SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence

2023/2024 Prof: Sara Bernardini

Lab 1: Rational Agents & Blind Search

Francesco Argenziano
email: argenziano@diag.uniroma1.it

*The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello, Prof. Daniele Nardi and Dott. Fabio Previtali.

Exercise 1: Agent Performance and Utility. ()

Consider both the explanations of performance measure and utility function from the lecture slides. Then, answer the following two questions.

(i) What is the difference between a performance measure and a utility function?

(ii) Describe the relation between the performance measure (Critic) and the utility function (Performance element) for a learning agent.

Exercise 1: Agent Performance and Utility. ()

Consider both the explanations of performance measure and utility function from the lecture slides. Then, answer the following two questions.

(i) What is the difference between a performance measure and a utility function?

(ii) Describe the relation between the performance measure (Critic) and the utility function (Performance element) for a learning agent.

(i) The performance measure evaluates how desirable a state of the environment is, independent of the agent, while the utility function is used by the agent to evaluate the desirability of its currently observed state.

Exercise 1: Agent Performance and Utility. ()

Consider both the explanations of performance measure and utility function from the lecture slides. Then, answer the following two questions.

(i) What is the difference between a performance measure and a utility function?

(ii) Describe the relation between the performance measure (Critic) and the utility function (Performance element) for a learning agent.

(i) The performance measure evaluates how desirable a state of the environment is, independent of the agent, while the utility function is used by the agent to evaluate the desirability of its currently observed state.

(ii) A learning agent can increase its knowledge-base over time. It usually has a performance component which uses a utility function to evaluate the current state based on the available knowledge-base and decides on the next action. To actually learn new facts it gets feedback about its actions from a performance measure, which is integrated to the agent's knowledge-base.

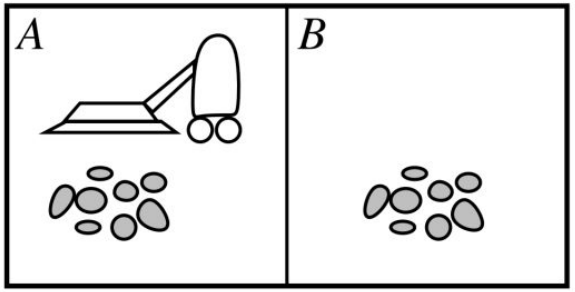
Let P be the set of possible percepts and T the lifetime of the agent (the total numbers of percepts it will receive).

1) What is the size of P for the vacuum cleaning agent?

2) How many entries does the lookup table of the vacuum cleaning agent contain after 3 time steps?
a) if we store **only the percepts that were perceived**?
b) if we store **all possible percepts**?

3) How many entries will the lookup table contain if we store all possible percepts, i.e. all possible percept sequences for the whole life time of the agent?

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 1: Ex 2. Vacuum cleaning agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Figure 2: Ex.2 Lookup table of vacuum cleaner agent

Let P be the set of possible percepts and T the lifetime of the agent (the total numbers of percepts it will receive).

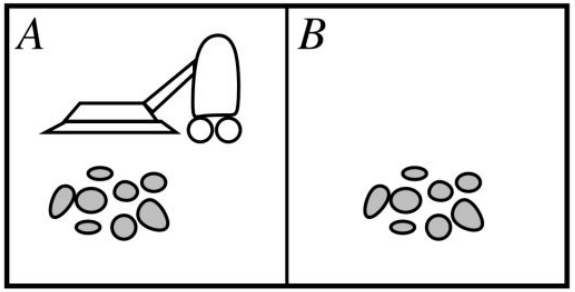
1) What is the size of P for the vacuum cleaning agent?

2) How many entries does the lookup table of the vacuum cleaning agent contain after 3 time steps?
a) if we store **only the percepts that were perceived**?
b) if we store **all possible percepts**?

3) How many entries will the lookup table contain if we store all possible percepts, i.e. all possible percept sequences for the whole life time of the agent?

1) 4

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 1: Ex 2. Vacuum cleaning agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Figure 2: Ex.2 Lookup table of vacuum cleaner agent

Let P be the set of possible percepts and T the lifetime of the agent (the total numbers of percepts it will receive).

1) What is the size of P for the vacuum cleaning agent?

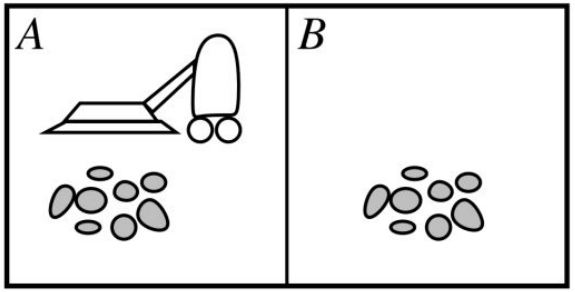
2) How many entries does the lookup table of the vacuum cleaning agent contain after 3 time steps?
a) if we store **only the percepts that were perceived**?
b) if we store **all possible percepts**?

3) How many entries will the lookup table contain if we store all possible percepts, i.e. all possible percept sequences for the whole life time of the agent?

1) 4

2a) 3

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 1: Ex 2. Vacuum cleaning agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Figure 2: Ex.2 Lookup table of vacuum cleaner agent

Let P be the set of possible percepts and T the lifetime of the agent (the total numbers of percepts it will receive).

1) What is the size of P for the vacuum cleaning agent?

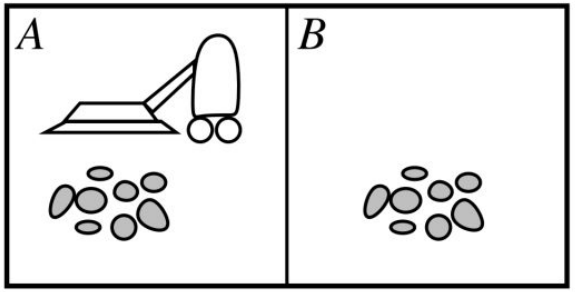
2) How many entries does the lookup table of the vacuum cleaning agent contain after 3 time steps?
a) if we store **only the percepts that were perceived**?
b) if we store **all possible percepts**?

3) How many entries will the lookup table contain if we store all possible percepts, i.e. all possible percept sequences for the whole life time of the agent?

1) 4

2a) 3
2b) 84

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 1: Ex 2. Vacuum cleaning agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Figure 2: Ex.2 Lookup table of vacuum cleaner agent

Let P be the set of possible percepts and T the lifetime of the agent (the total numbers of percepts it will receive).

1) What is the size of P for the vacuum cleaning agent?

2) How many entries does the lookup table of the vacuum cleaning agent contain after 3 time steps?
a) if we store **only the percepts that were perceived**?
b) if we store **all possible percepts**?

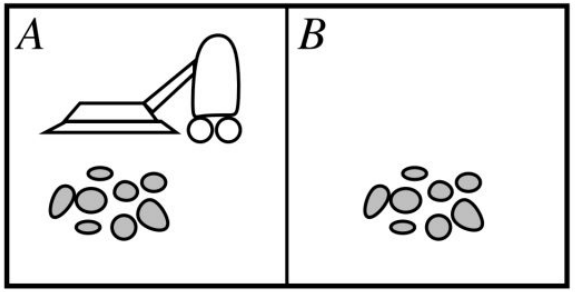
3) How many entries will the lookup table contain if we store all possible percepts, i.e. all possible percept sequences for the whole life time of the agent?

1) 4

2a) 3
2b) 84

3) $\sum_{t=1}^{T} 4^t$

Vacuum-cleaner world



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

Figure 1: Ex 2. Vacuum cleaning agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
⋮	⋮

Figure 2: Ex.2 Lookup table of vacuum cleaner agent

Exercise 3: Agent Rationality. ()

Again consider the example of the vacuum cleaner agent with the following specifications:

- The "geography" is known a priori, but not the distribution of dirt and the location of the agent.
- Clean squares stay clean, sucking removes dirt.
- Left and right moves move the agent to the other location, unless they would take the agent outside the room. In this latter situation, the agent stays where it is.
- The only available actions are LEFT, RIGHT, SUCK.
- The agent correctly perceives its location and whether there is dirt.
- The performance measure awards 1 point for each new clean square at each time step.

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

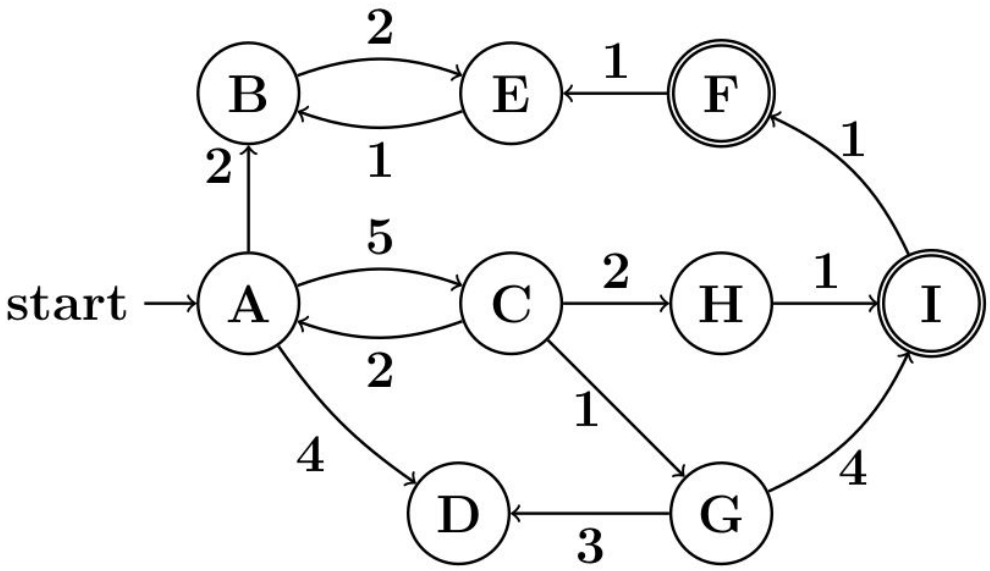
Figure 3: Action function of vacuum cleaner agent

Given the information above, is the simple vacuum cleaner funtion in Figure 3 rational or not?

Consider the state space depicted here, where A is the initial state, and F and I are goal states. The transitions are annotated by their costs. List all the states that are:

1) solvable
2) dead-ends
3) not reachable from G
4) reachable from H

Assume every node can reach itself

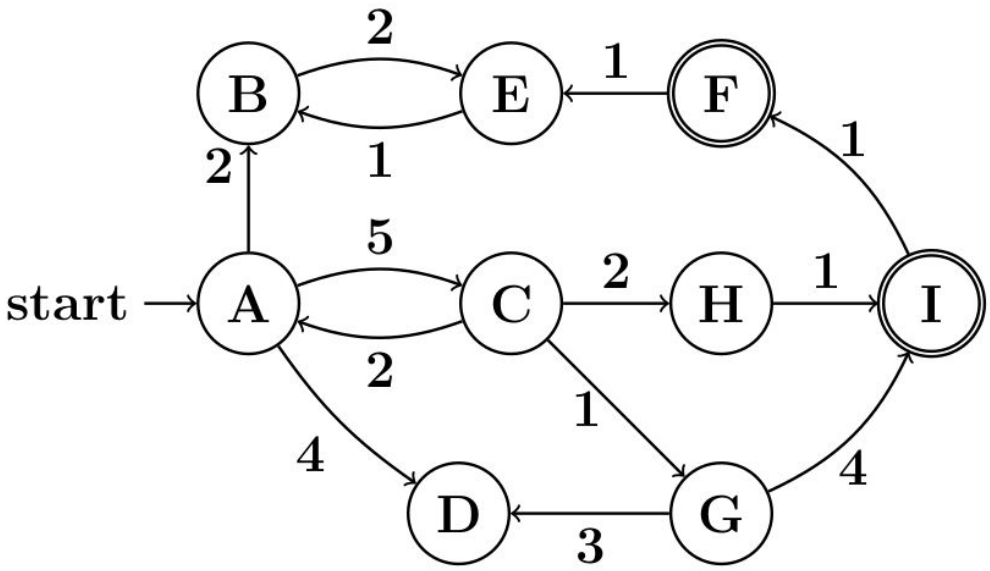


Consider the state space depicted here, where A is the initial state, and F and I are goal states. The transitions are annotated by their costs. List all the states that are:

