Graph Search: Breadth First Search

ROB 102: Introduction to AI & Programming

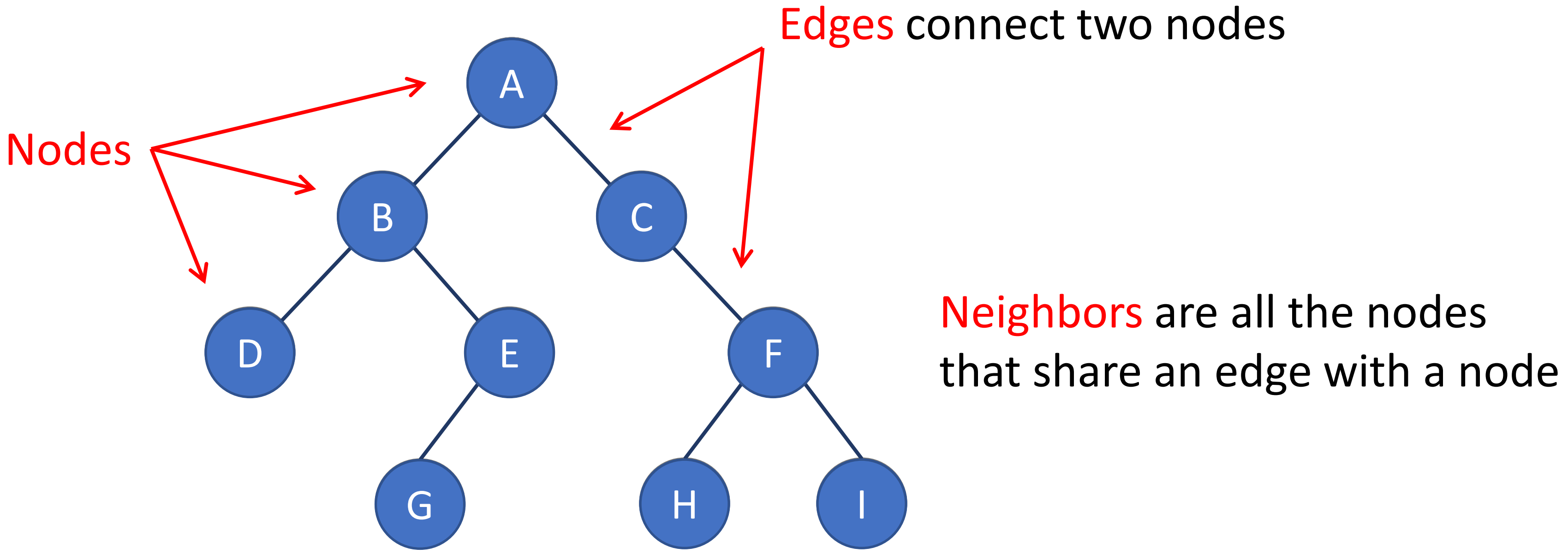
2021/11/09

Today

1. Breadth First Search (BFS)
    - Review
    - Examples
2. Group BFS activity
3. C++ activity

Graph Search: Example

Find a path from A to H.



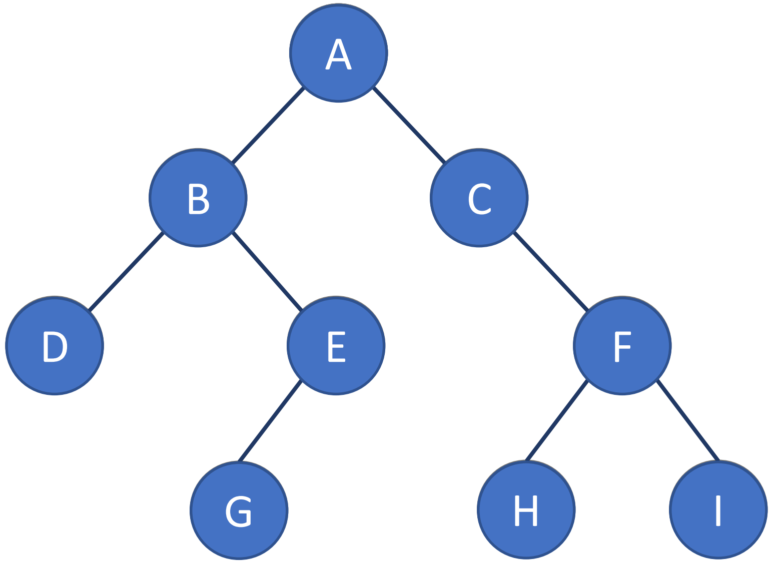
Nodes

Edges connect two nodes

Neighbors are all the nodes that share an edge with a node

Breadth First Search Example (1)

Find a path from A to H.



Nodes to visit next

Visit list:

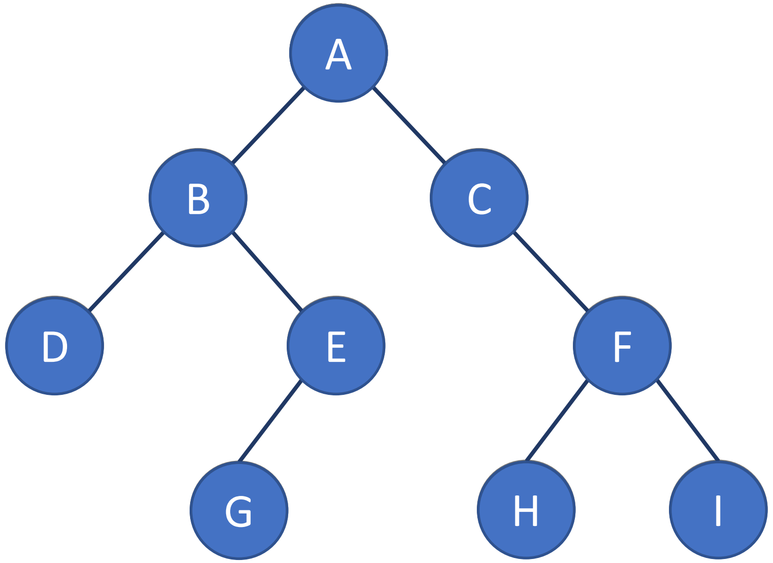
Current node:

Visited nodes:

Fully explored nodes

Breadth First Search Example (1)

Find a path from A to H.

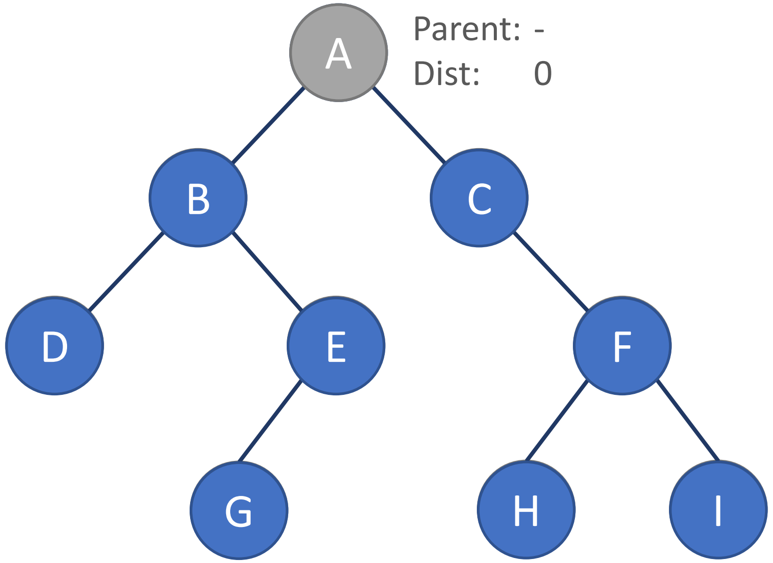


Breadth First Search Algorithm

1. Add start node to visit list. Set distance to zero.
2. Pop the first node from the front of visit list. Set as current node.
3. If current node is goal, go to 6.
4. Add each unvisited neighbor of current to back of visit list.
5. Set parent of each neighbor to current, set distance to current distance + 1. Go to 2.
6. Trace path backwards through parents.

Breadth First Search Example (1)

Find a path from A to H.



Parent: -
Dist: 0

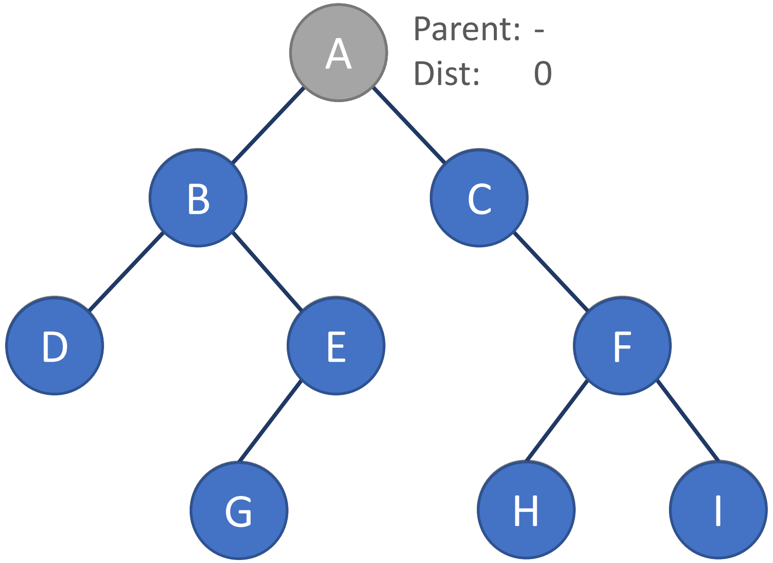
Visit list:

Current node:

Visited nodes:

Breadth First Search Example (1)

Find a path from A to H.



Parent: -
Dist: 0

Visit list:

[ ]

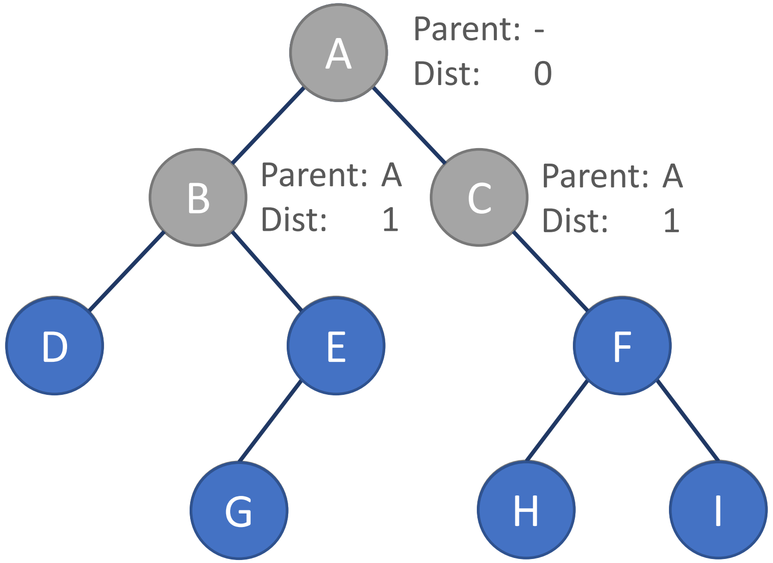
Current node: A

Visited nodes:

[ ]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[B C]

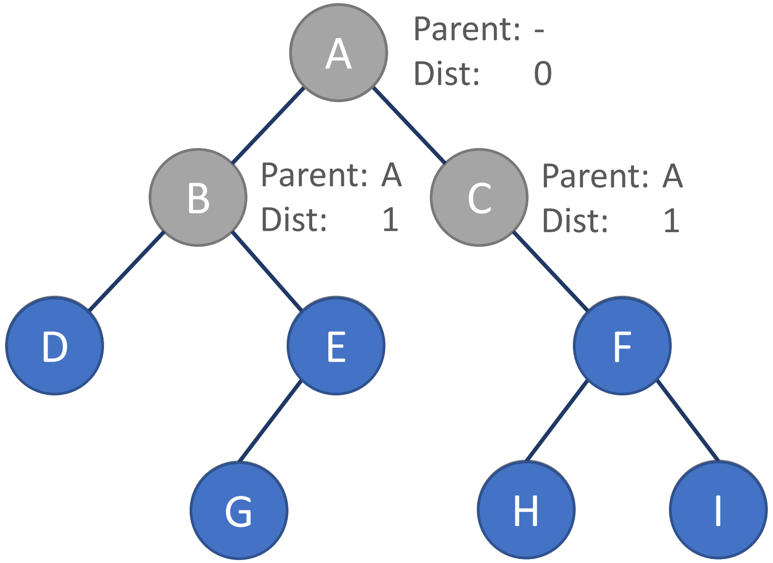
Current node: A

Visited nodes:

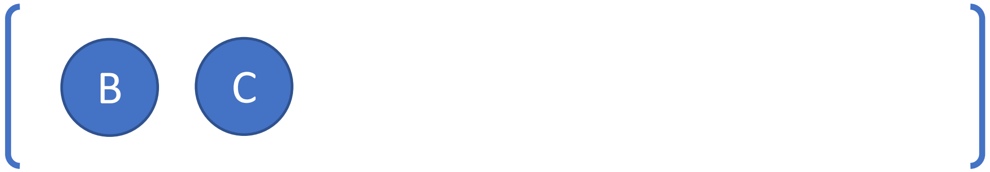
[ ]

Breadth First Search Example (1)

Find a path from A to H.

