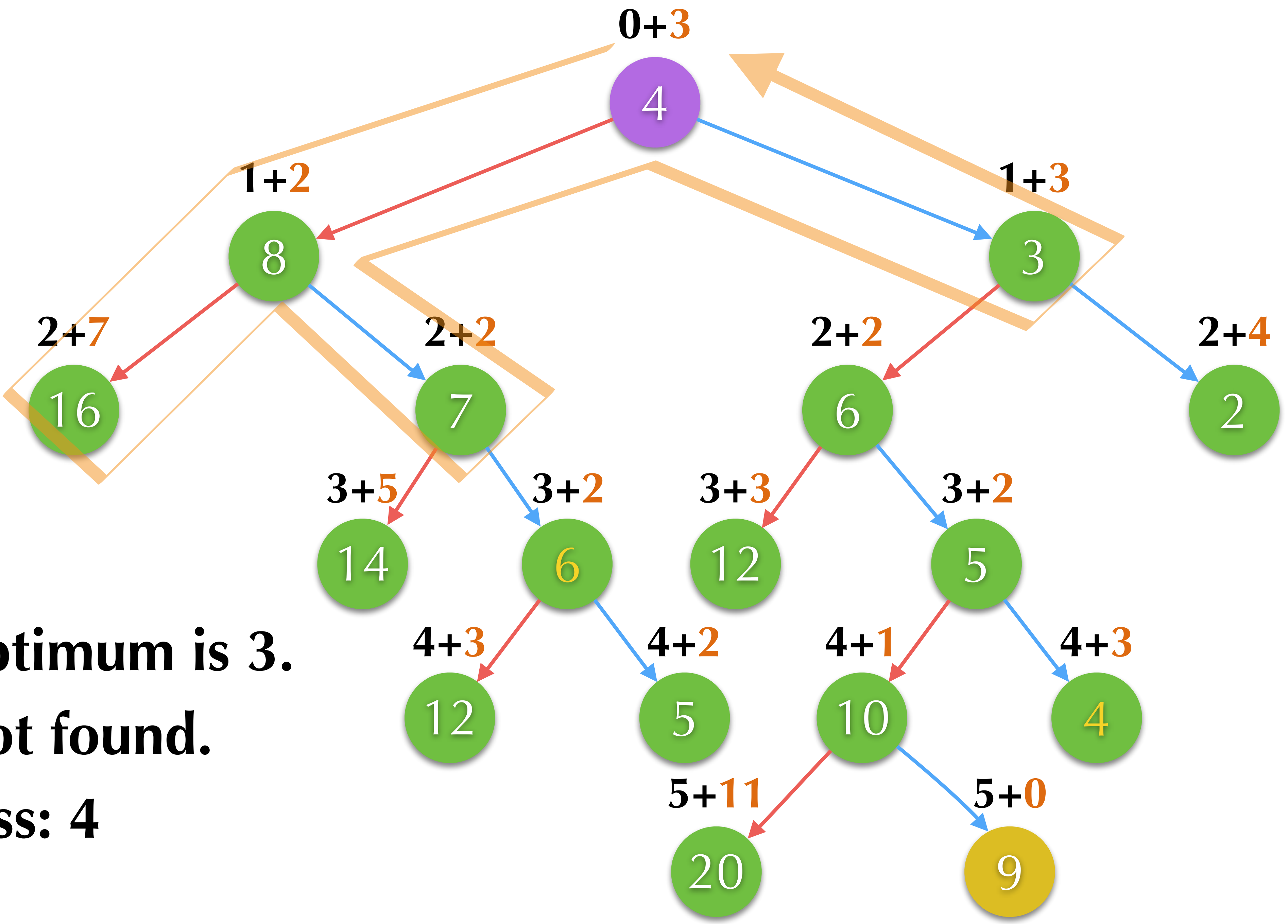
A-star


0+3
4
1+2
8
1+3
3
2+7
16
2+2
7
2+2
6
2+4
2
3+5
14
3+2
6
3+3
12
3+2
5
4+3
12
4+2
5
4+1
10
4+3
4
5+11
20
5+0
9

A-star

- Concept: best first search
    - Use priority queue
- Pros
    - Still easy to implement
    - Less memory consumption than ordinary BFS
- Cons
    - More memory consumption than DFS
        - Might be too large
    - Hard to design an admissible heuristic