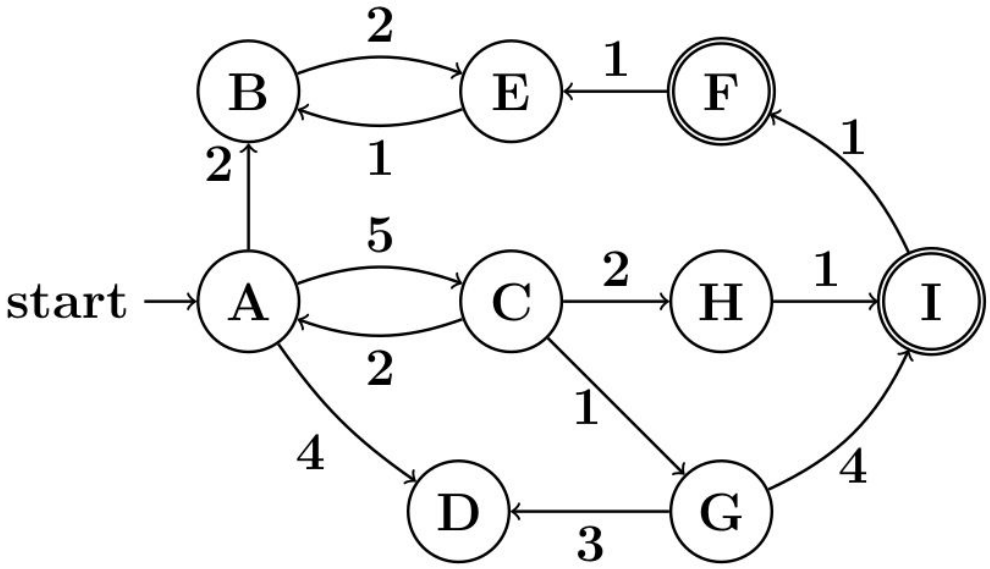
1) solvable
2) dead-ends
3) not reachable from G
4) reachable from H

Assume every node can reach itself

1) A, C, F, G, H, I



Consider the state space depicted here, where A is the initial state, and F and I are goal states. The transitions are annotated by their costs. List all the states that are:

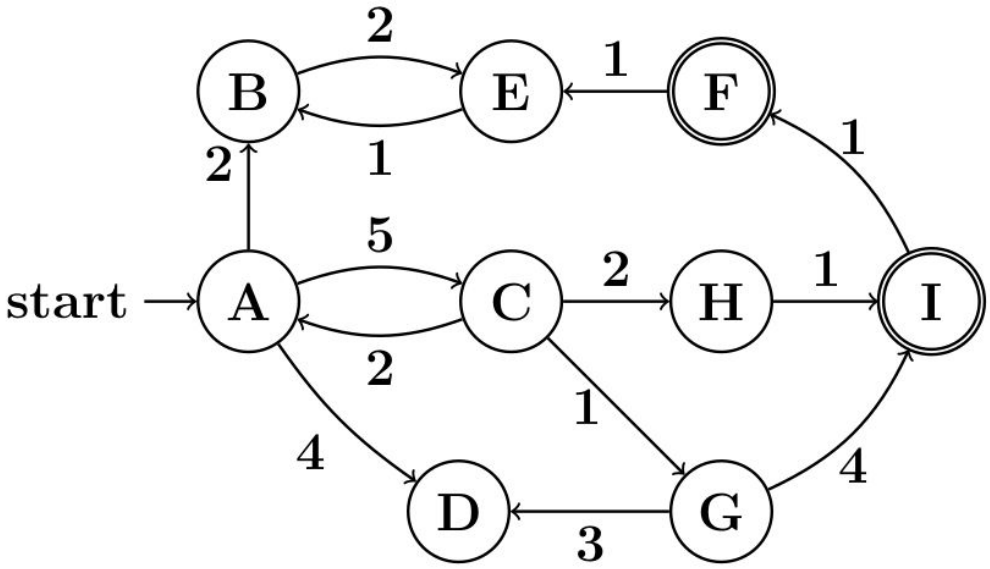


1) solvable
2) dead-ends
3) not reachable from G
4) reachable from H

Assume every node can reach itself

1) A, C, F, G, H, I
2) B, D, E

Consider the state space depicted here, where A is the initial state, and F and I are goal states. The transitions are annotated by their costs. List all the states that are:

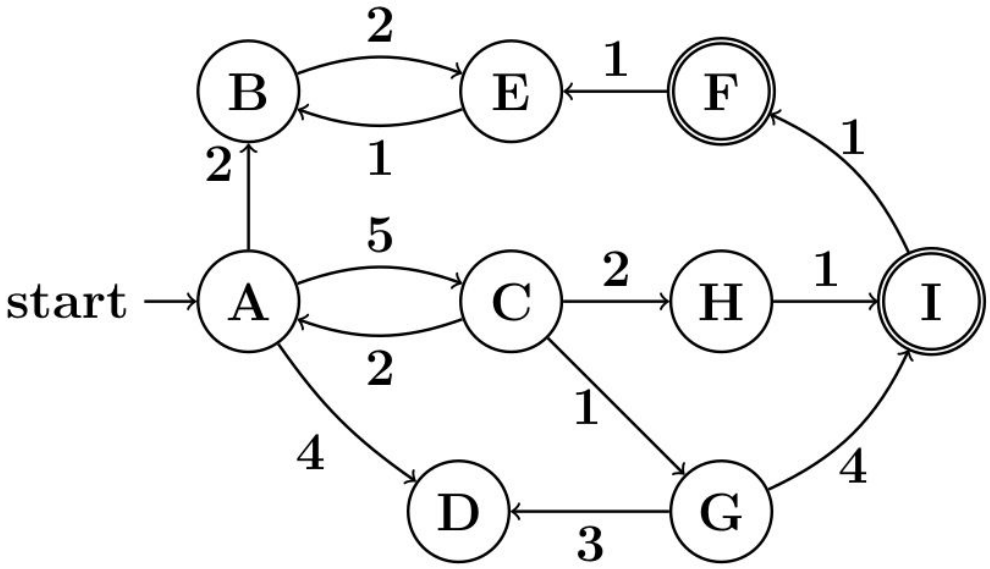


1) solvable
2) dead-ends
3) not reachable from G
4) reachable from H

Assume every node can reach itself

1) A, C, F, G, H, I
2) B, D, E
3) A, C, H

Consider the state space depicted here, where A is the initial state, and F and I are goal states. The transitions are annotated by their costs. List all the states that are:



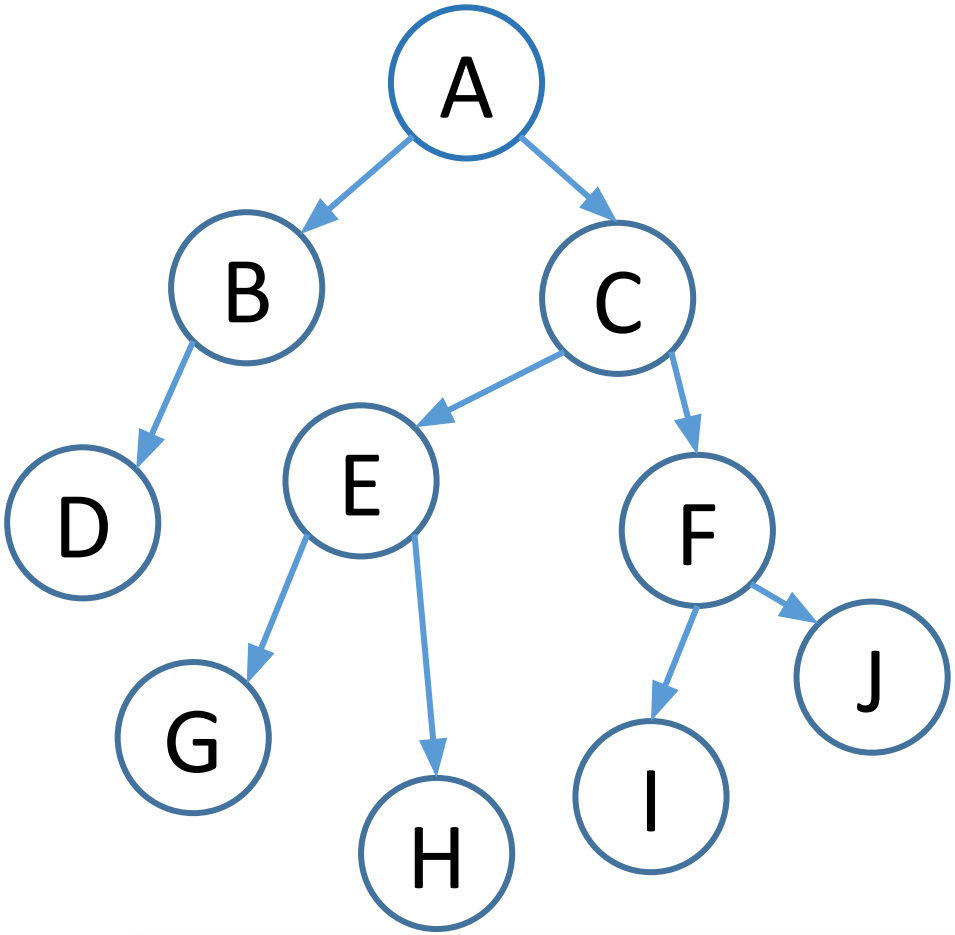
1) solvable
2) dead-ends
3) not reachable from G
4) reachable from H