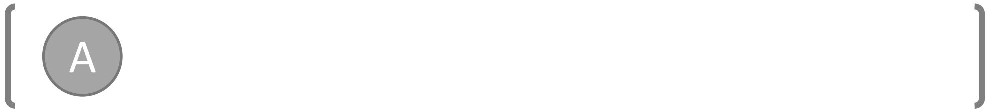


Visit list:



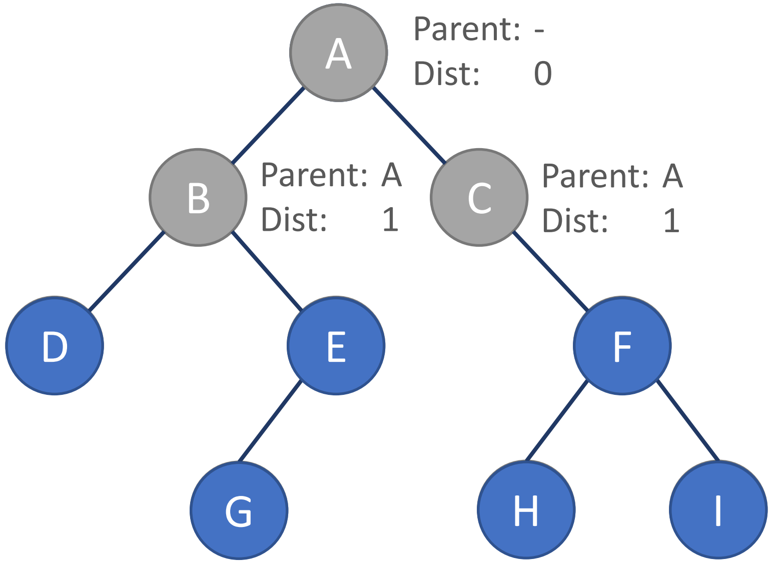
Current node:

Visited nodes:



Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[C]

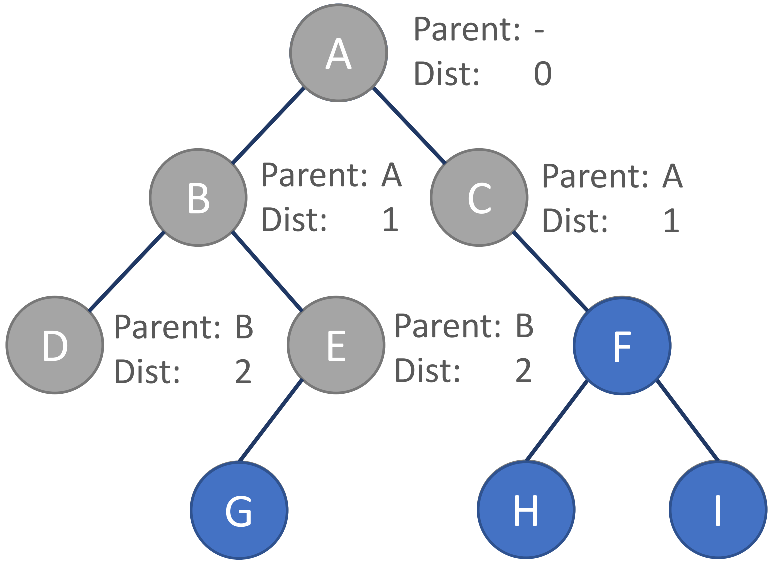
Current node: B

Visited nodes:

[A]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[C D E]

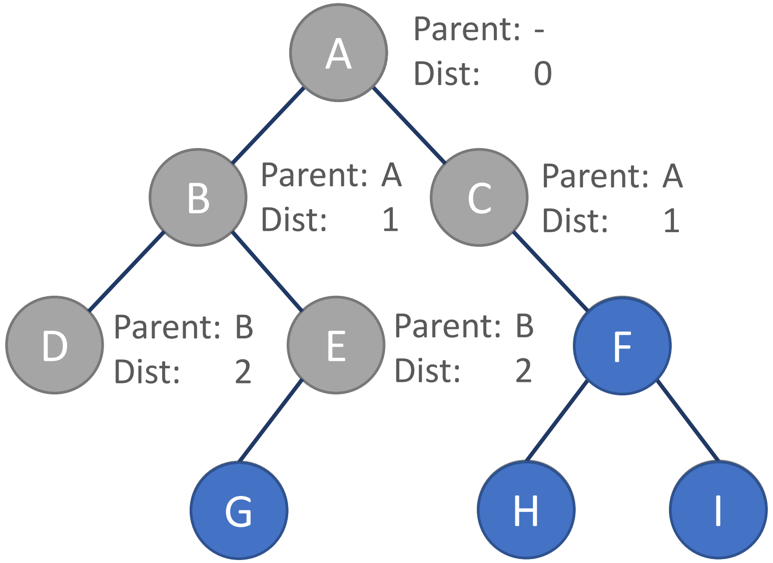
Current node: B

Visited nodes:

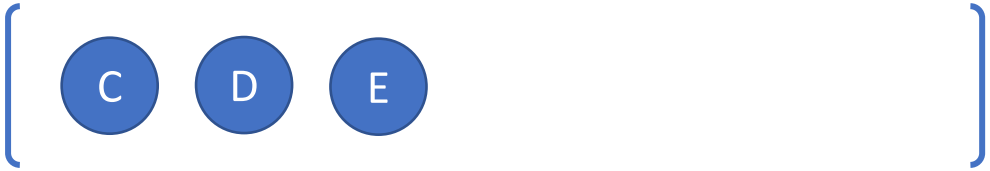
[A]

Breadth First Search Example (1)

Find a path from A to H.

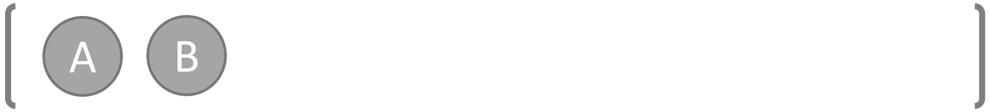


Visit list:



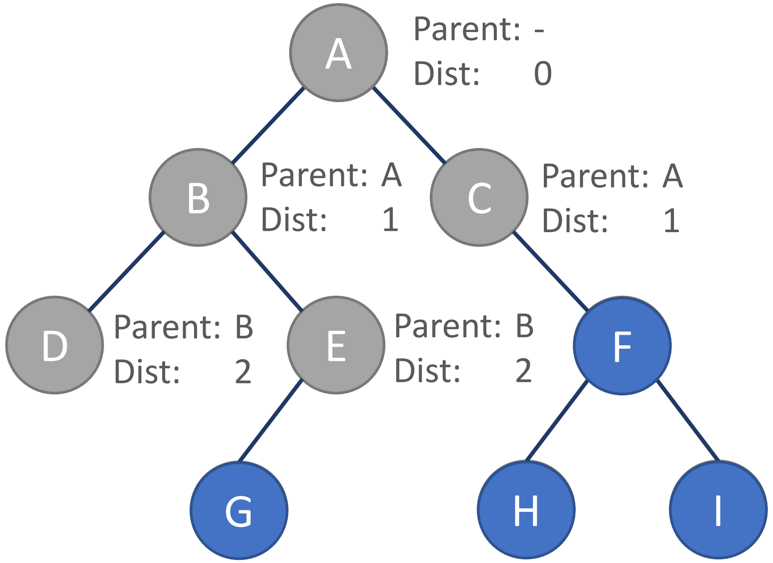
Current node:

Visited nodes:



Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[D E]

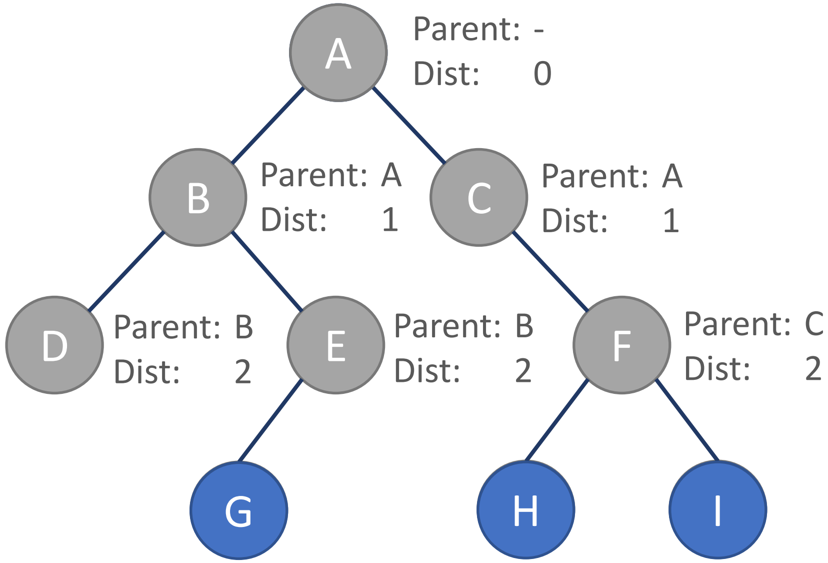
Current node: C

Visited nodes:

[A B]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[D E F]

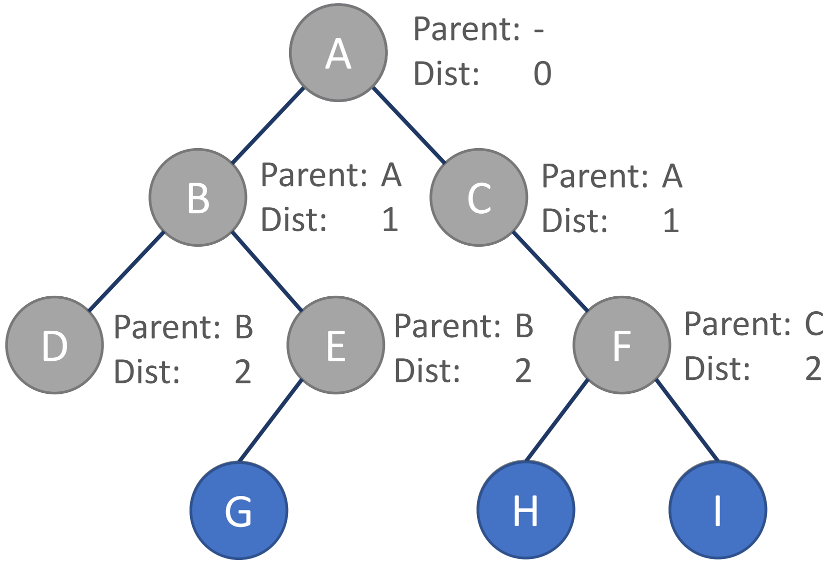
Current node: C

Visited nodes:

[A B]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[D E F]

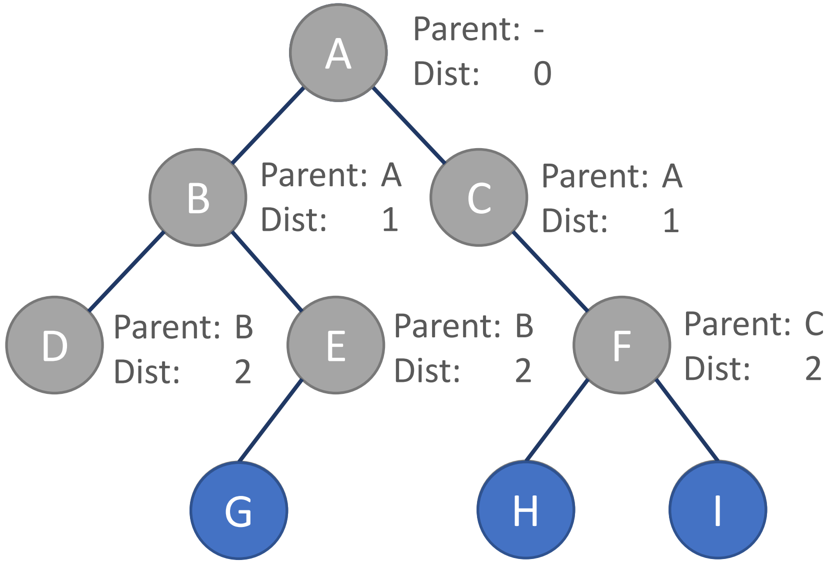
Current node:

Visited nodes:

[A B C]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[E F]