IDA*

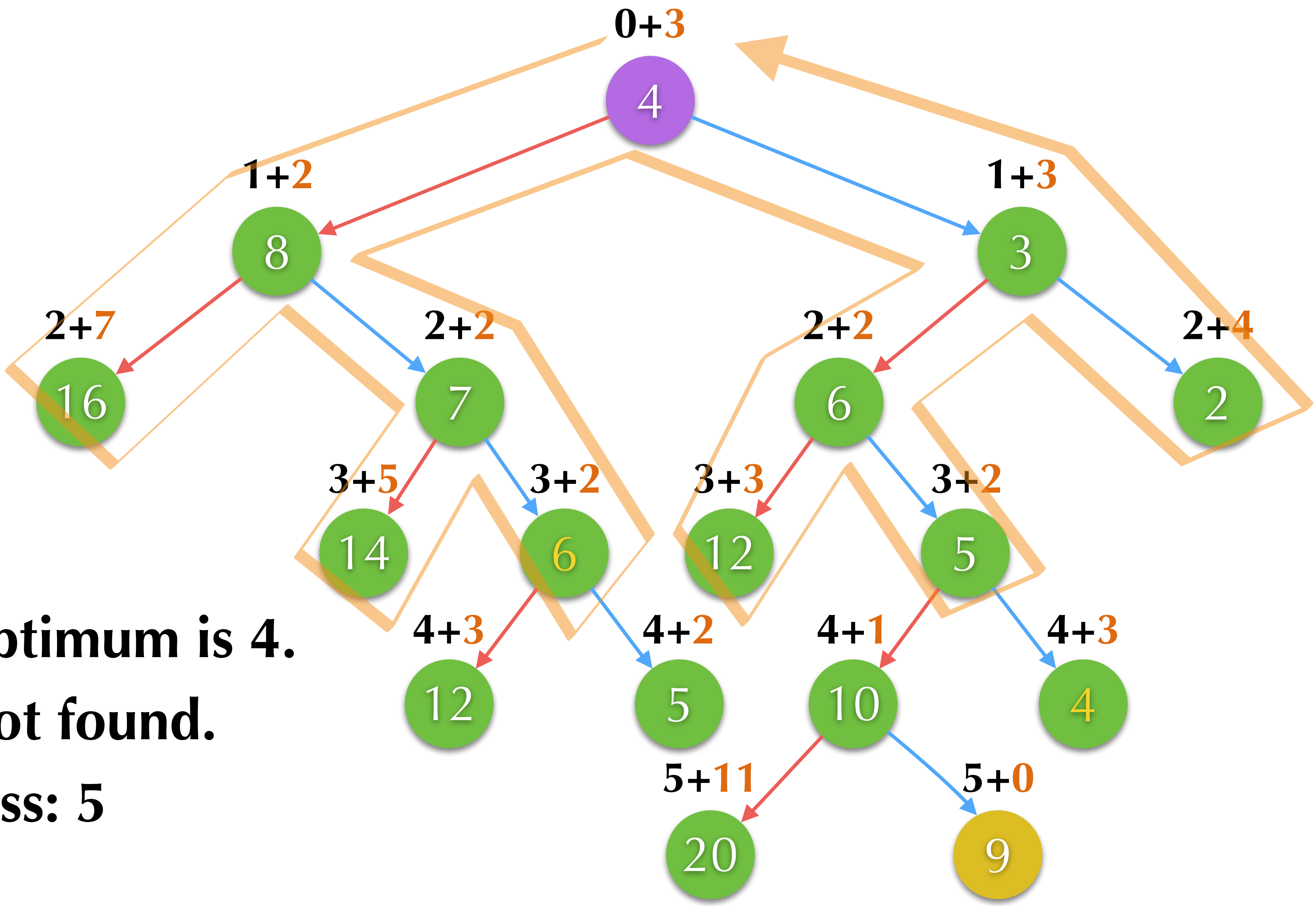


Guess: optimum is 3.

Result: not found.