Assume every node can reach itself

1) A, C, F, G, H, I
2) B, D, E
3) A, C, H
4) B, E, F, H, I

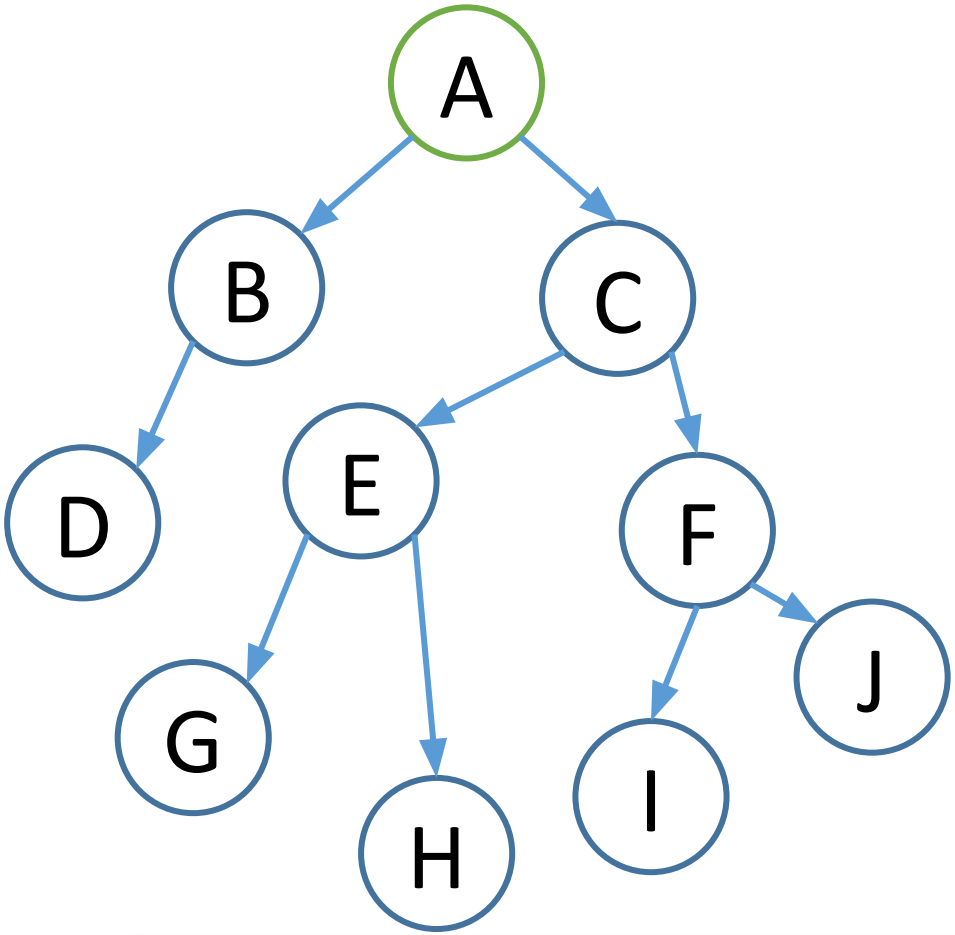
Breadth-first, Depth-first

Breadth-first search:
visit: {}
expansion: {}



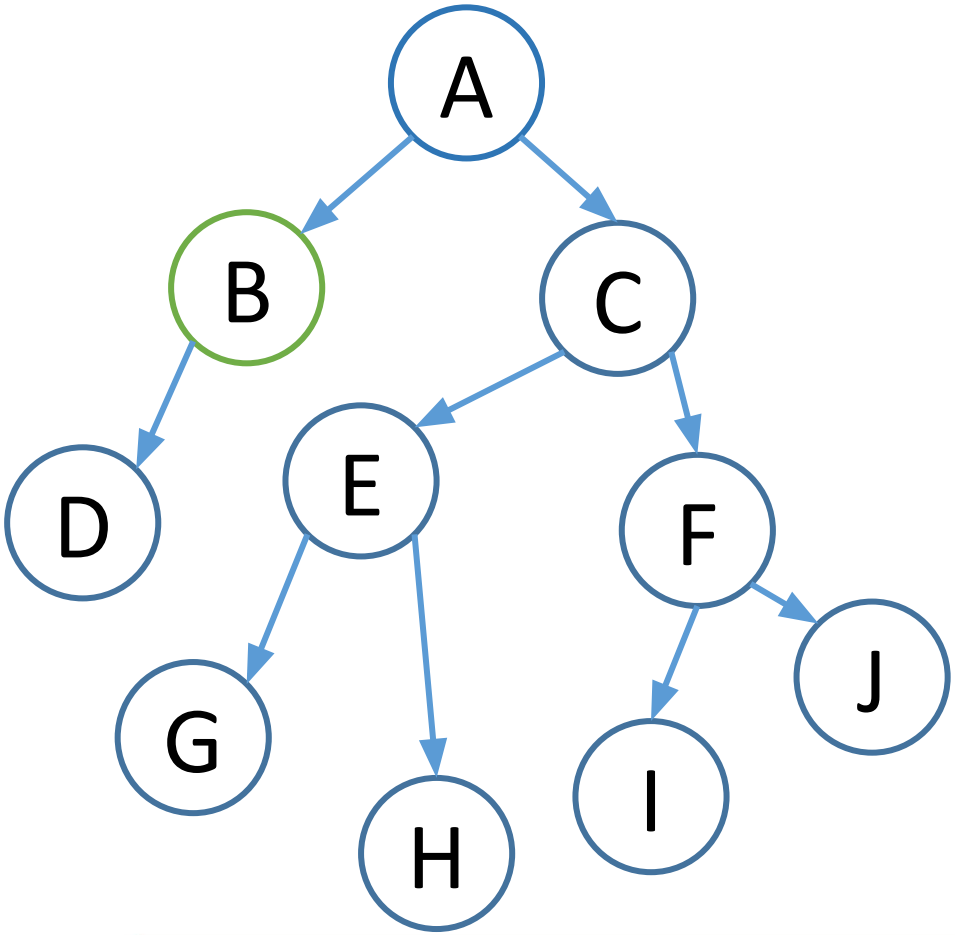
Breadth-first, Depth-first

Breadth-first search:
visit: {A}
expansion: {A}



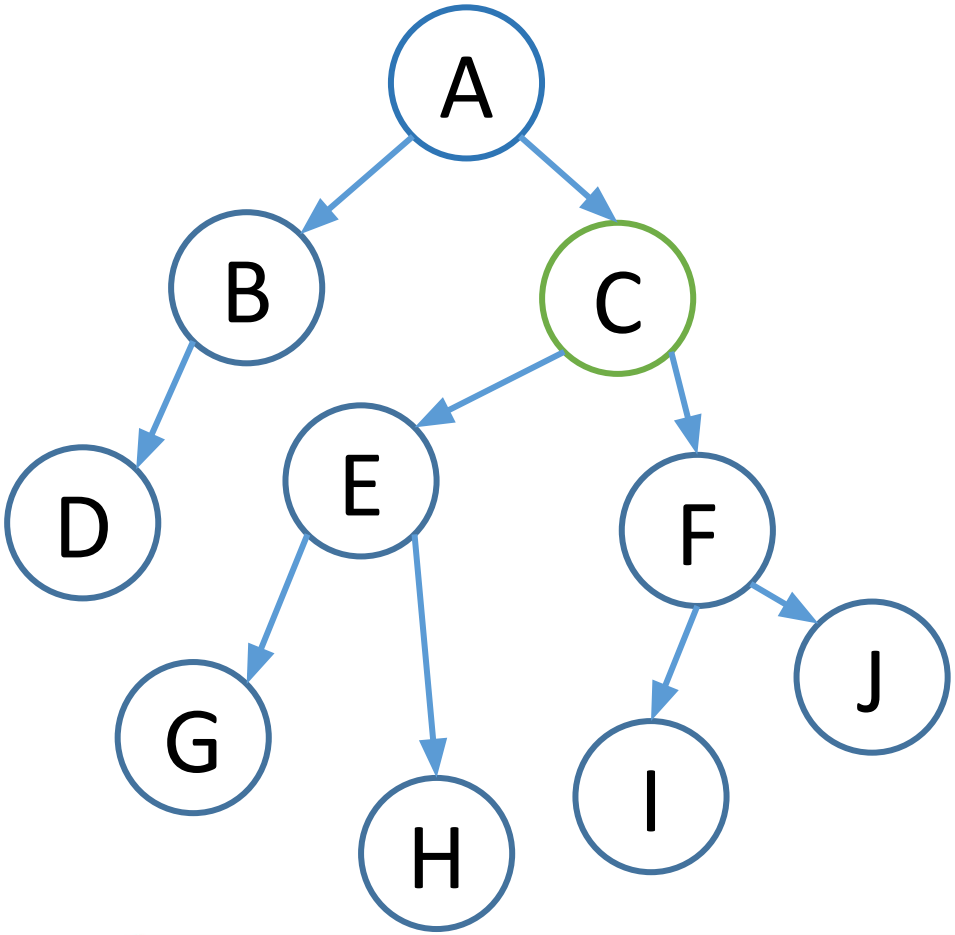
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B}
expansion: {A,B}



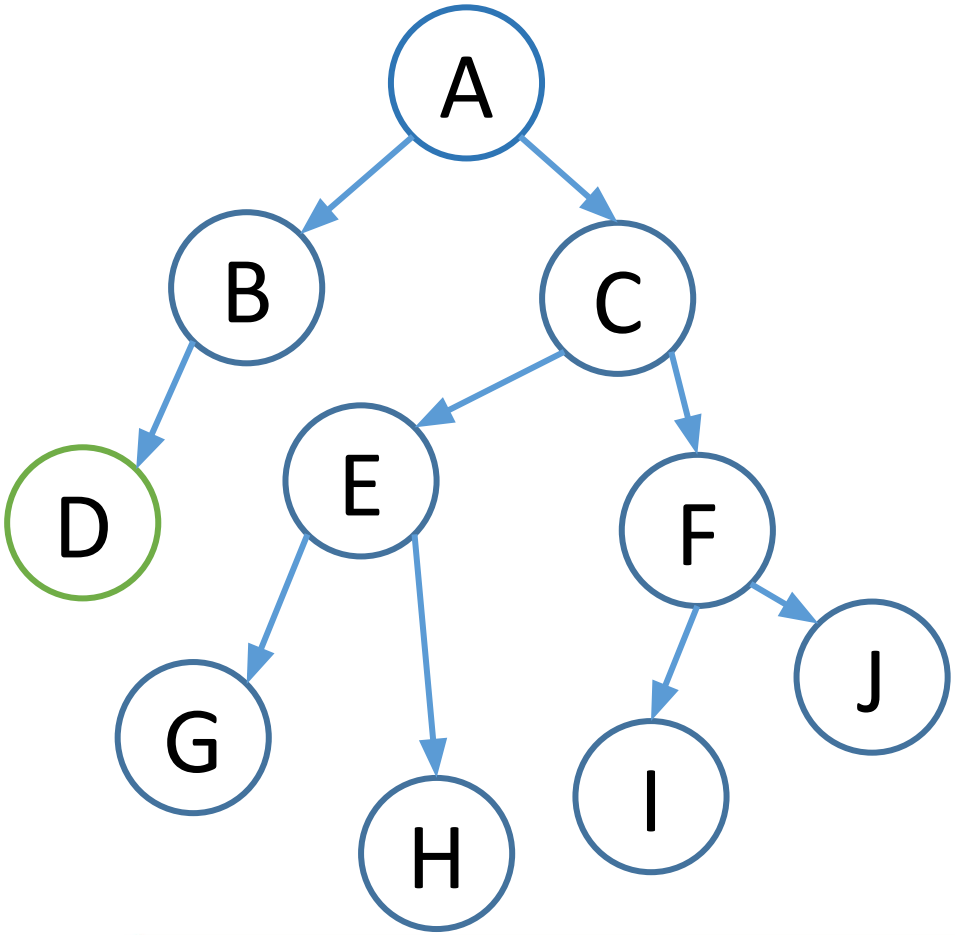
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C}
expansion: {A,B,C}



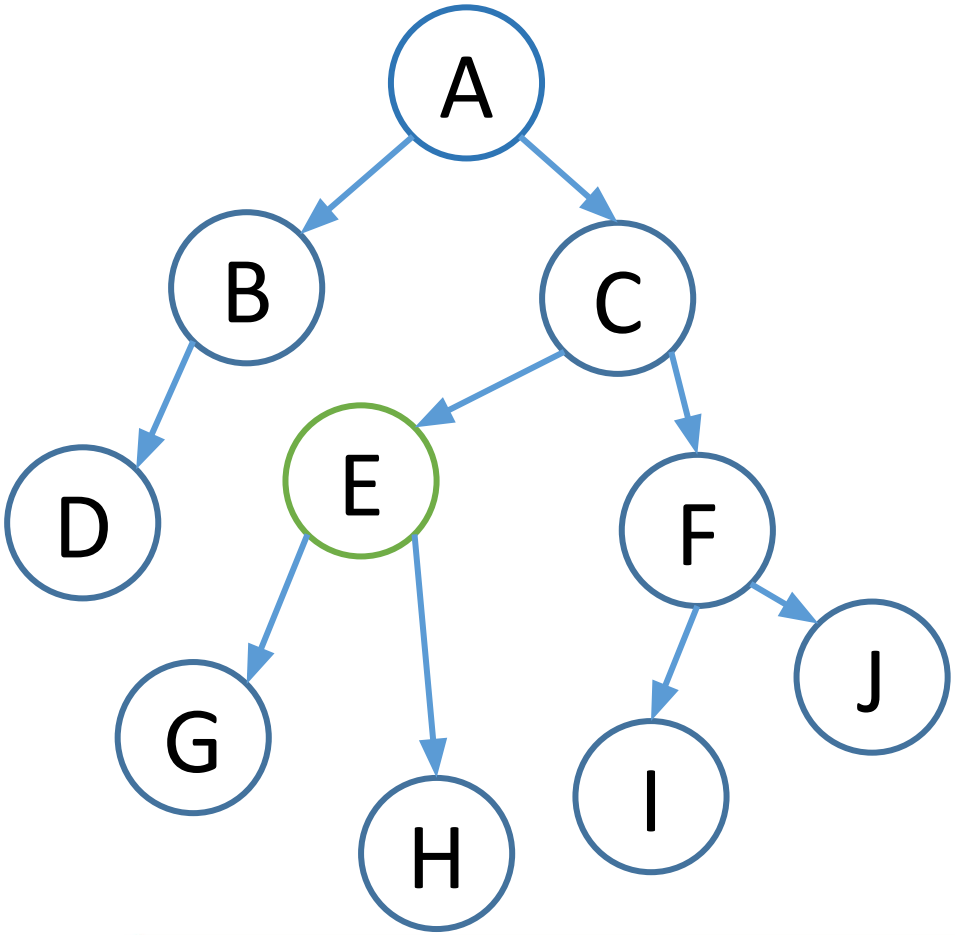
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D}
expansion: {A,B,C}



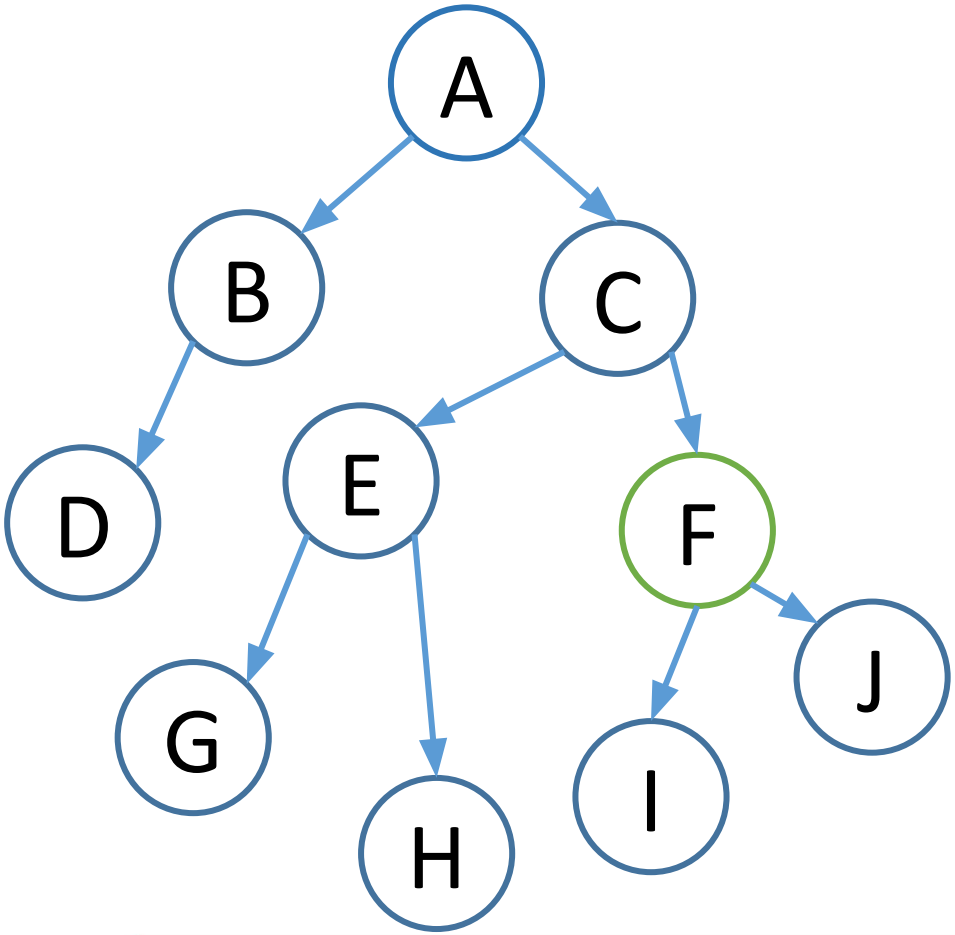
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E}
expansion: {A,B,C,E}