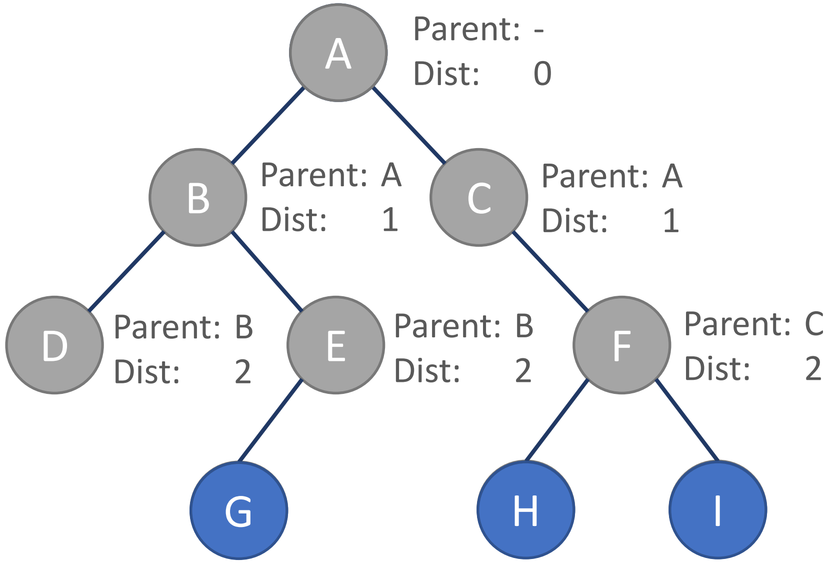
Current node: D

Visited nodes:

[A B C]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[E F]

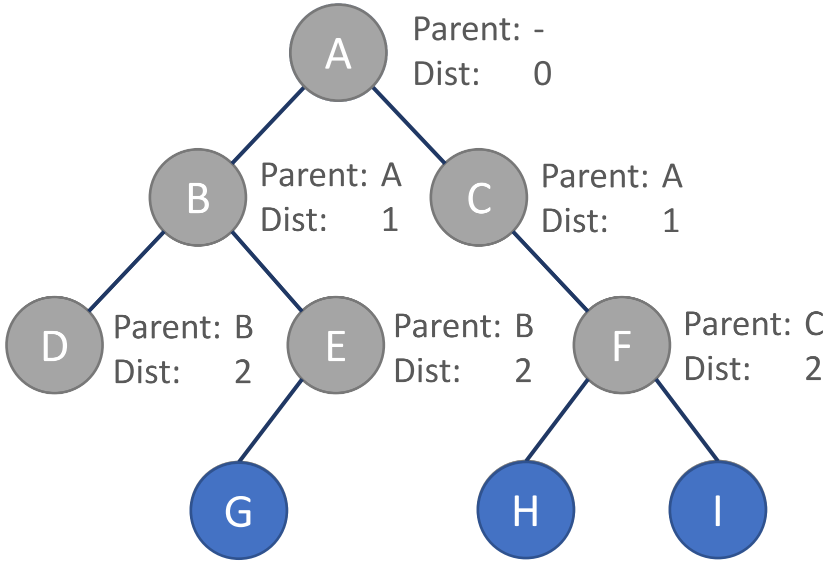
Current node:

Visited nodes:

[A B C D]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[F]

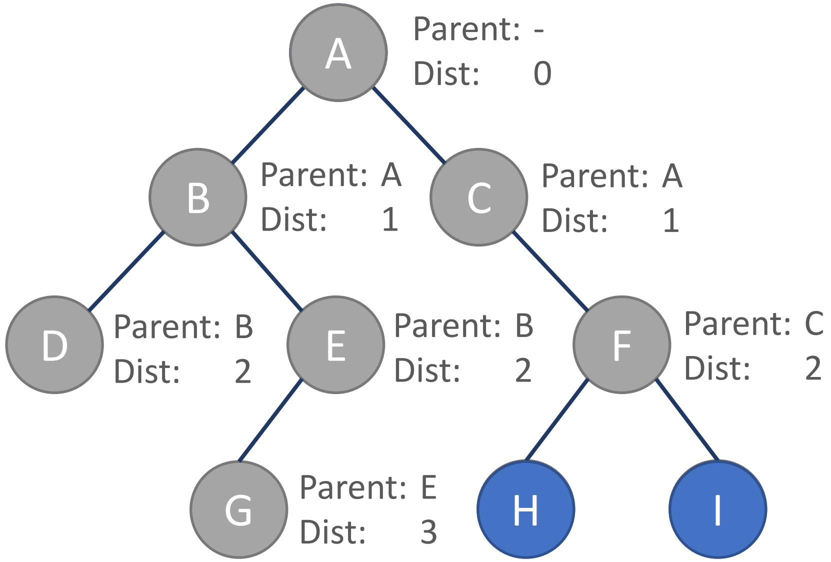
Current node: E

Visited nodes:

[A B C D]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[F G]

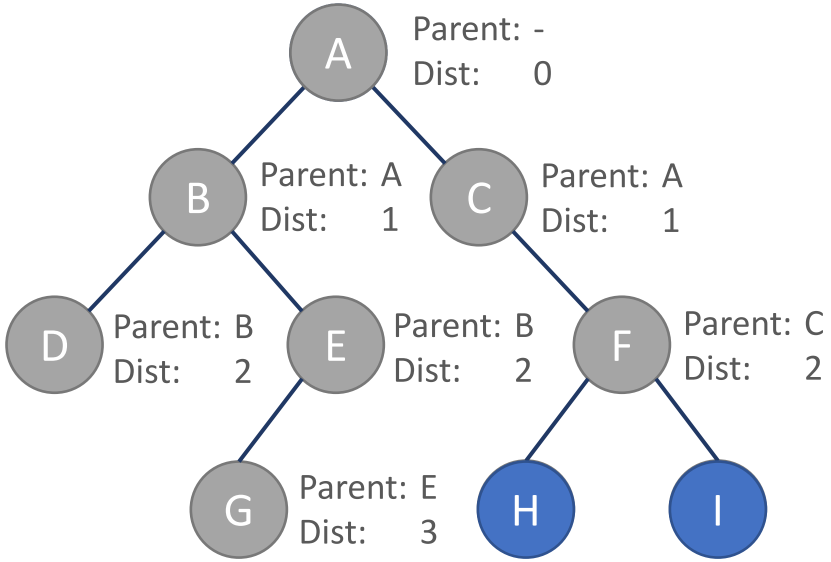
Current node: E

Visited nodes:

[A B C D]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[F G]

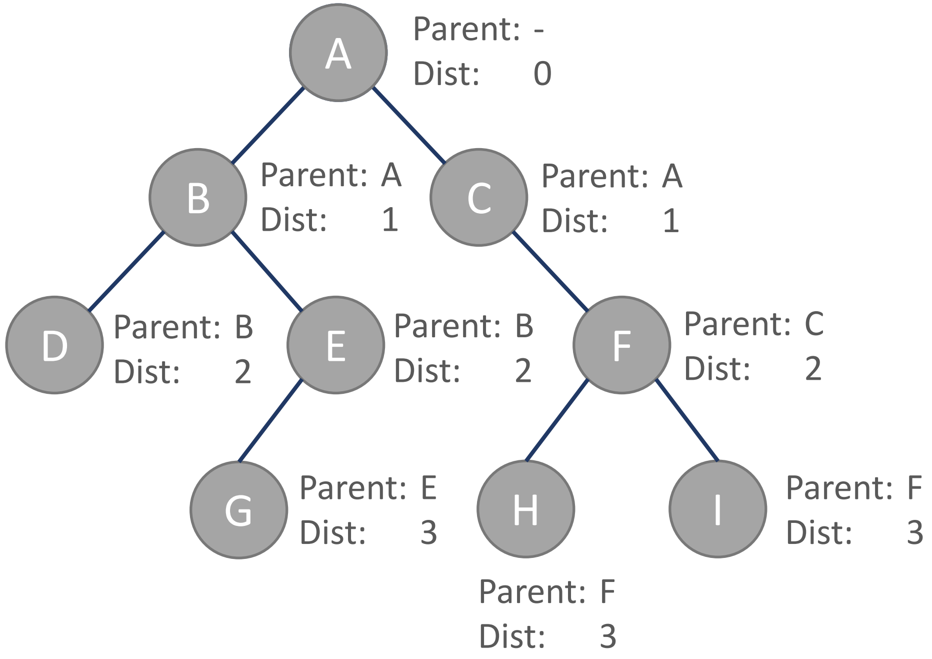
Current node:

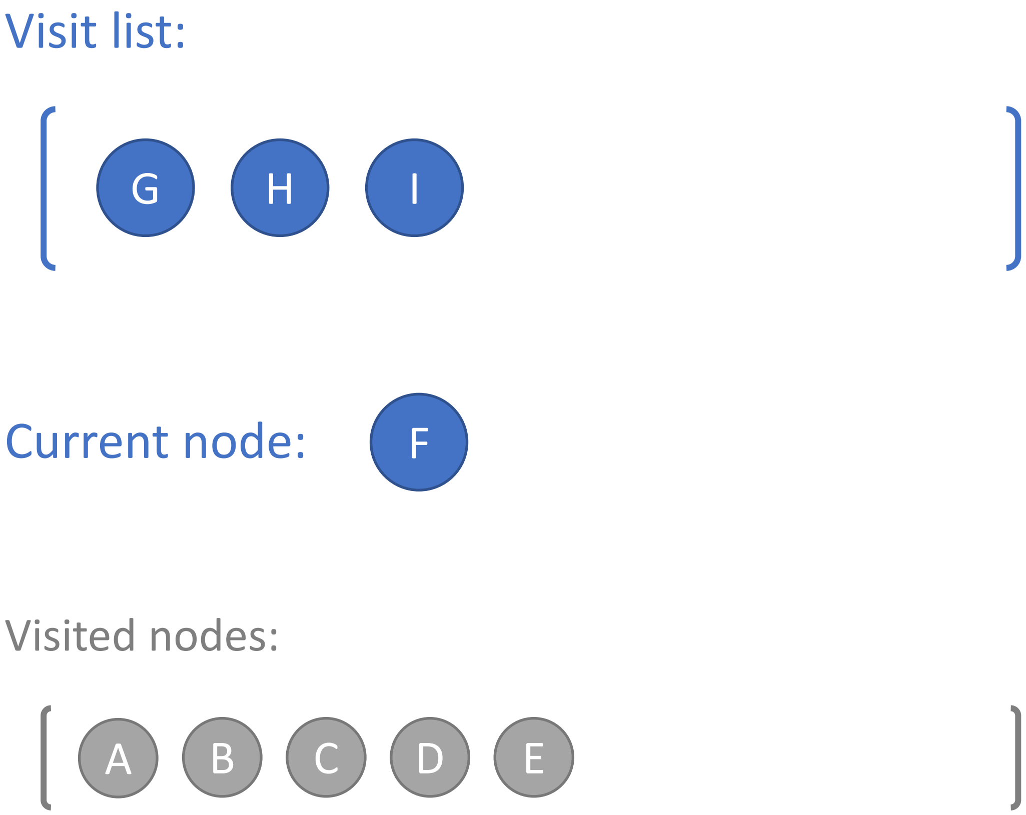
Visited nodes:

[A B C D E]

Breadth First Search Example (1)

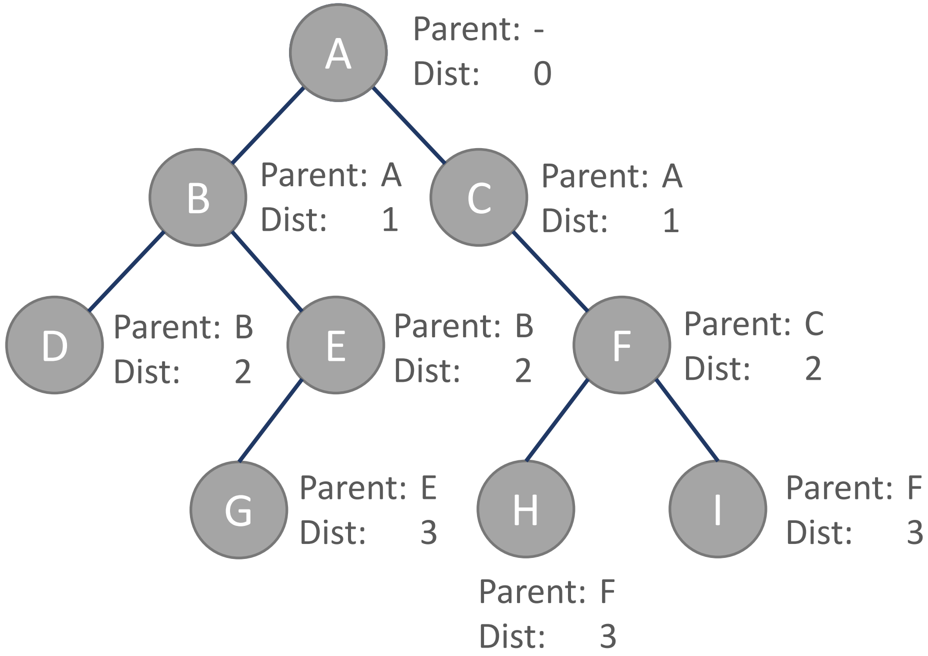
Find a path from A to H.





Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[H I]

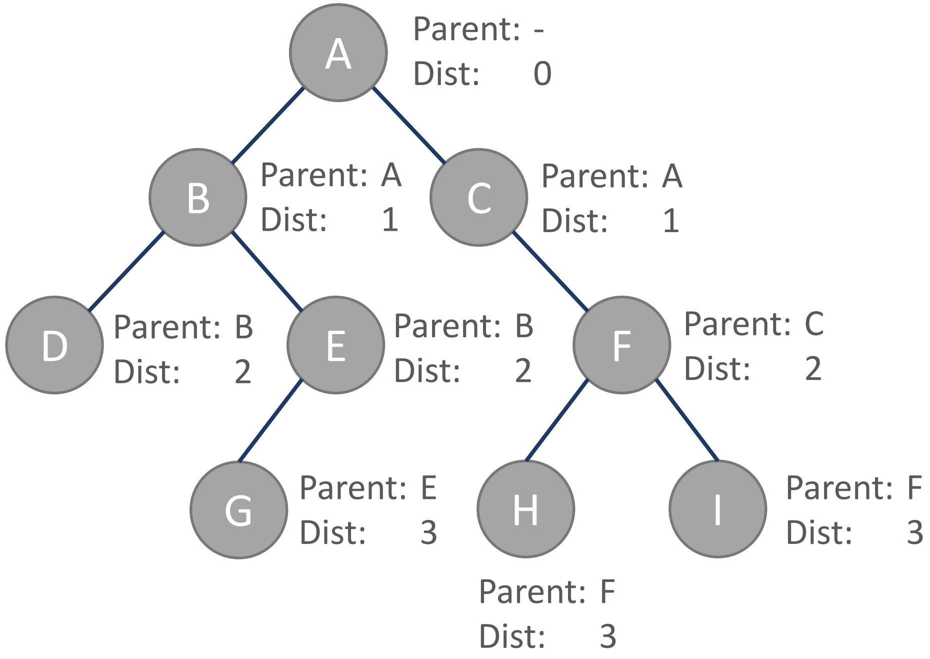
Current node: G

Visited nodes:

[A B C D E F]

Breadth First Search Example (1)

Find a path from A to H.



A
Parent: -
Dist: 0

B
Parent: A
Dist: 1

C
Parent: A
Dist: 1

D
Parent: B
Dist: 2

E
Parent: B
Dist: 2

F
Parent: C
Dist: 2

G
Parent: E
Dist: 3

H
Parent: F
Dist: 3

I
Parent: F
Dist: 3

Visit list:

[I]

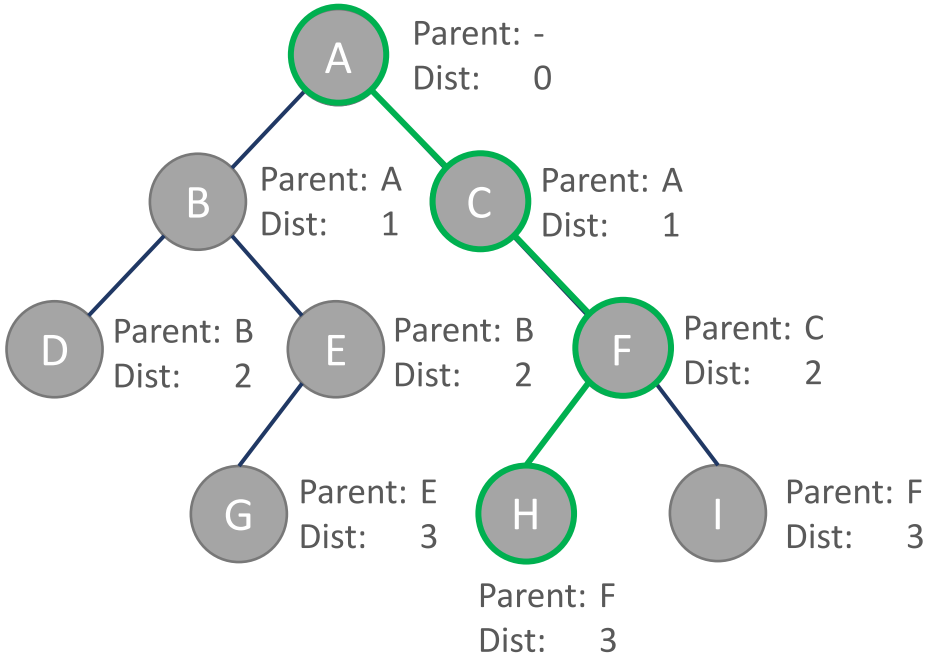
Current node: H Goal!

Visited nodes:

[A B C D E F G]

Breadth First Search Example (1)

Find a path from A to H.



Visit list:

[I]

Current node: H

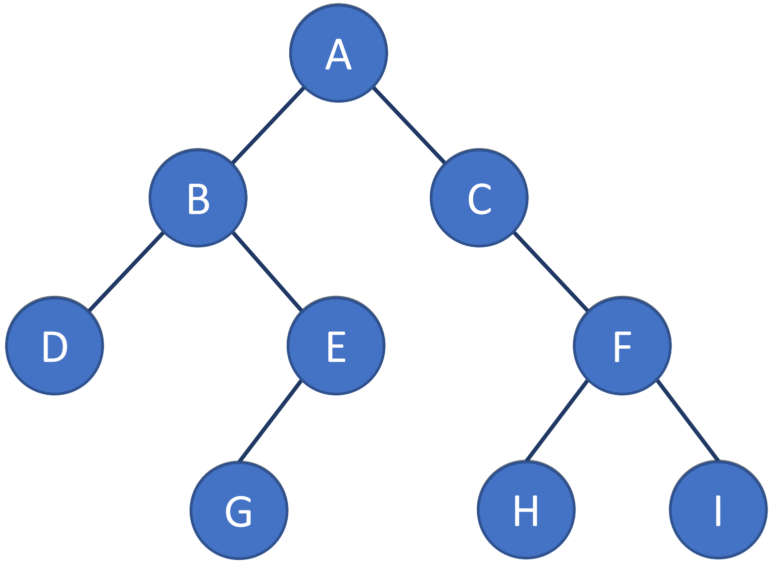
Goal!

Visited nodes:

[A B C D E F G]

Breadth First Search Example (1)

Find a path from A to H.

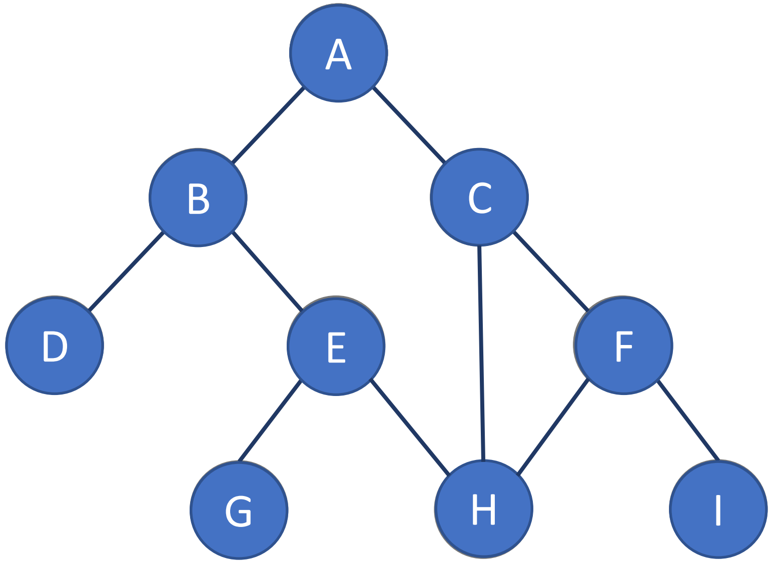


Breadth First Search Algorithm

1. Add start node to visit list. Set distance to zero.
2. Pop the first node from the front of visit list. Set as current node.
3. If current node is goal, go to 6.
4. Add each unvisited neighbor of current to back of visit list.
5. Set parent of each neighbor to current, set distance to current distance + 1. Go to 2.
6. Trace path backwards through parents.

Breadth First Search Example (2)

Find a path from A to H.



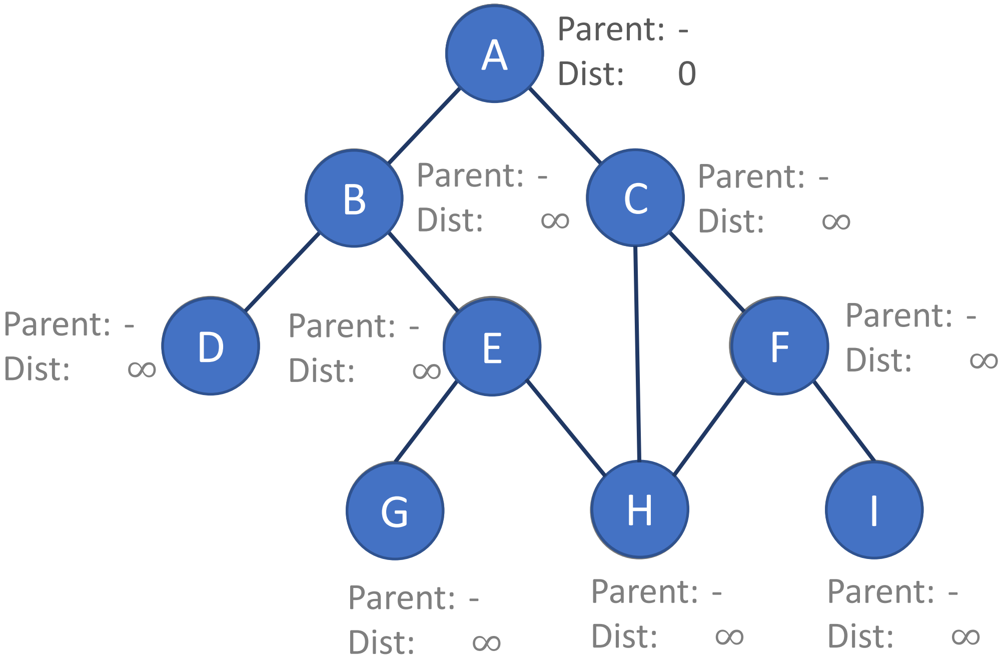
What if the graph has loops?

Update the neighbor’s parent and distance only if its distance is greater than the current node’s distance + 1.

```
if nbr.dist > curr.dist + 1 do:
    nbr.dist = curr.dist + 1
    nbr.parent = curr
```

Breadth First Search Example (2)

Find a path from A to H.



```
if nbr.dist > curr.dist + 1 do:
    nbr.dist = curr.dist + 1
    nbr.parent = curr
```

Visit list:

[ ]

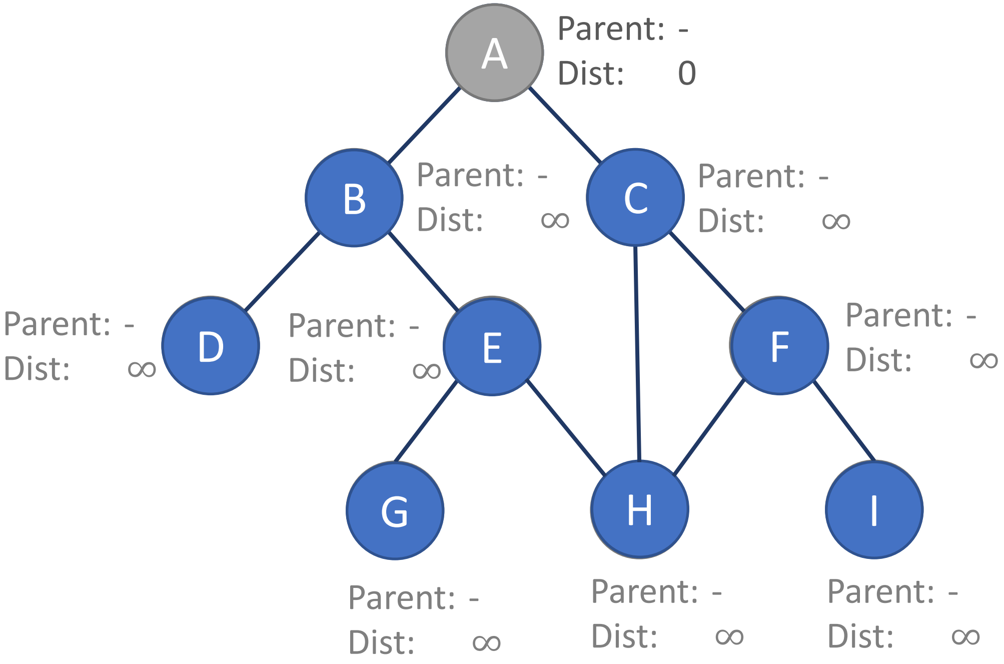
Current node:

Visited nodes:

[ ]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[A]