Next guess: 4

IDA*

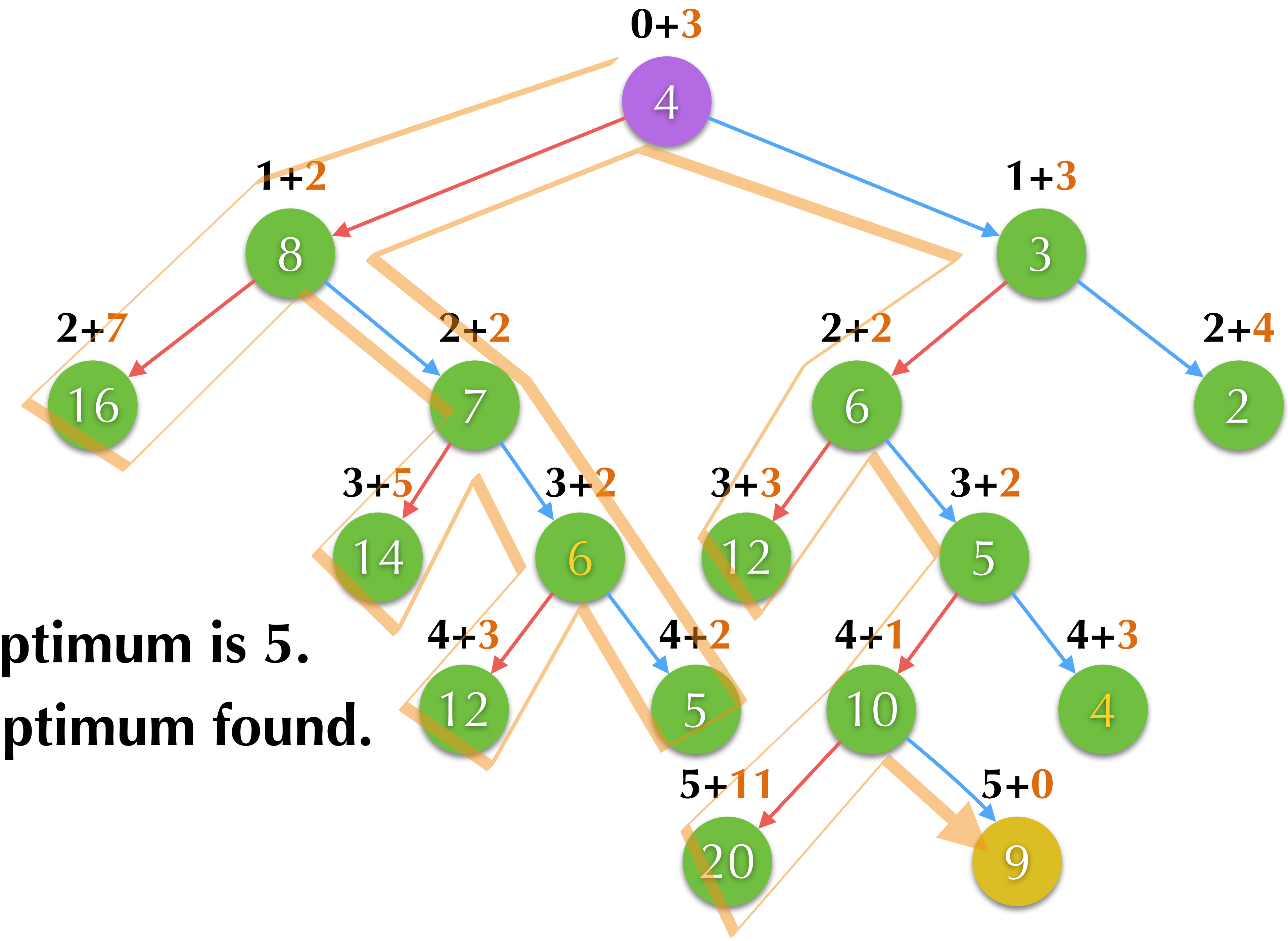


Guess: optimum is 4.

Result: not found.

Next guess: 5

IDA*



Guess: optimum is 5.

Result: optimum found.

IDA*

- Simulate best first by iterative deepening
- Pros
  - Still easy to implement (no priority queue required)
  - Memory usage can be very low (close to DFS)
- Cons
  - Might visit a state for many times
  - Hard to design an admissible heuristic
  - Can appear in your midterm

Assignment
Week 3

- UVa 10422
  - bitwise operations on long long variables
  - Meet in the middle? A*? IDA*?
- UVa 11212
  - Still bitwise operations on long long variables
  - Breadth first search is too slow?