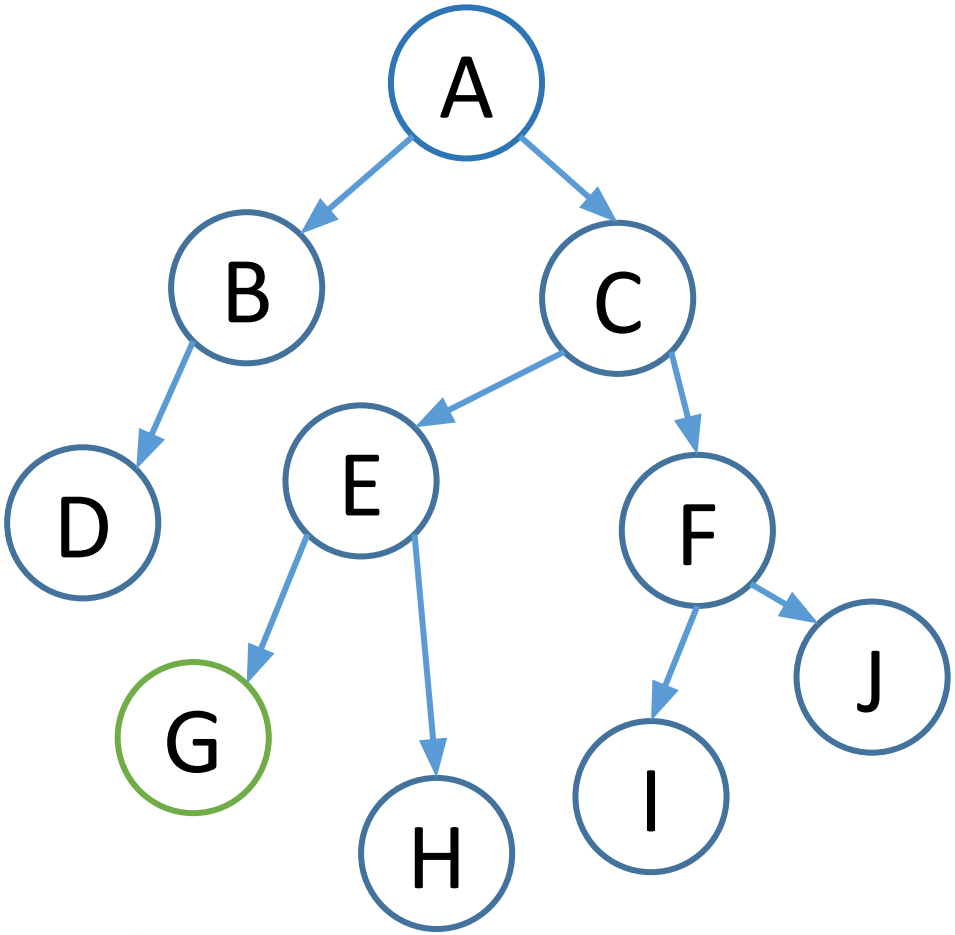
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E,F}
expansion: {A,B,C,E,F}



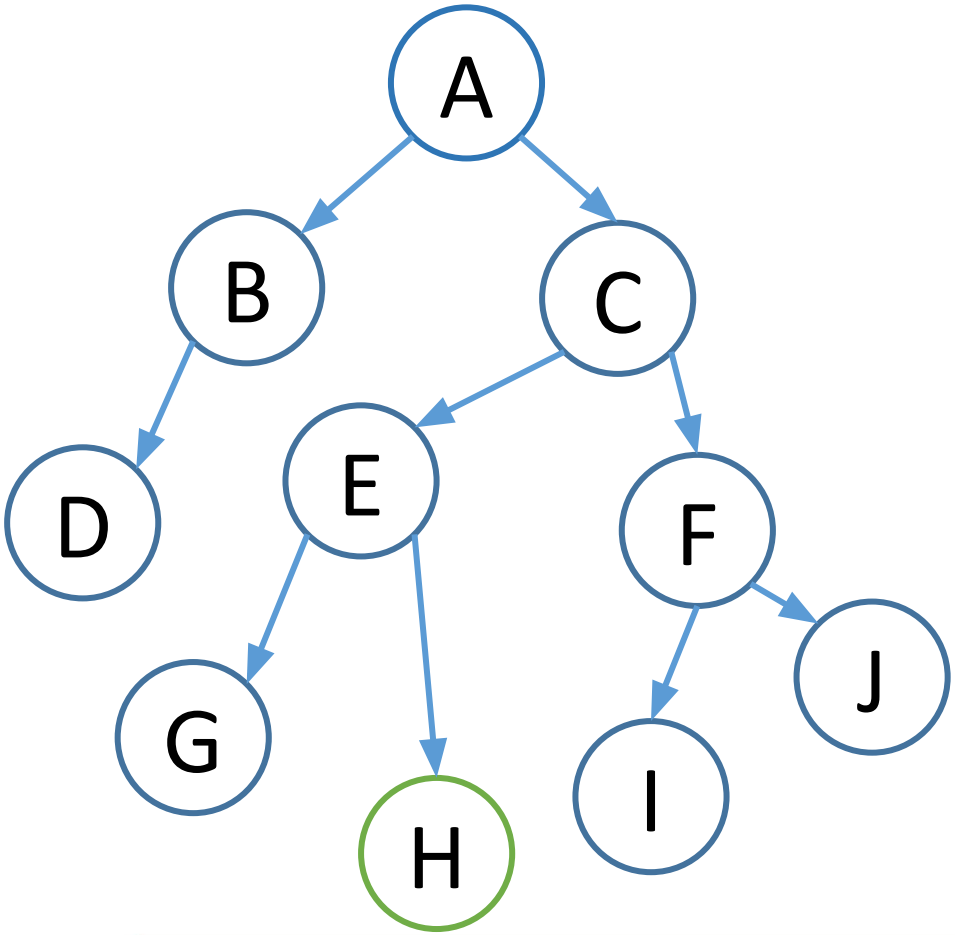
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E,F,G}
expansion: {A,B,C,E,F}



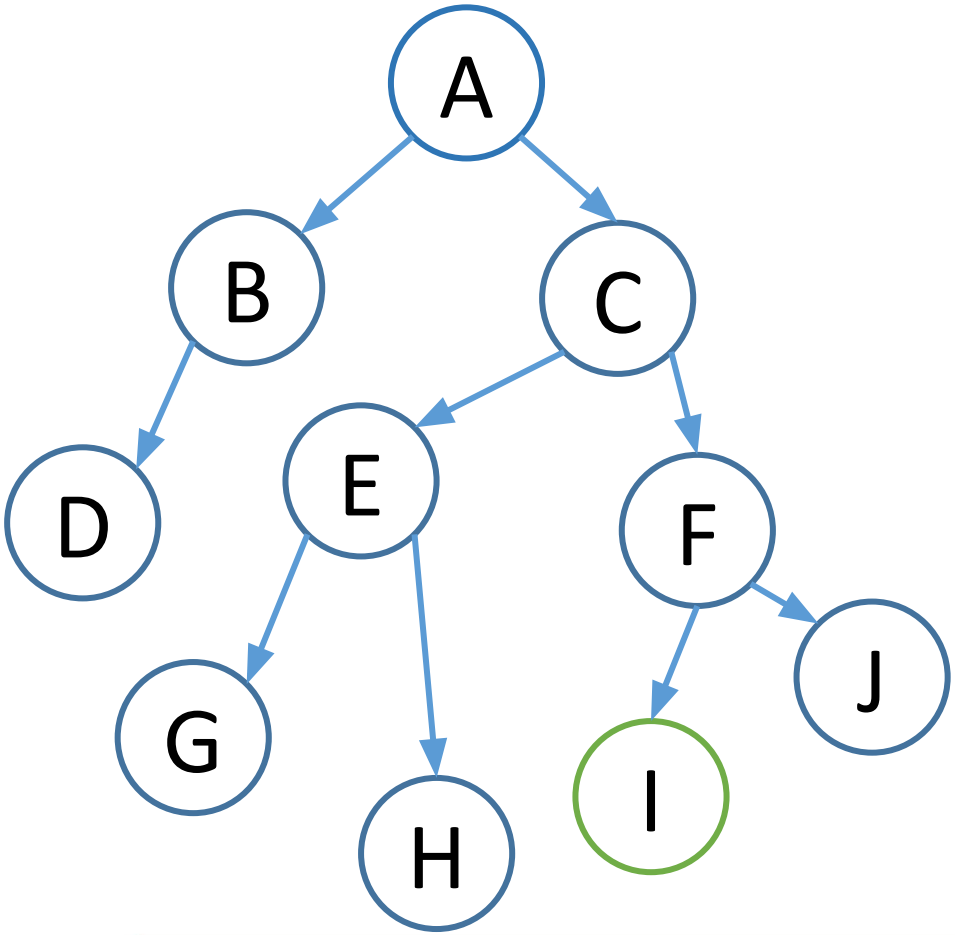
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E,F,G,H}
expansion: {A,B,C,E,F}



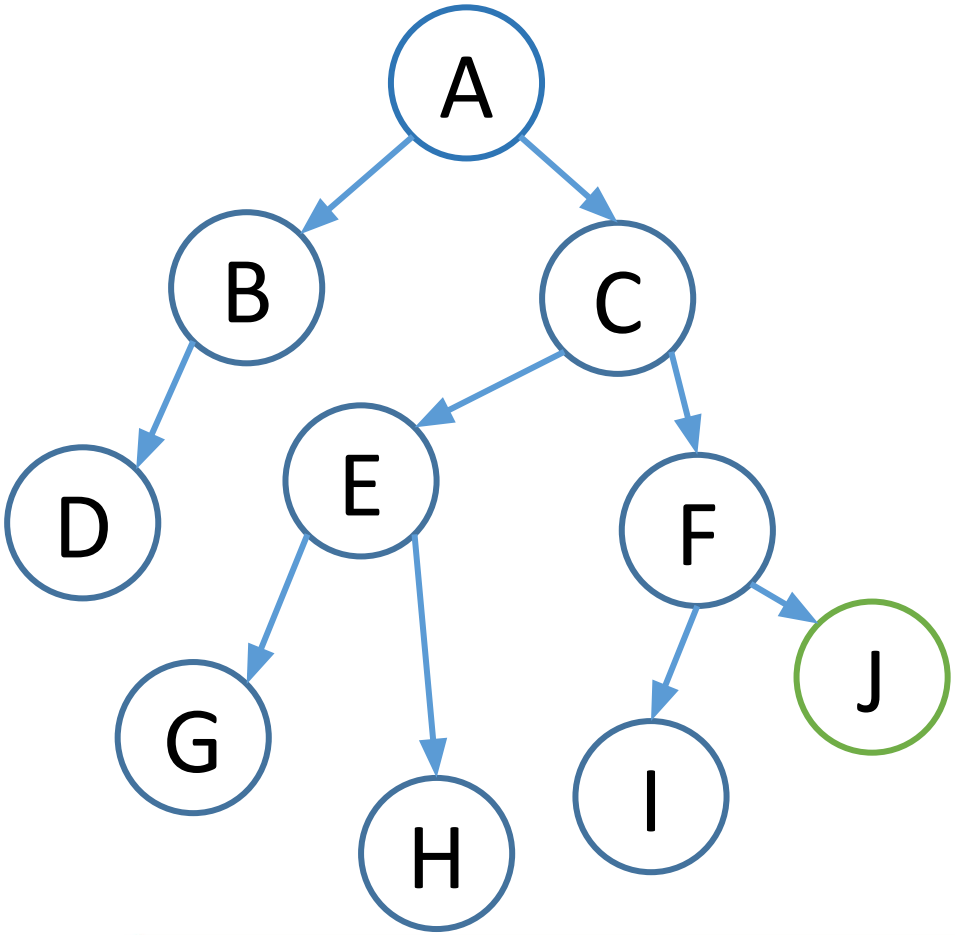
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E,F,G,H,I}
expansion: {A,B,C,E,F}