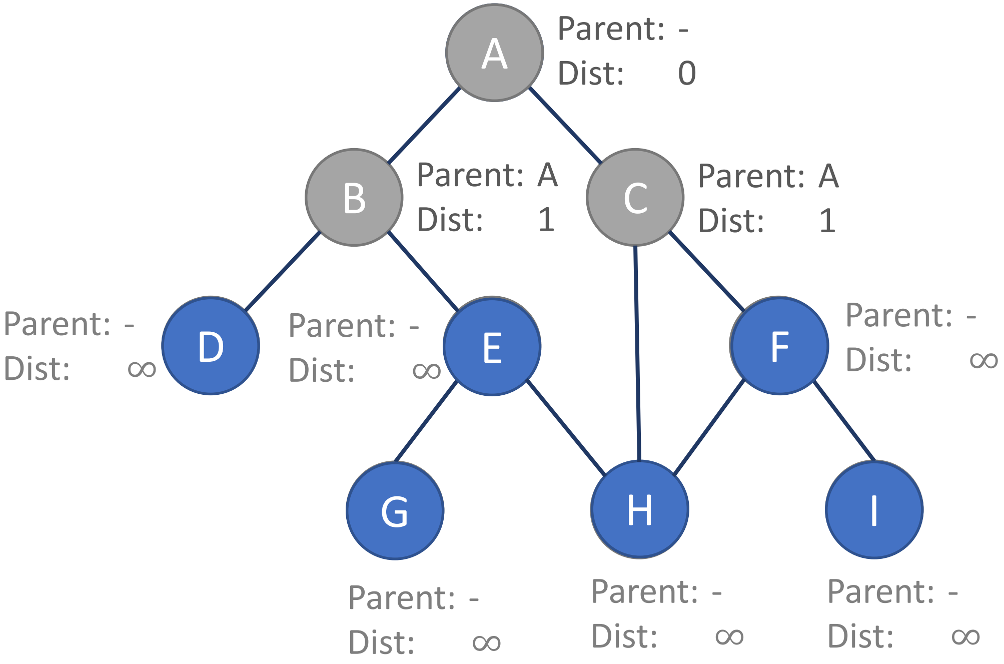
Current node:

Visited nodes:

[ ]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[B C]

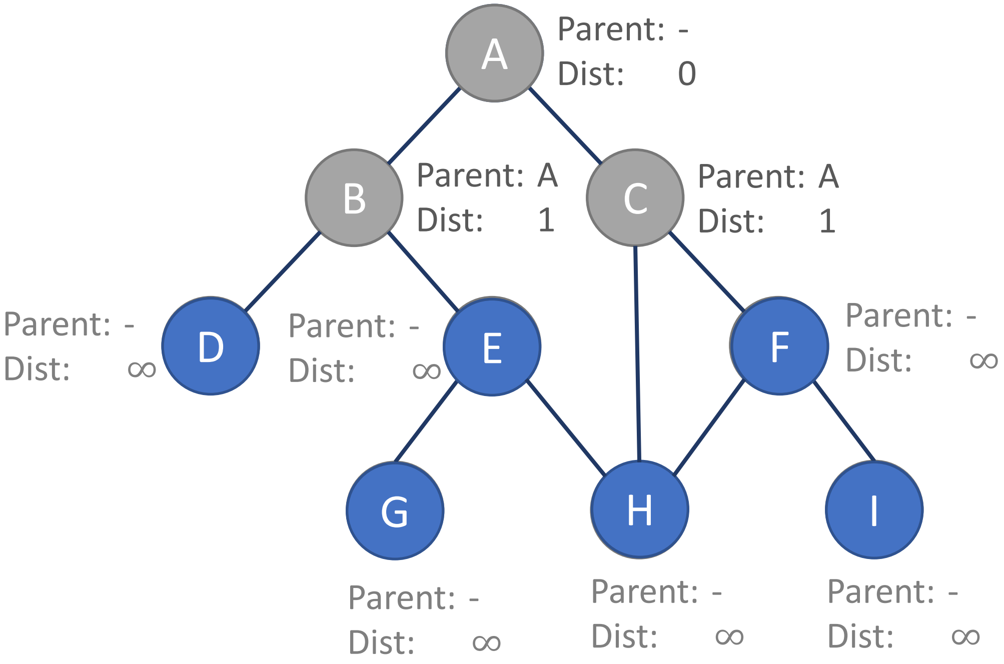
Current node: A

Visited nodes:

[ ]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[C]

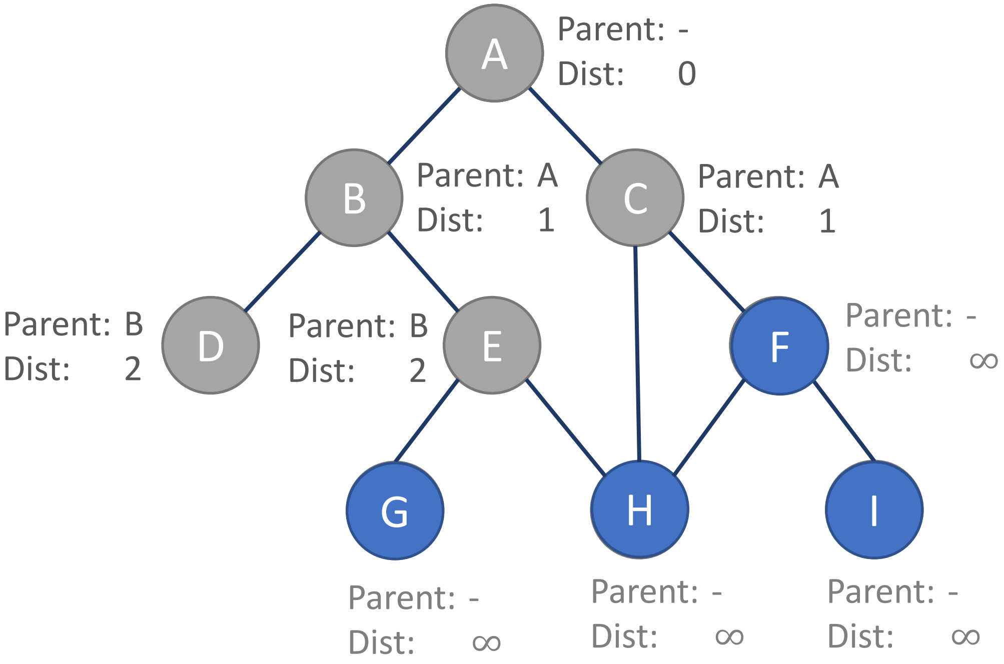
Current node: B

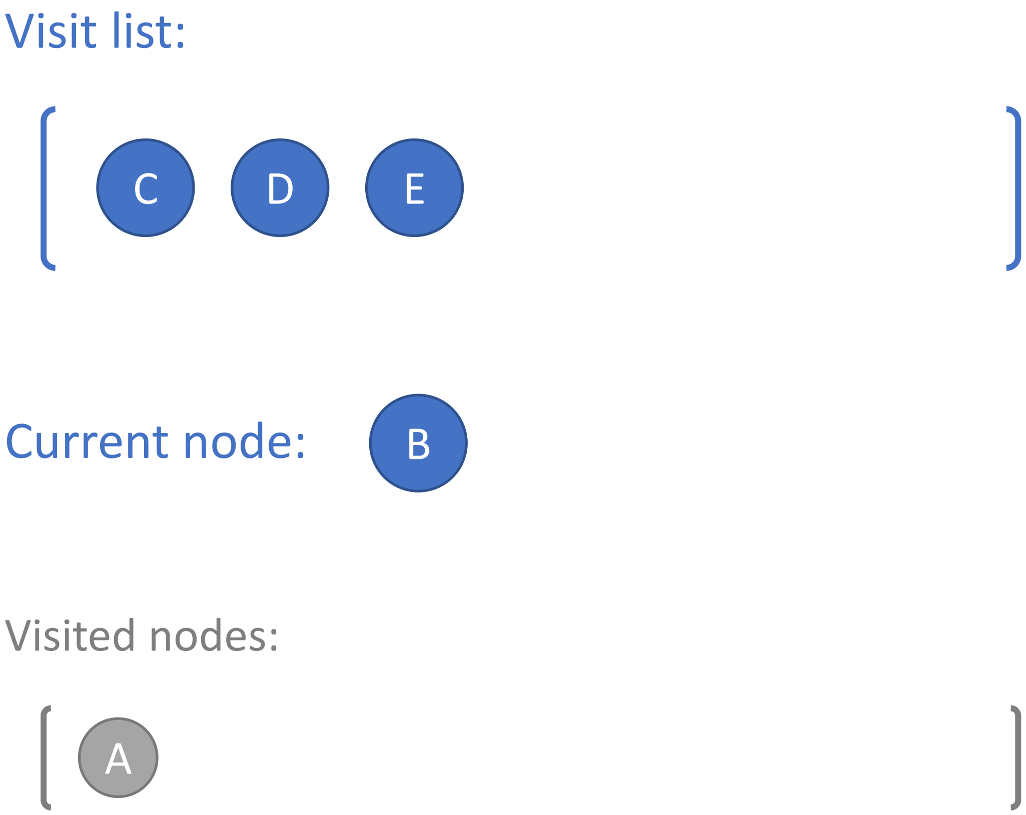
Visited nodes:

[A]

Breadth First Search Example (2)

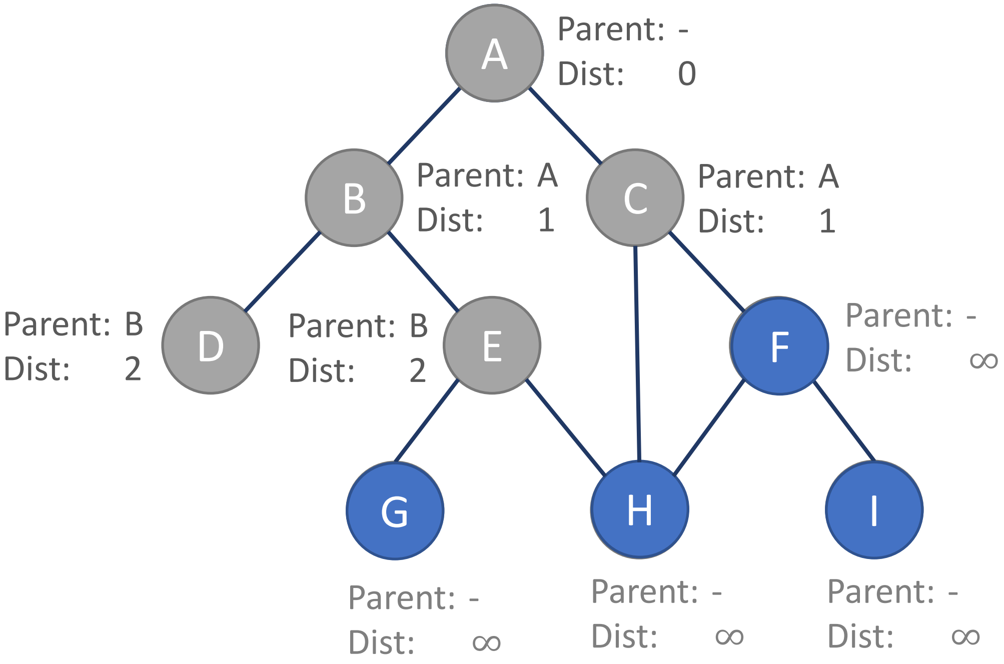
Find a path from A to H.





Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[D E]

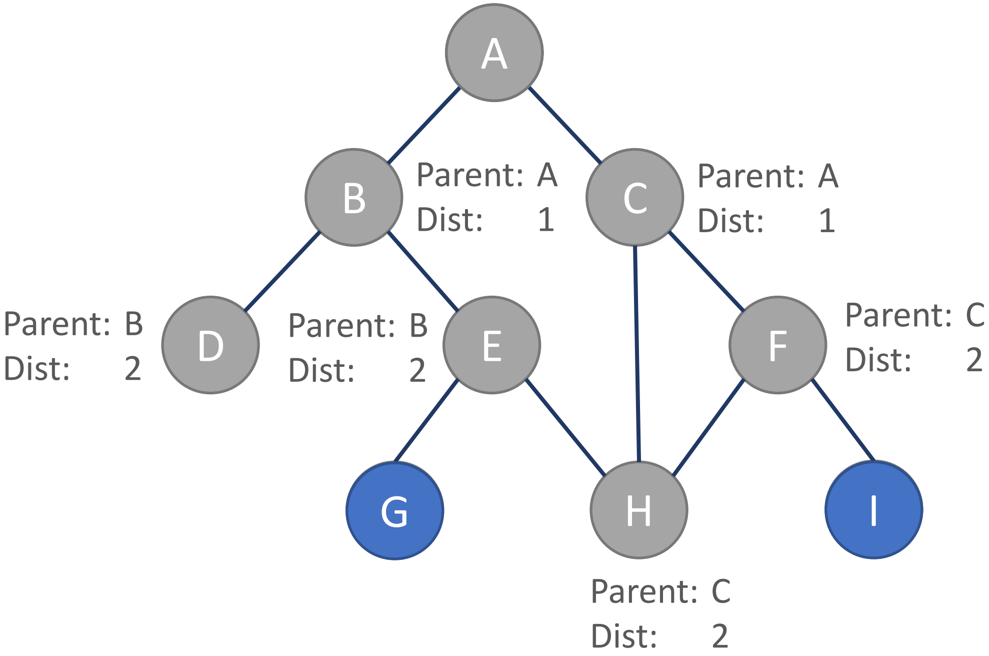
Current node: C

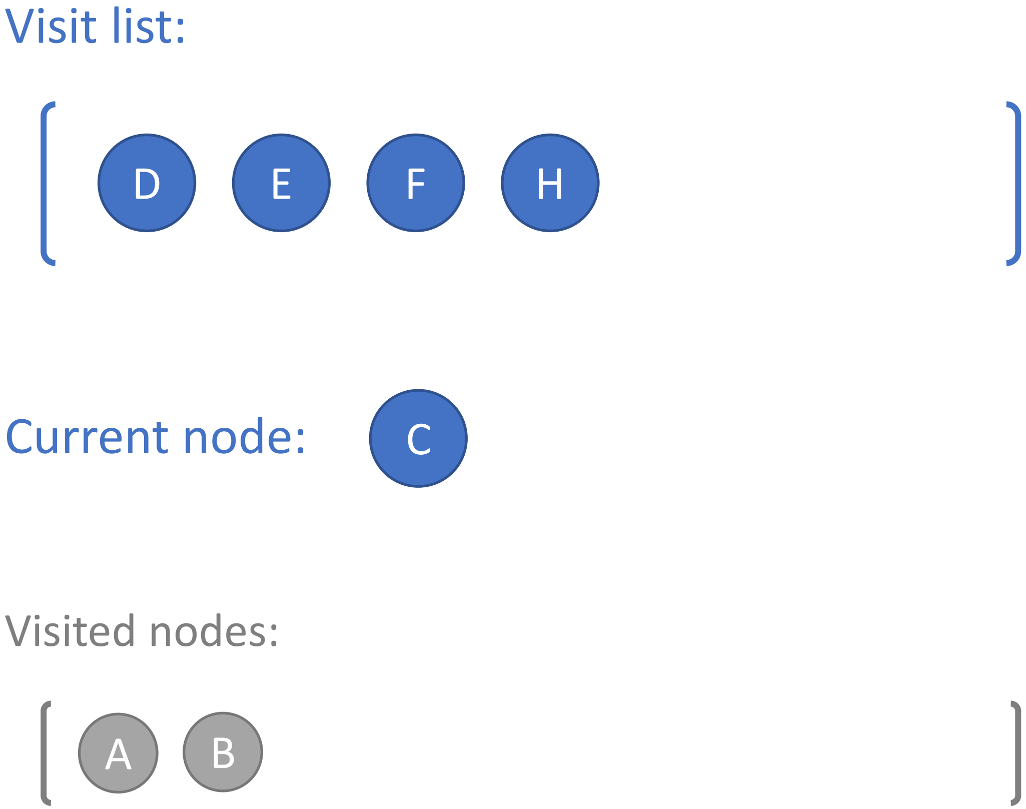
Visited nodes:

[A B]

Breadth First Search Example (2)

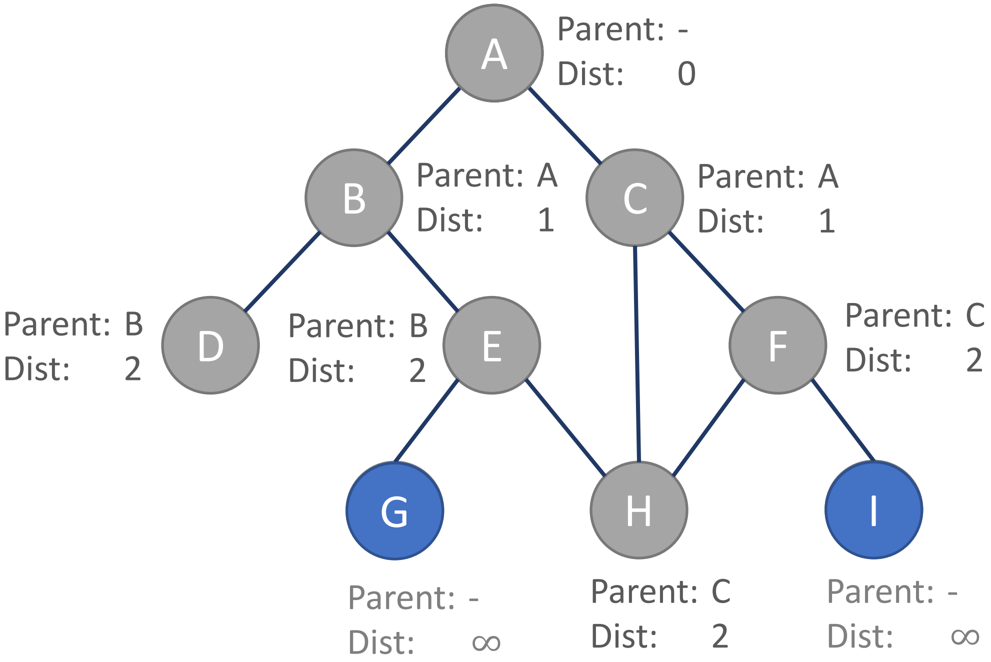
Find a path from A to H.





Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[E F H]

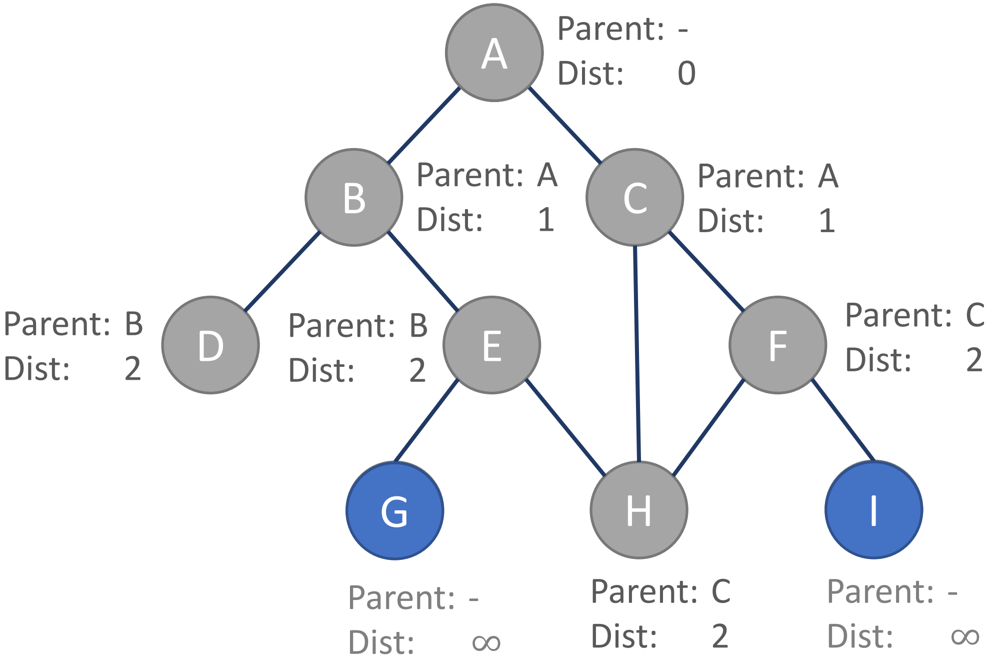
Current node: D

Visited nodes:

[A B C]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[F H]

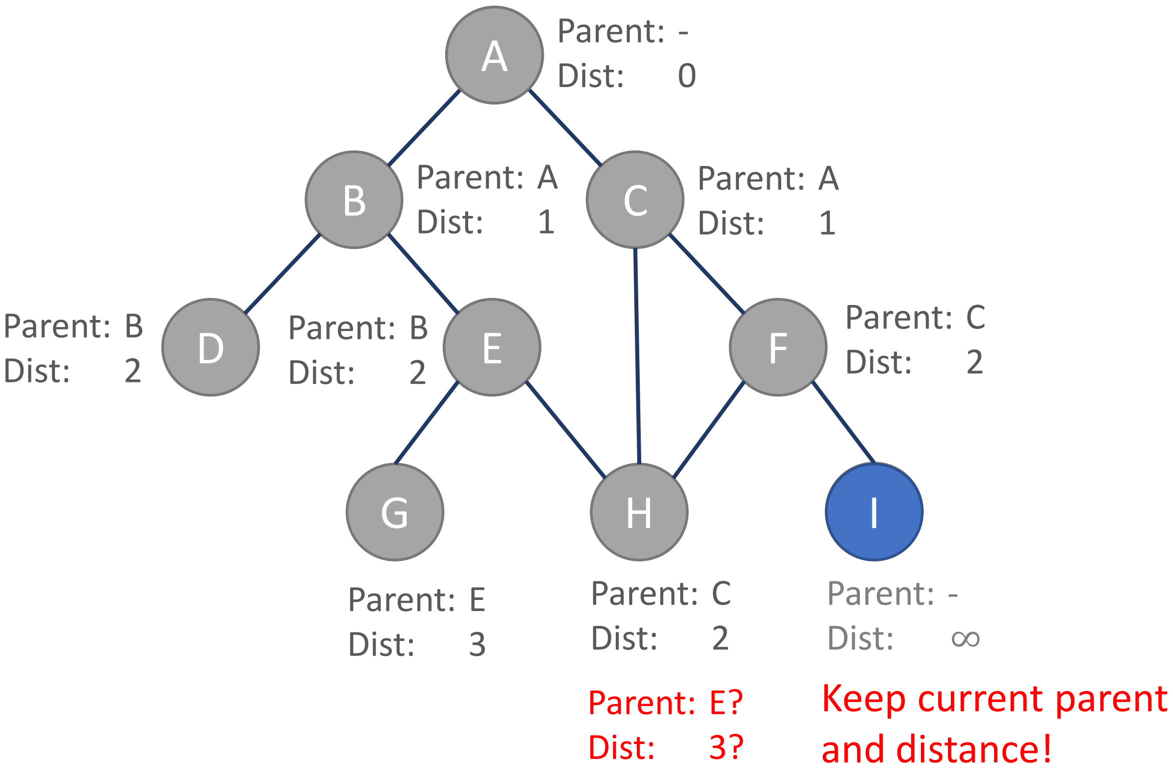
Current node: E

Visited nodes:

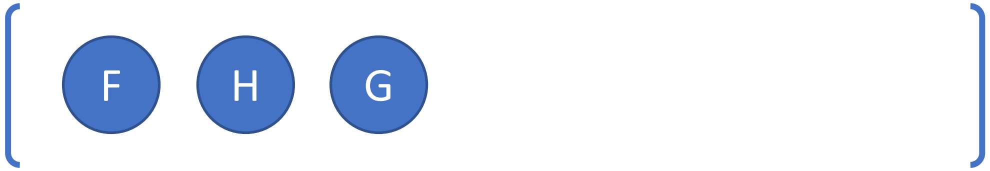
[A B C D]

Breadth First Search Example (2)

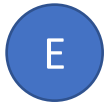
Find a path from A to H.



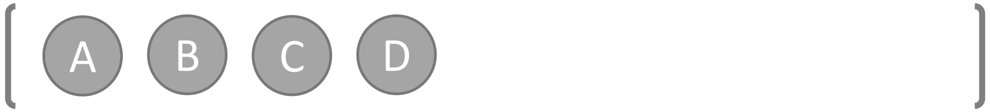
Visit list:



Current node:

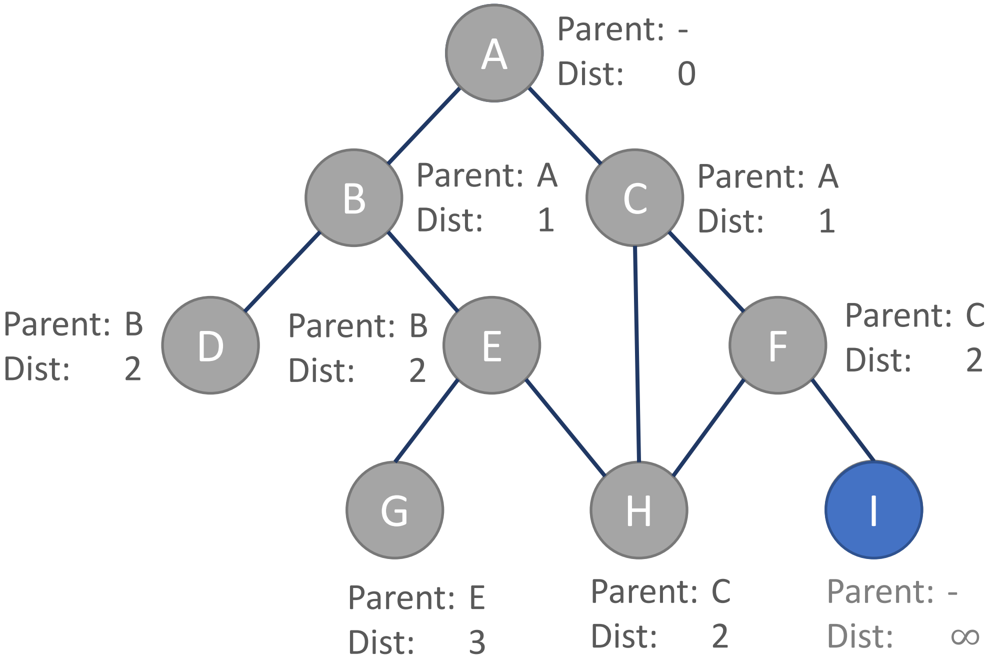


Visited nodes:



Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[H G]

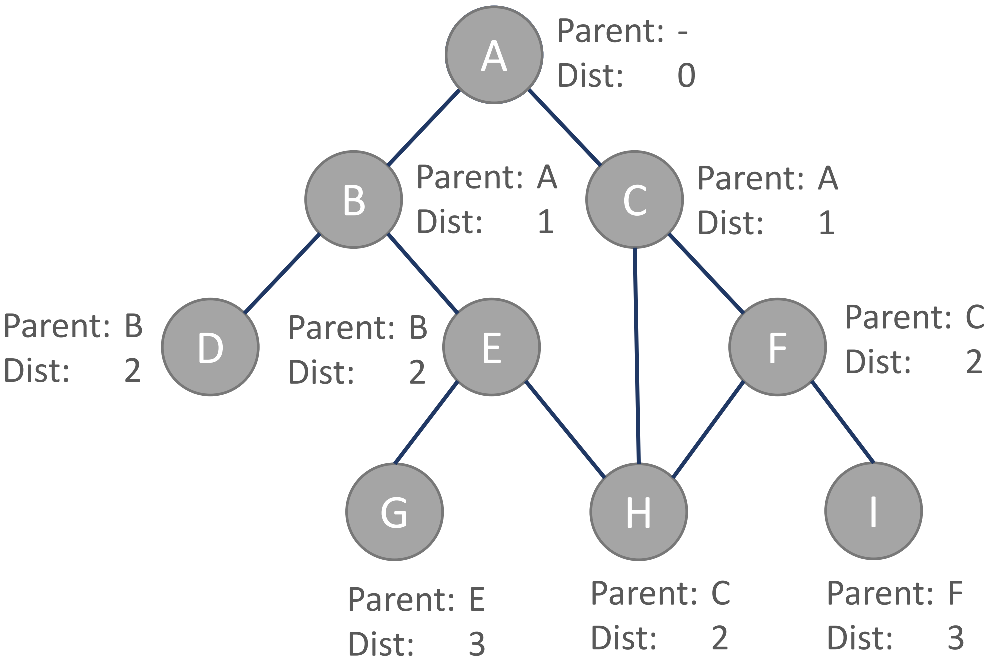
Current node: F

Visited nodes:

[A B C D E]

Breadth First Search Example (2)

Find a path from A to H.



Parent: F?
Dist: 3?

Keep current parent and distance!

Visit list:

[H G I]

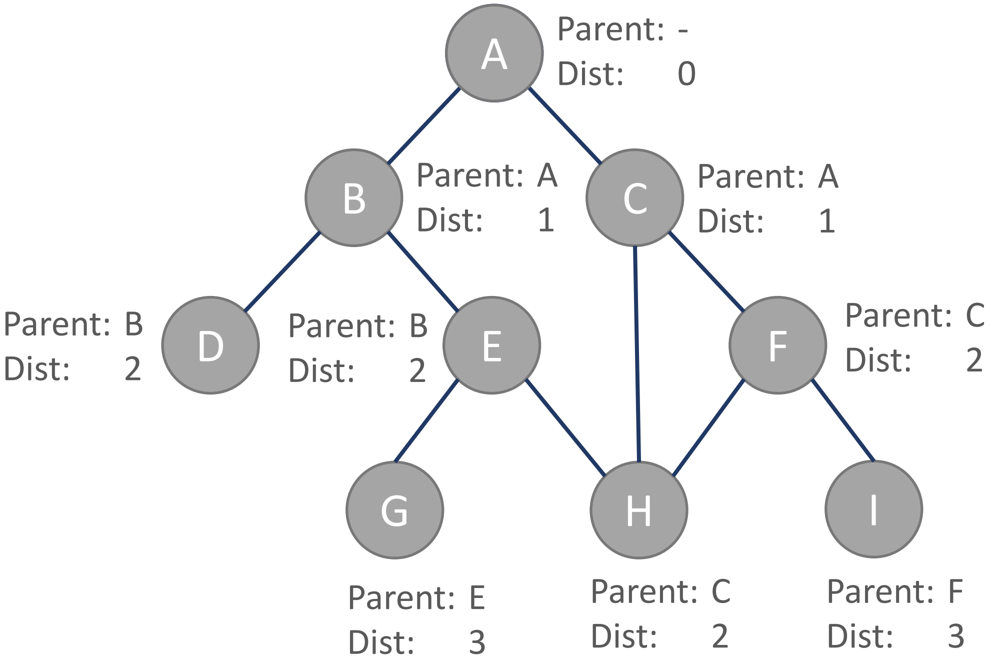
Current node: F

Visited nodes:

[A B C D E]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[G I]

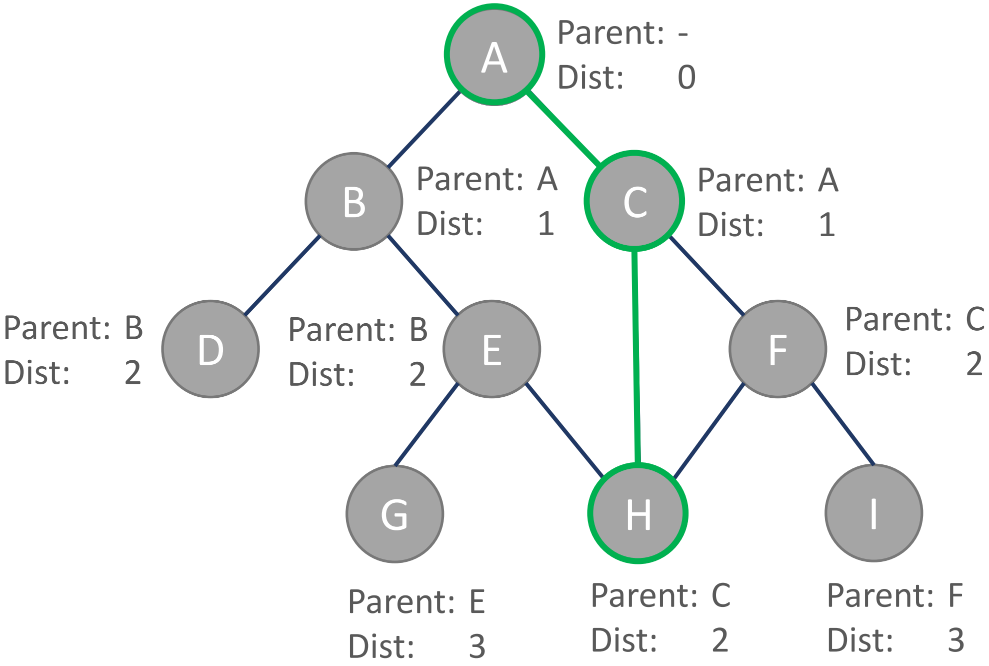
Current node: H Goal!

Visited nodes:

[A B C D E F]

Breadth First Search Example (2)

Find a path from A to H.



Visit list:

[G I]

Current node: H Goal!

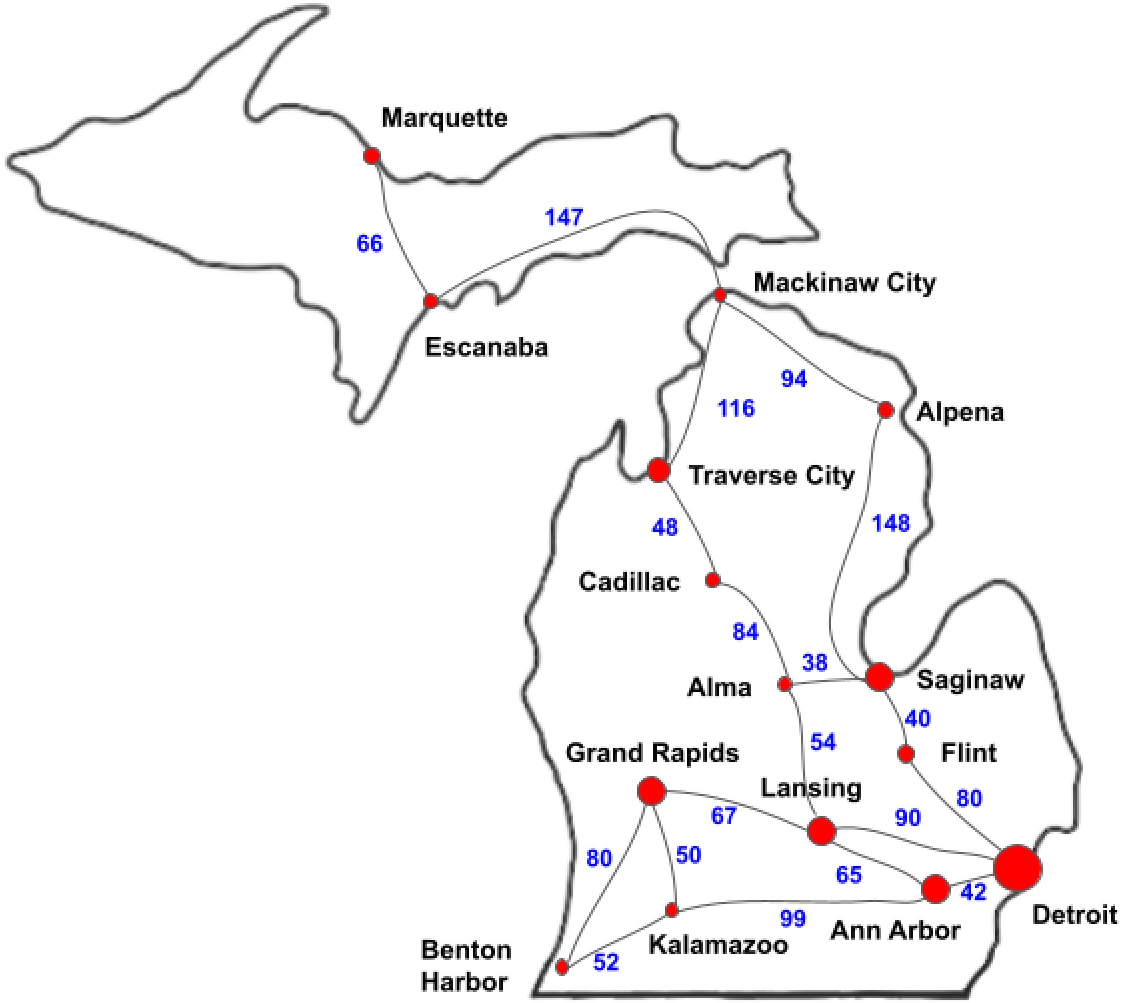
Visited nodes:

[A B C D E F]

Group Activity

What if the edges have different lengths?