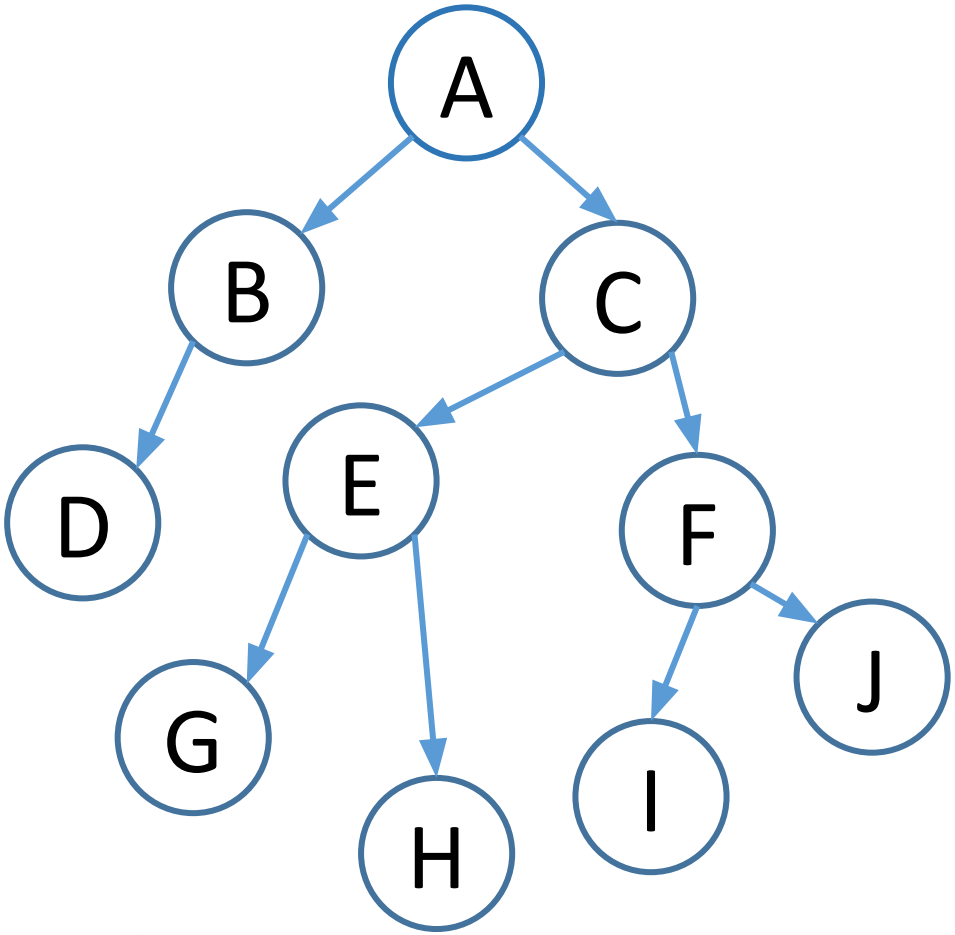
Breadth-first, Depth-first

Breadth-first search:
visit: {A,B,C,D,E,F,G,H,I,J}
expansion: {A,B,C,E,F}



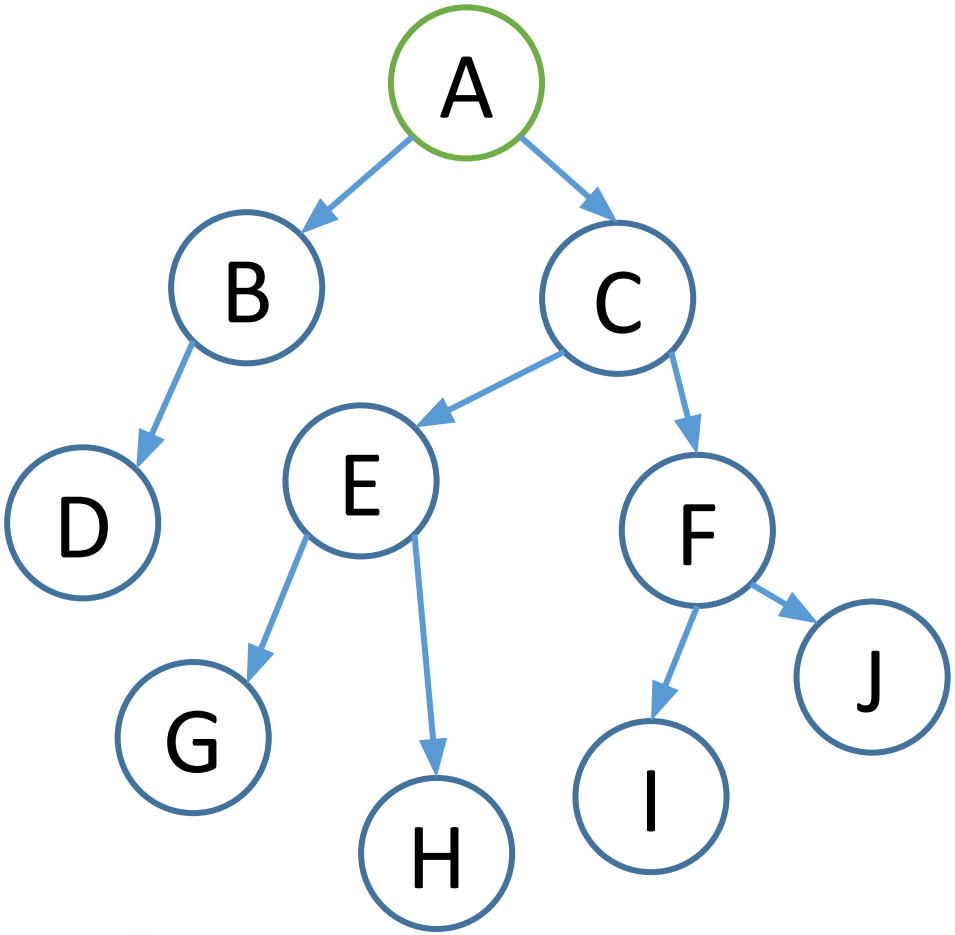
Breadth-first, Depth-first

Depth-first search:
expansion: {}
visit: {}



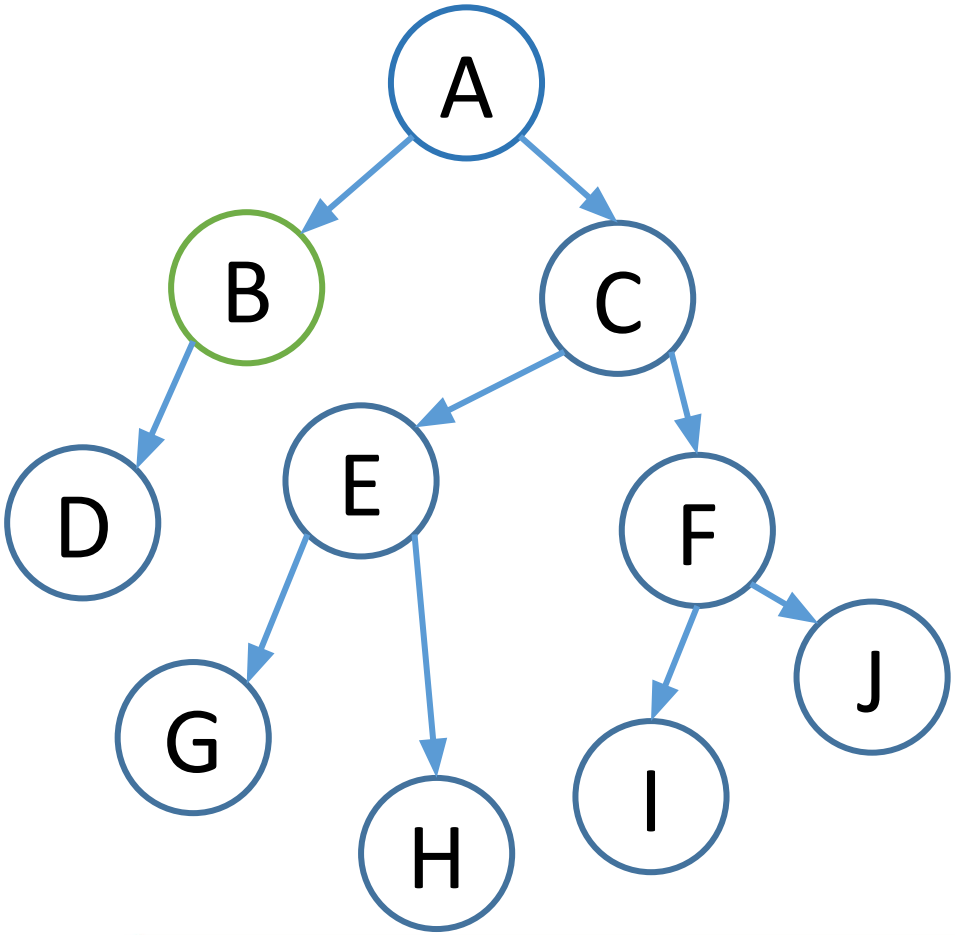
Breadth-first, Depth-first

Depth-first search:
visit: {A}
expansion: {A}



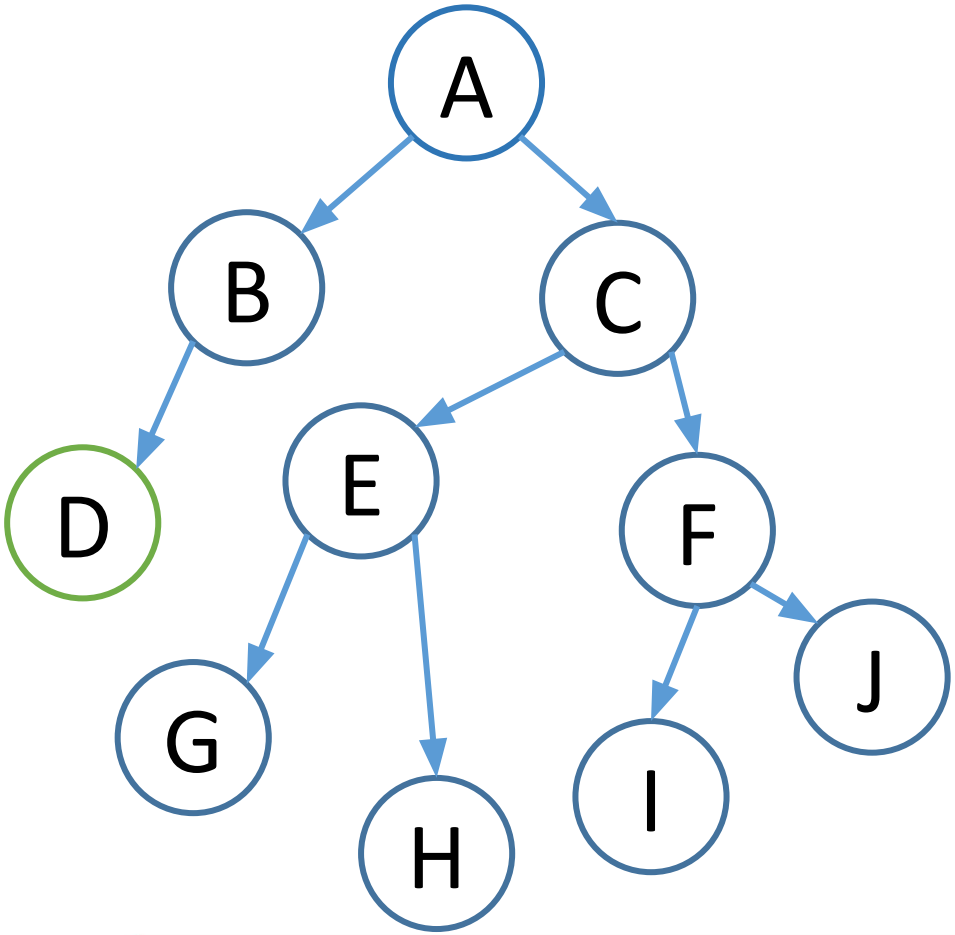
Breadth-first, Depth-first

Depth-first search:
visit: {A,B}
expansion: {A,B}



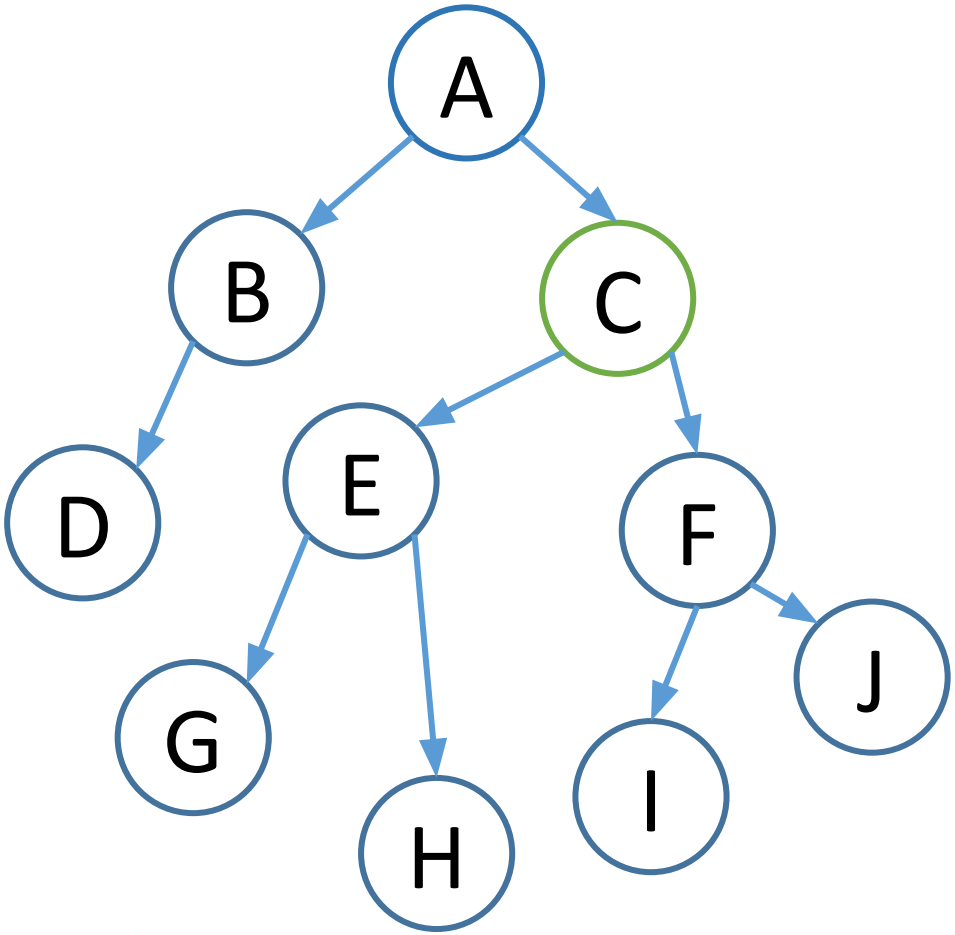
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D}
expansion: {A,B}