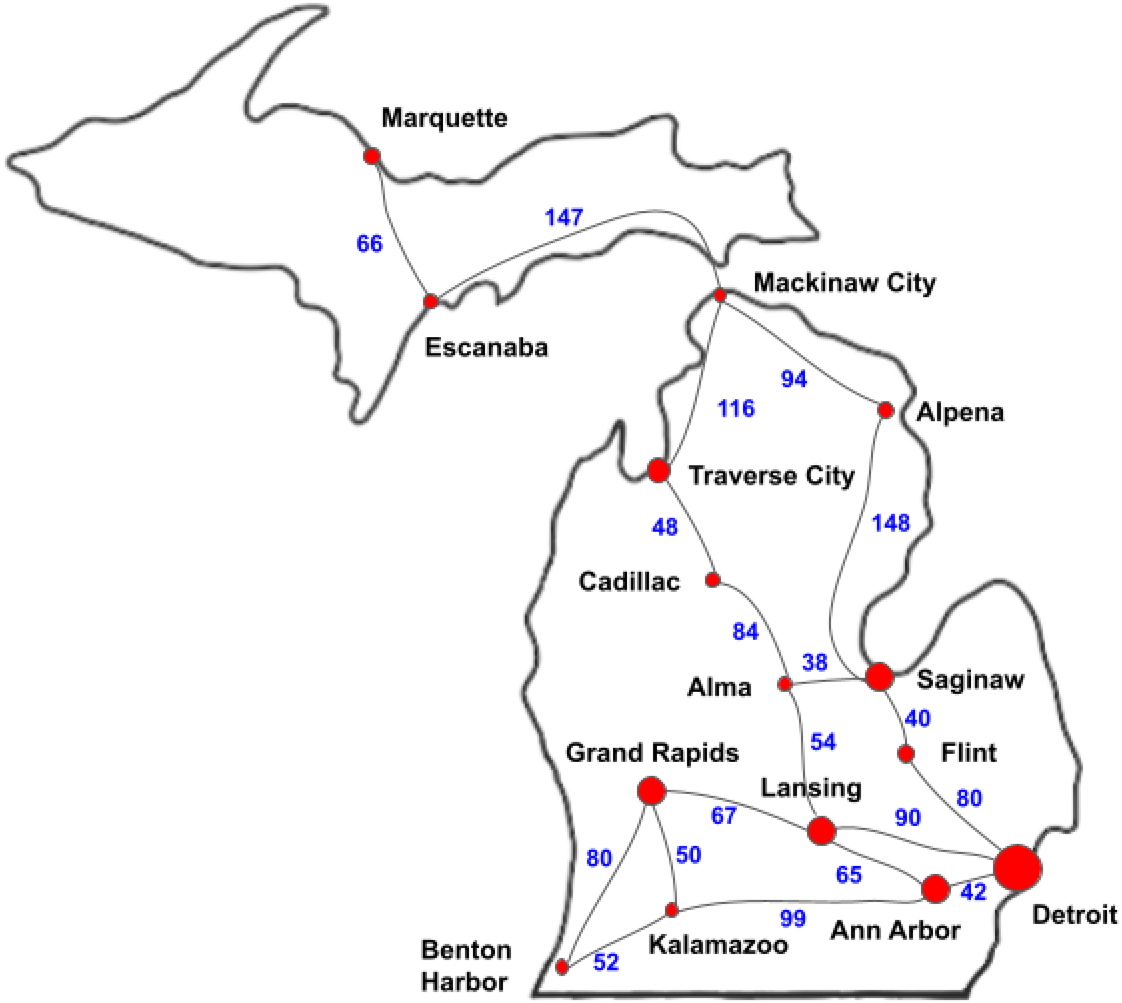
1. Add start node to visit list. Set distance to zero.
2. Pop the first node from the front of visit list. Set as current node.
3. If current node is goal, go to 6.
4. Add each unvisited neighbor of current to back of visit list.
5. Set parent of each neighbor to current, set distance to current distance + edge length. Go to 2.
6. Trace path backwards through parents.



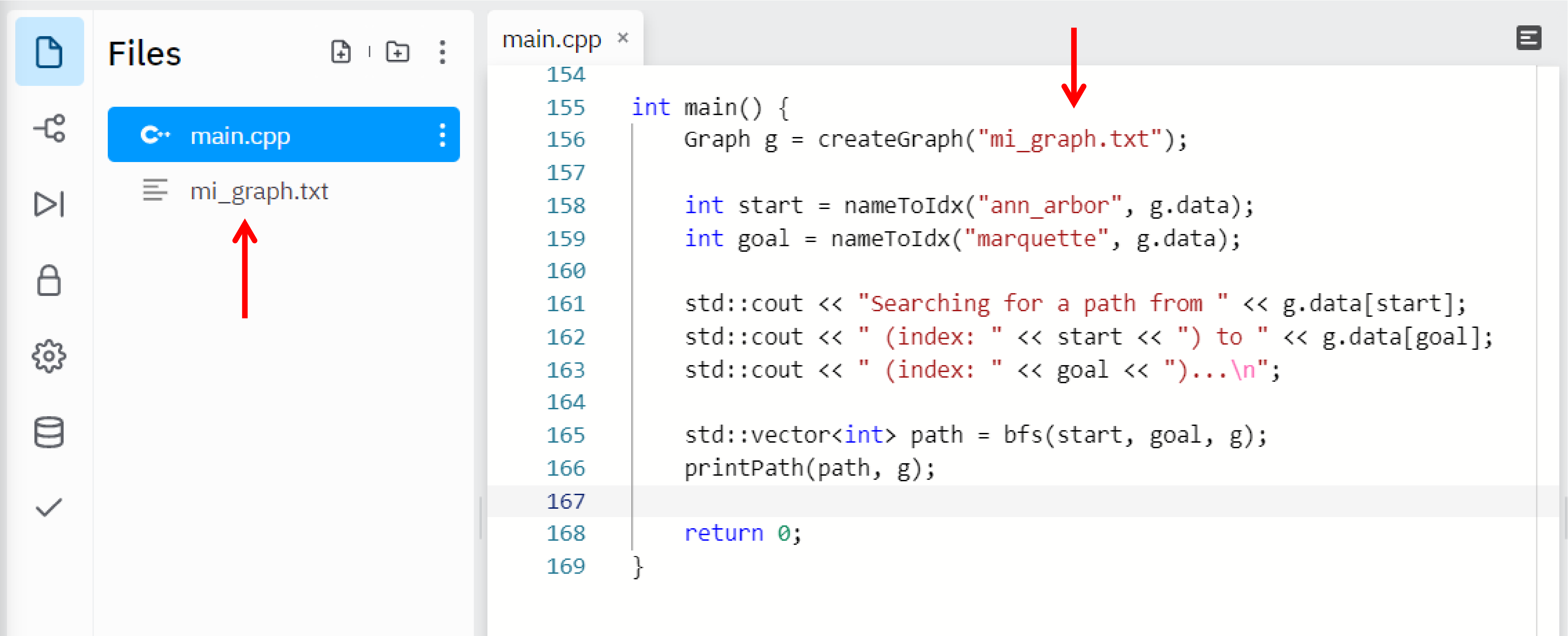
Group Activity

Find a path from Ann Arbor to Marquette.

- What is the length of the path?
- Is this the shortest path?
- In what order are the nodes visited?



BFS in C++



```cpp

int main() {
    Graph g = createGraph("mi_graph.txt");

    int start = nameToIdx("ann_arbor", g.data);
    int goal = nameToIdx("marquette", g.data);

    std::cout << "Searching for a path from " << g.data[start];
    std::cout << " (index: " << start << ") to " << g.data[goal];
    std::cout << " (index: " << goal << ")...\n";

    std::vector<int> path = bfs(start, goal, g);
    printPath(path, g);

    return 0;
}
```

BFS in C++

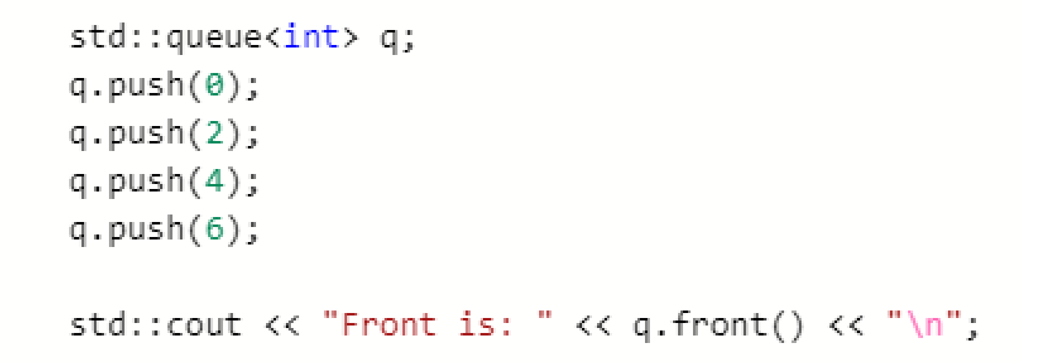
Things to consider:

- How can we represent and store the node data?
- How do we create a visit list?

BFS in C++: Queues

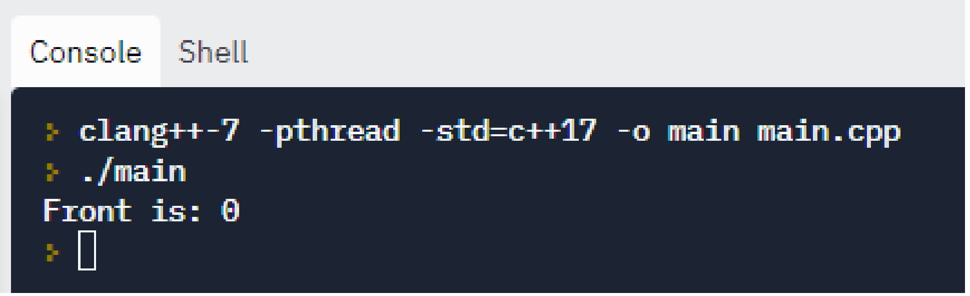
In BFS, we can use a queue for our visit list.

With a queue, we add new elements to the end of the list, and pop elements off the front of the list (first in, first out)



```
std::queue<int> q;
q.push(0);
q.push(2);
q.push(4);
q.push(6);

std::cout << "Front is: " << q.front() << "\n";
```

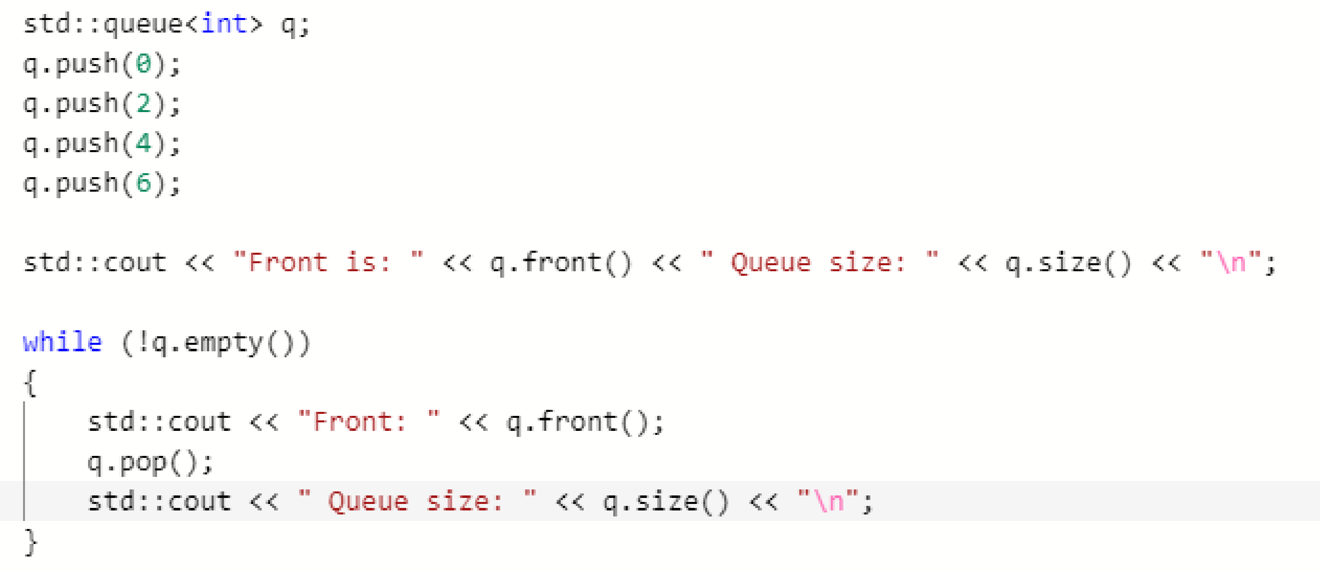


```
› clang++-7 -pthread -std=c++17 -o main main.cpp
› ./main
Front is: 0
› 
```

BFS in C++: Queues

In BFS, we can use a queue for our visit list.

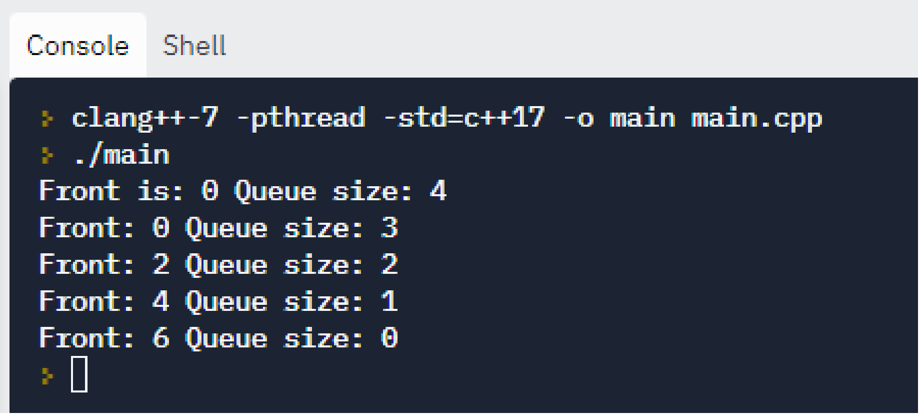
With a queue, we add new elements to the end of the list, and pop elements off the front of the list (first in, first out)



```
std::queue<int> q;
q.push(0);
q.push(2);
q.push(4);
q.push(6);

std::cout << "Front is: " << q.front() << " Queue size: " << q.size() << "\n";

while (!q.empty())
{
    std::cout << "Front: " << q.front();
    q.pop();
    std::cout << " Queue size: " << q.size() << "\n";
}
```



```
› clang++-7 -pthread -std=c++17 -o main main.cpp
› ./main
Front is: 0 Queue size: 4
Front: 0 Queue size: 3
Front: 2 Queue size: 2
Front: 4 Queue size: 1
Front: 6 Queue size: 0
› 
```