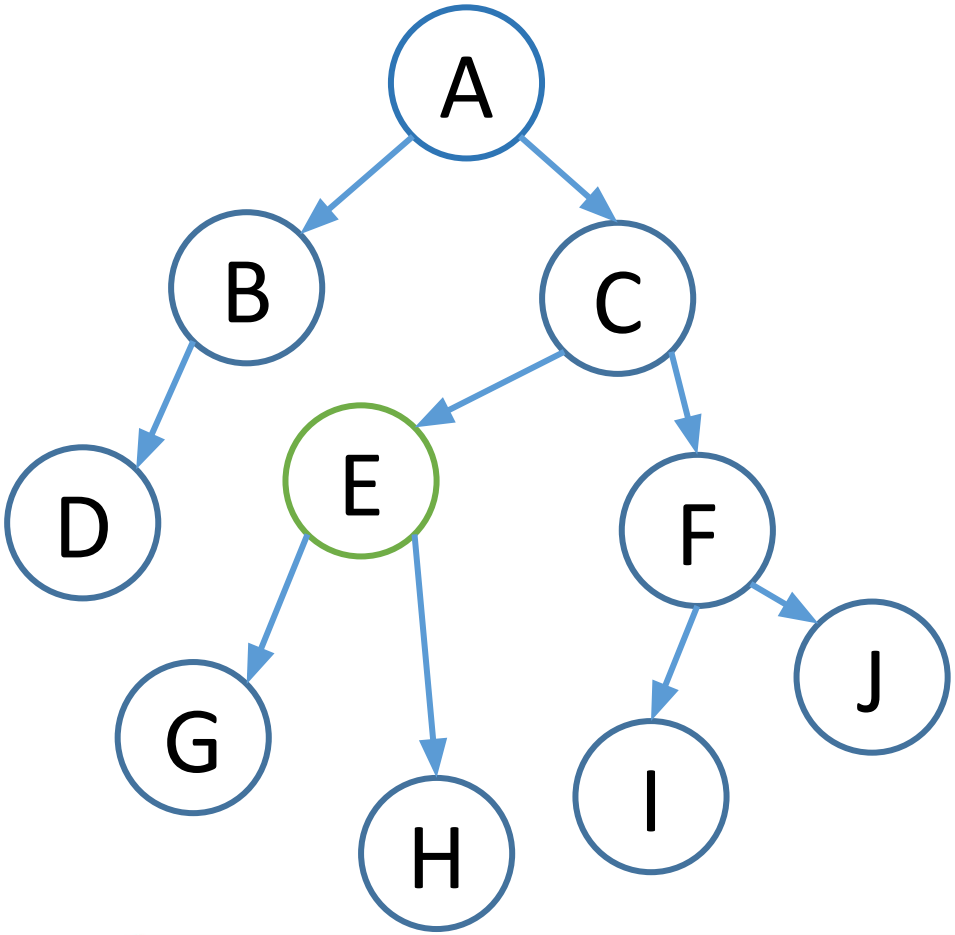
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C}
expansion: {A,B,C}



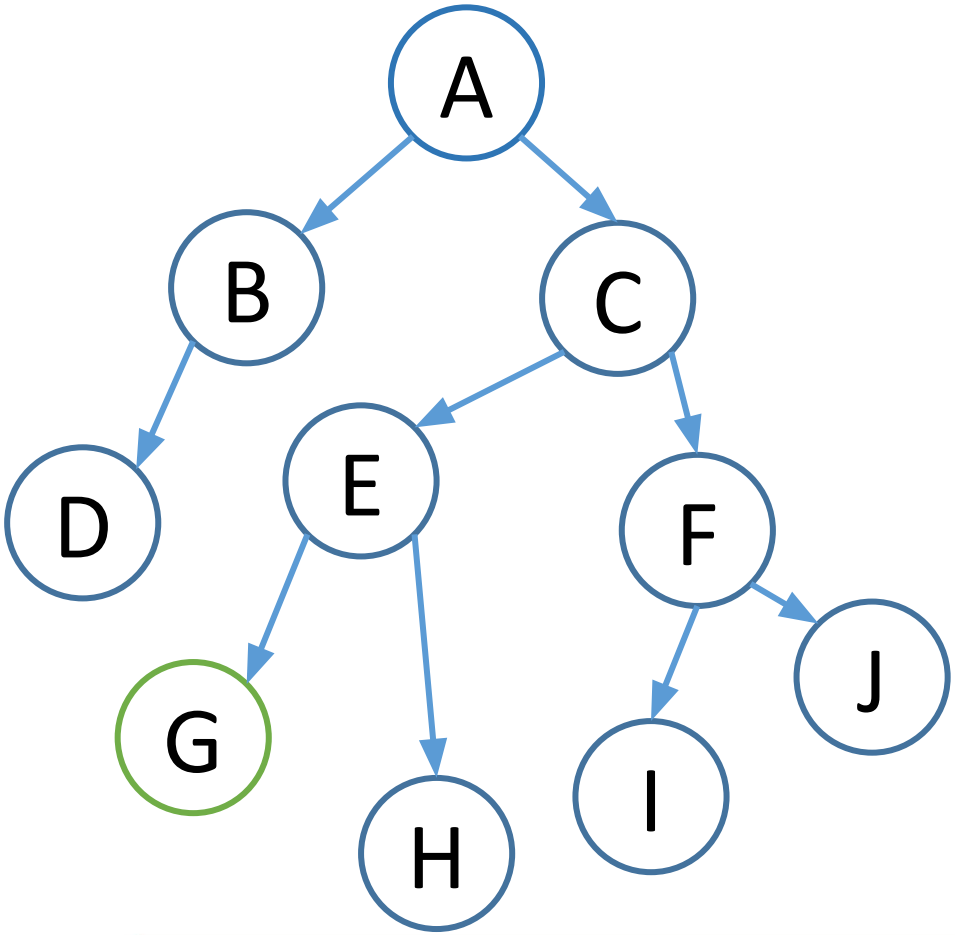
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E}
expansion: {A,B,C,E}



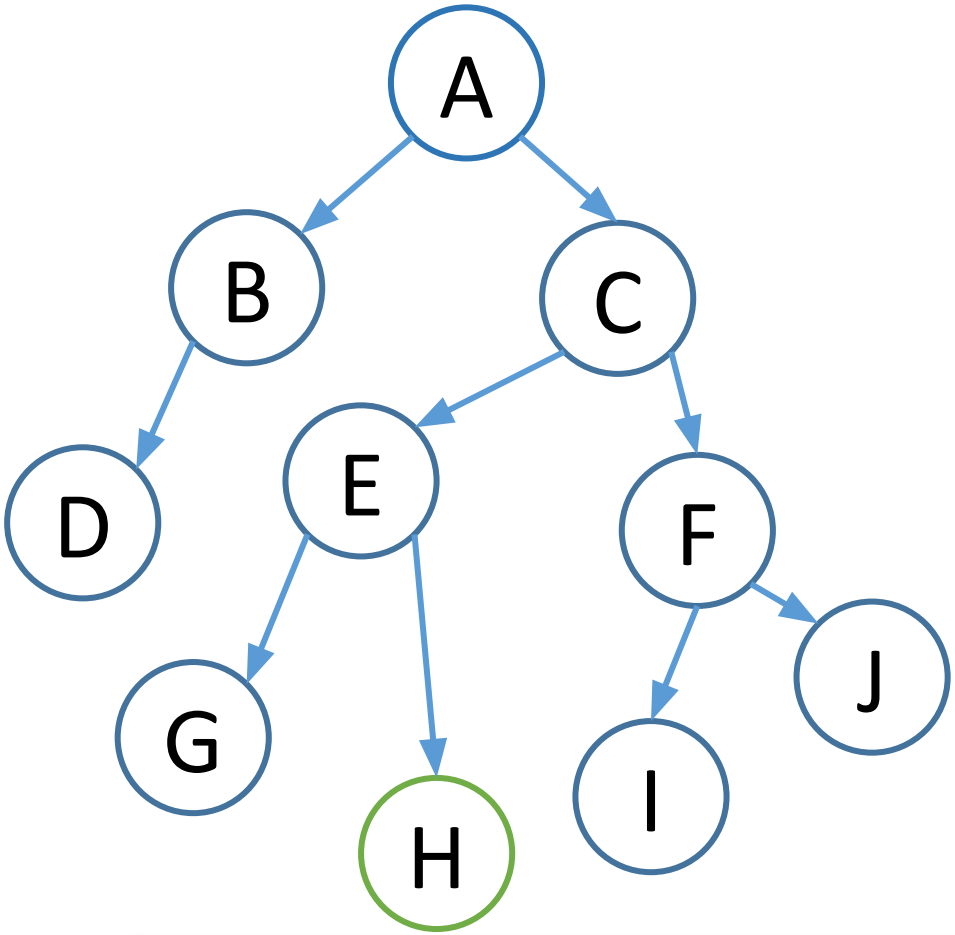
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E,G}
expansion: {A,B,C,E}



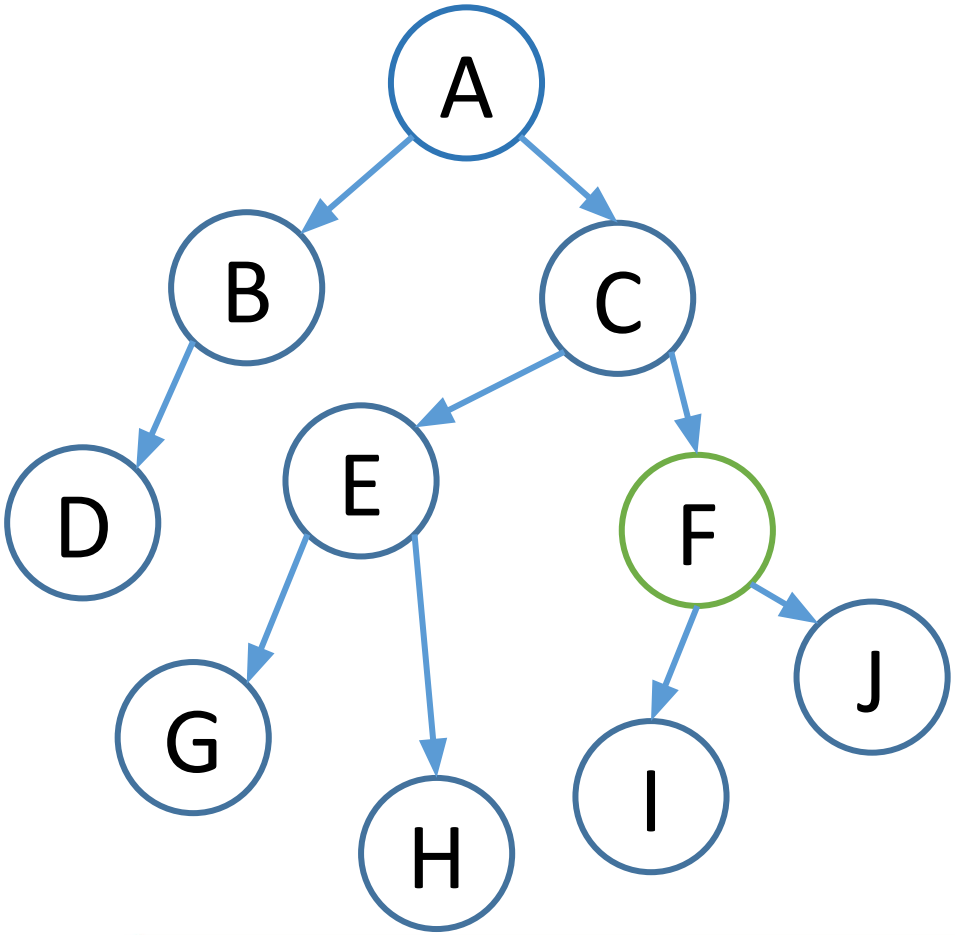
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E,G,H}
expansion: {A,B,C,E}



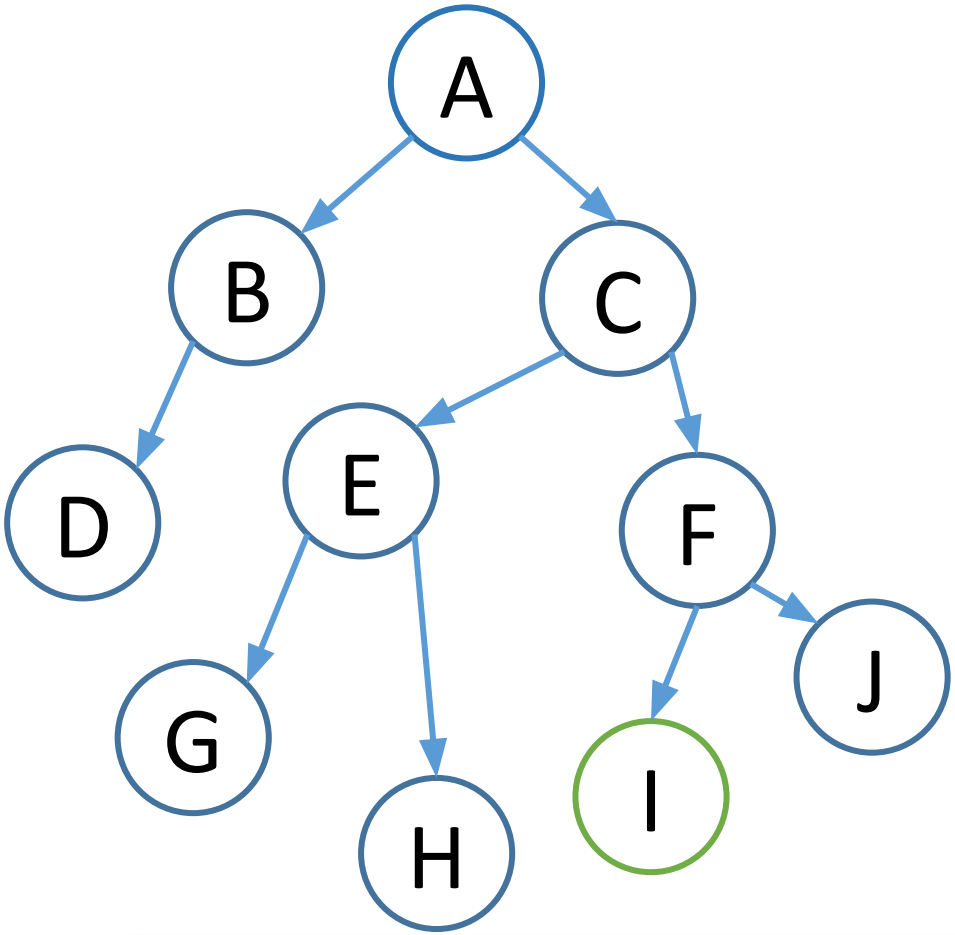
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E,G,H,F}
expansion: {A,B,C,E,F}



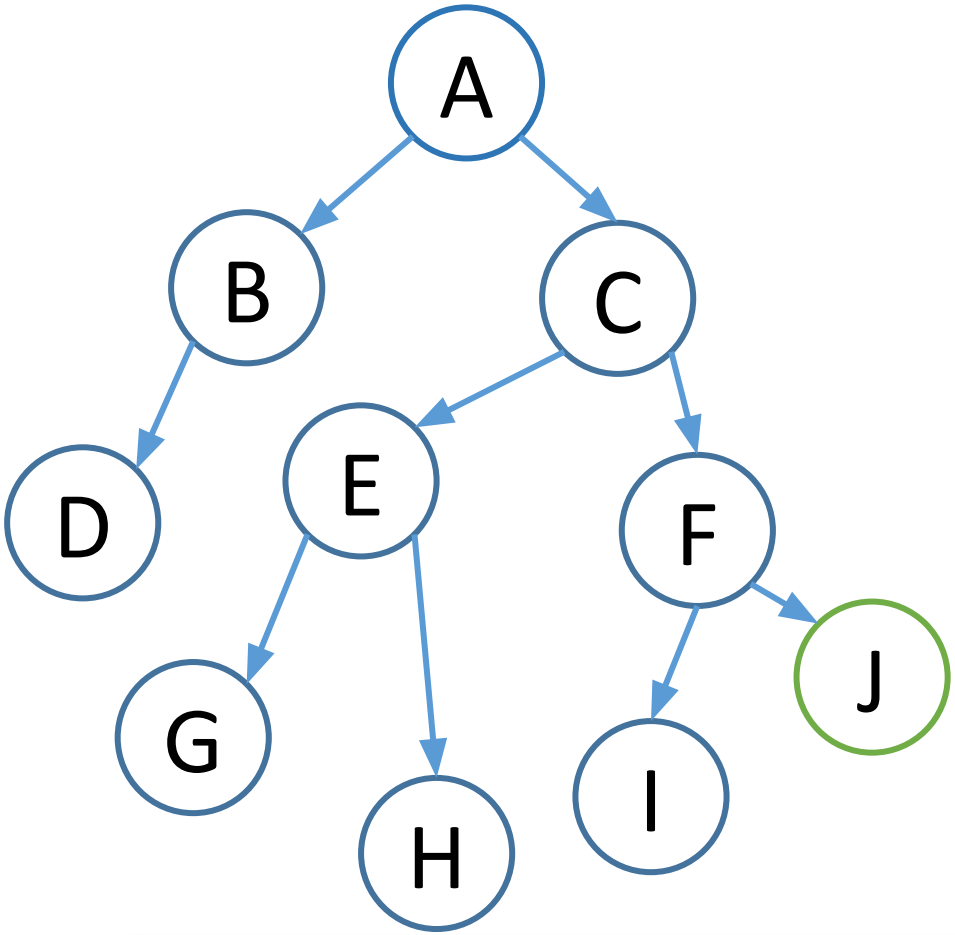
Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E,G,H,F,I}
expansion: {A,B,C,E,F}



Breadth-first, Depth-first

Depth-first search:
visit: {A,B,D,C,E,G,H,F,I,J}
expansion: {A,B,C,E,F}

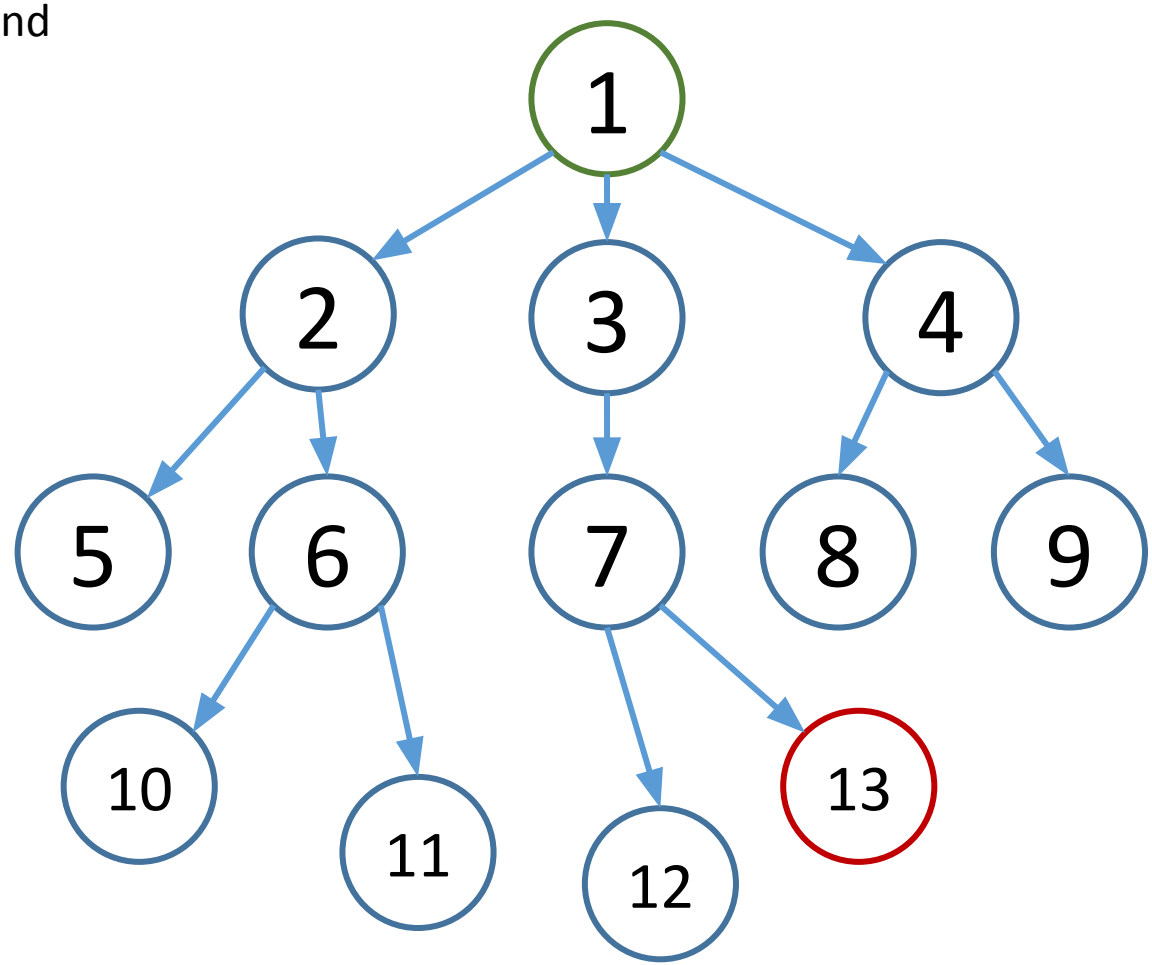


Example: Iterative Deepening Search

IDS: write the node **expansion** and node **visit.** Goal node: **13**

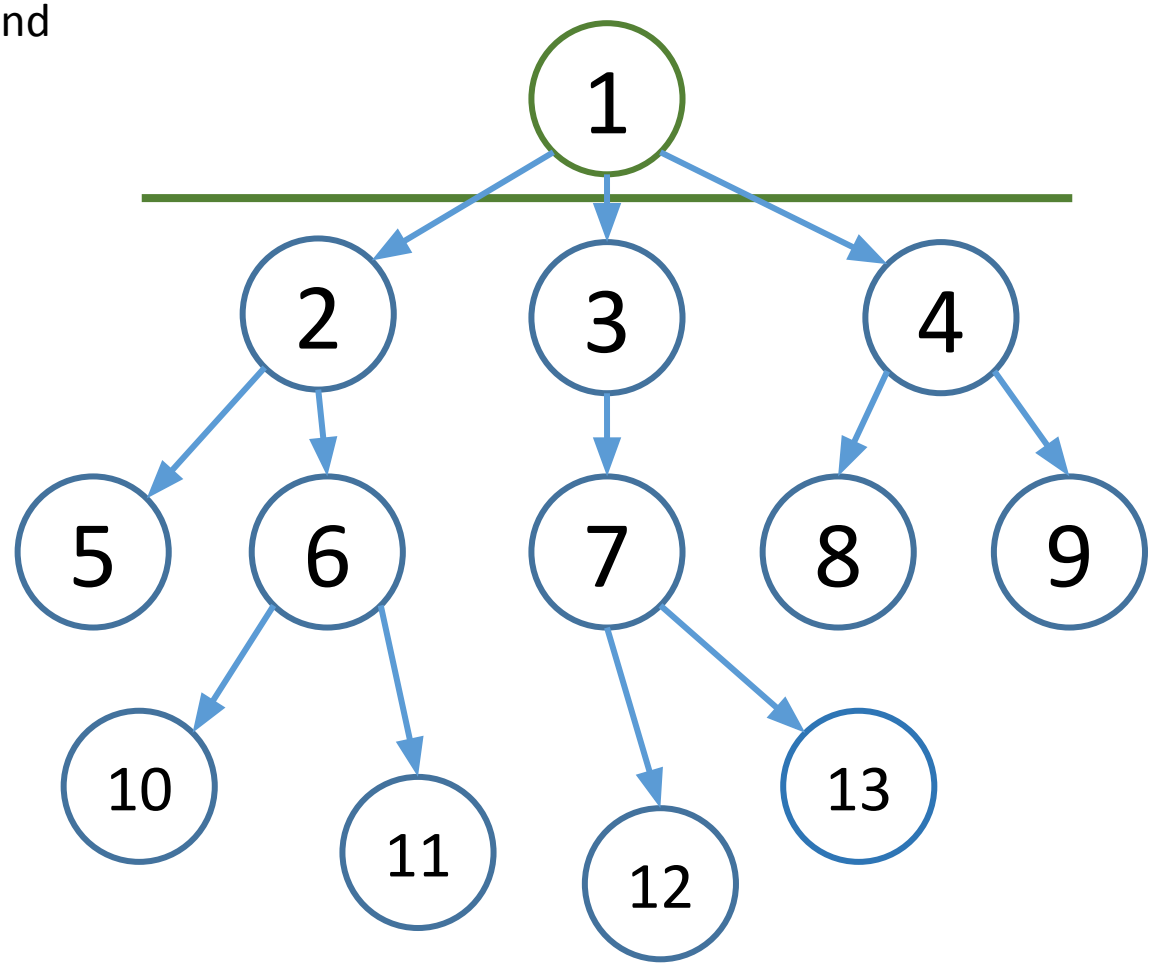
Expansion:

Visit:



Example: Iterative Deepening Search

IDS: write the node **expansion** and node **visit.** Goal node: **13**

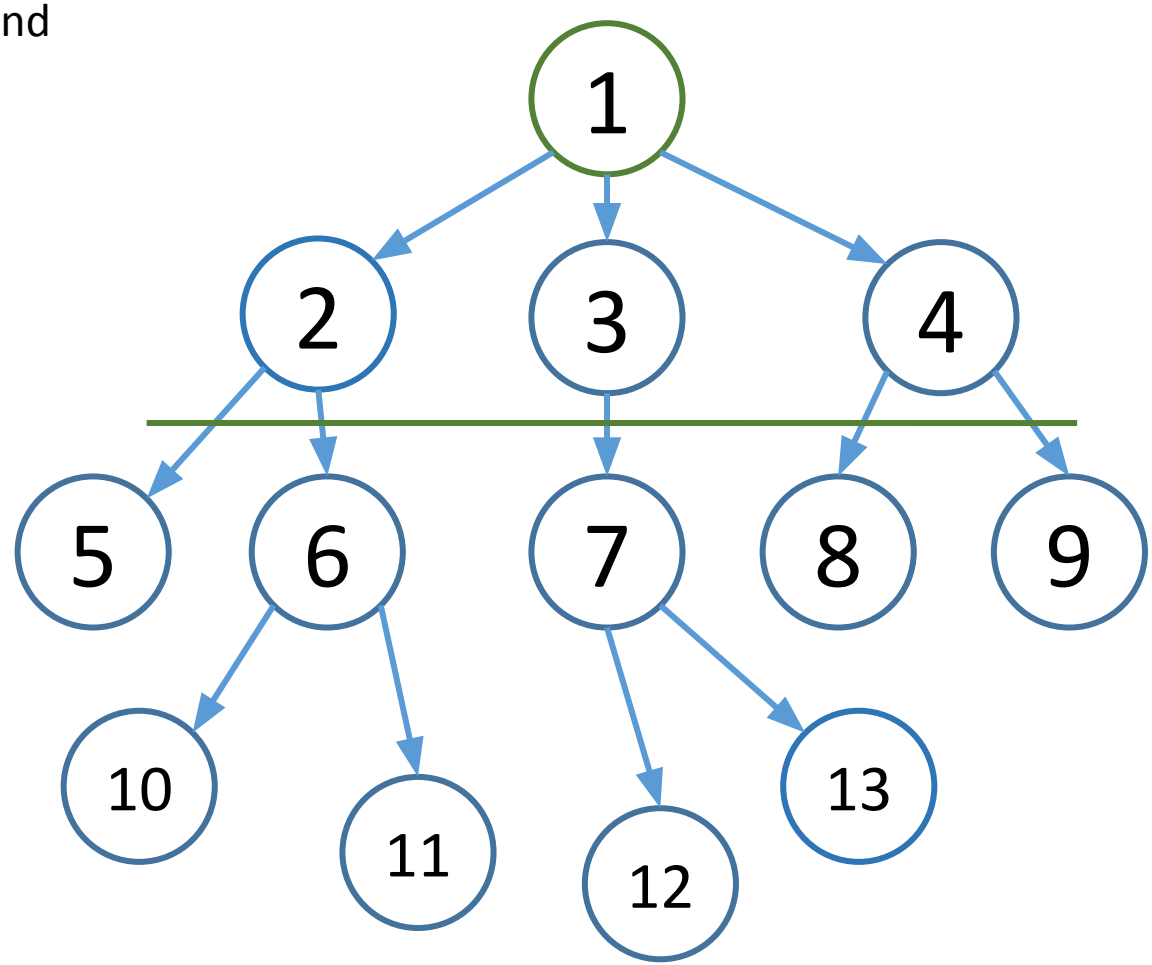


Expansion:
Level 0 : {}

Visit:
Level 0 : {1}

Example: Iterative Deepening Search

IDS: write the node **expansion** and node **visit.** Goal node: **13**

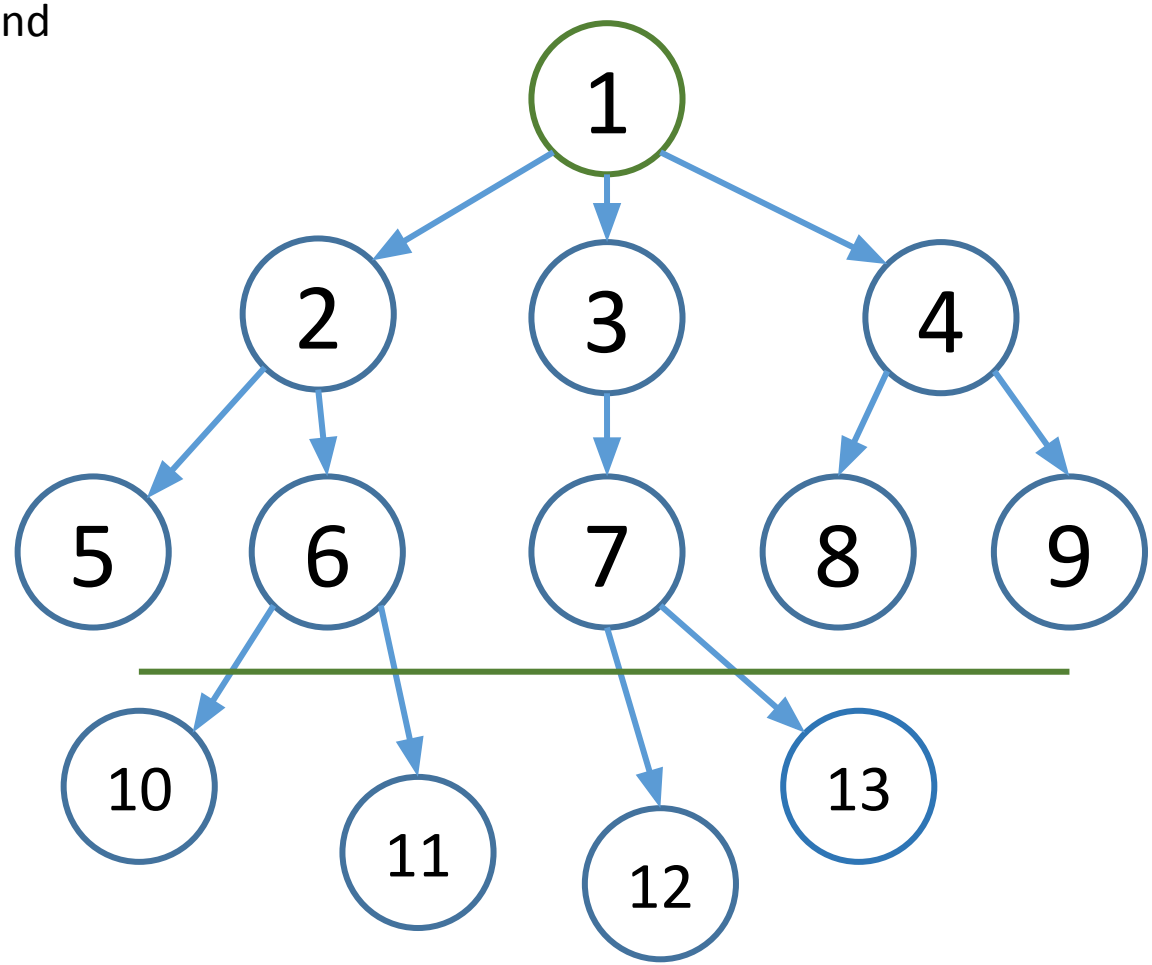


Expansion:
Level 0 : {}
Level 1 : {1}

Visit:
Level 0 : {1}
Level 1 : {1,2,3,4}

Example: Iterative Deepening Search

IDS: write the node **expansion** and node **visit.** Goal node: **13**

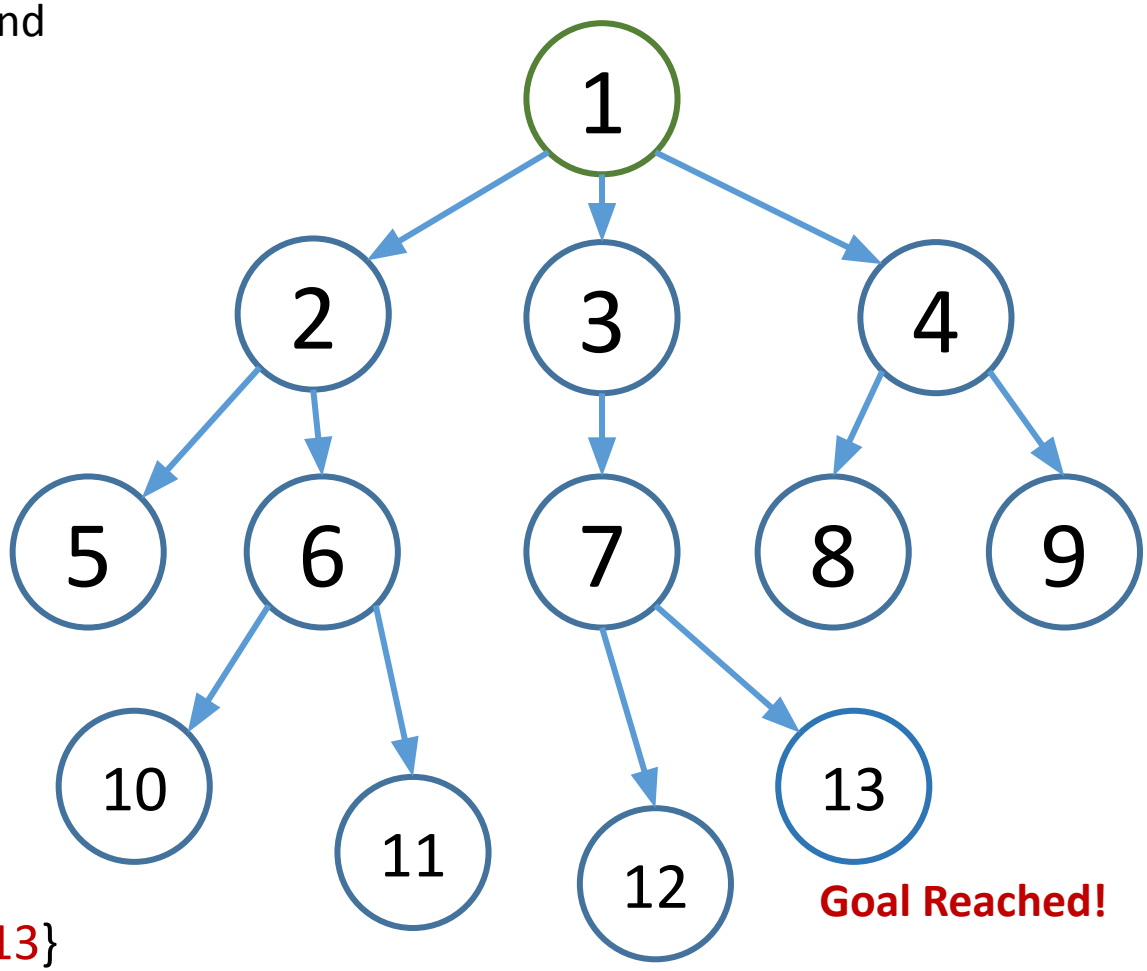


Expansion:
Level 0 : {}
Level 1 : {1}
Level 2 : {1,2,3,4}

Visit:
Level 0 : {1}
Level 1 : {1,2,3,4}
Level 2 : {1,2,5,6,3,7,4,8,9}

Example: Iterative Deepening Search

IDS: write the node **expansion** and node **visit.** Goal node: **13**



Expansion:
Level 0 : {}
Level 1 : {1}
Level 2 : {1,2,3,4}
Level 3 : {1,2,6,3,7}

Visit:
Level 0 : {1}
Level 1 : {1,2,3,4}
Level 2 : {1,2,5,6,3,7,4,8,9}
Level 3 : {1,2,5,6,10,11,3,7,12,13}