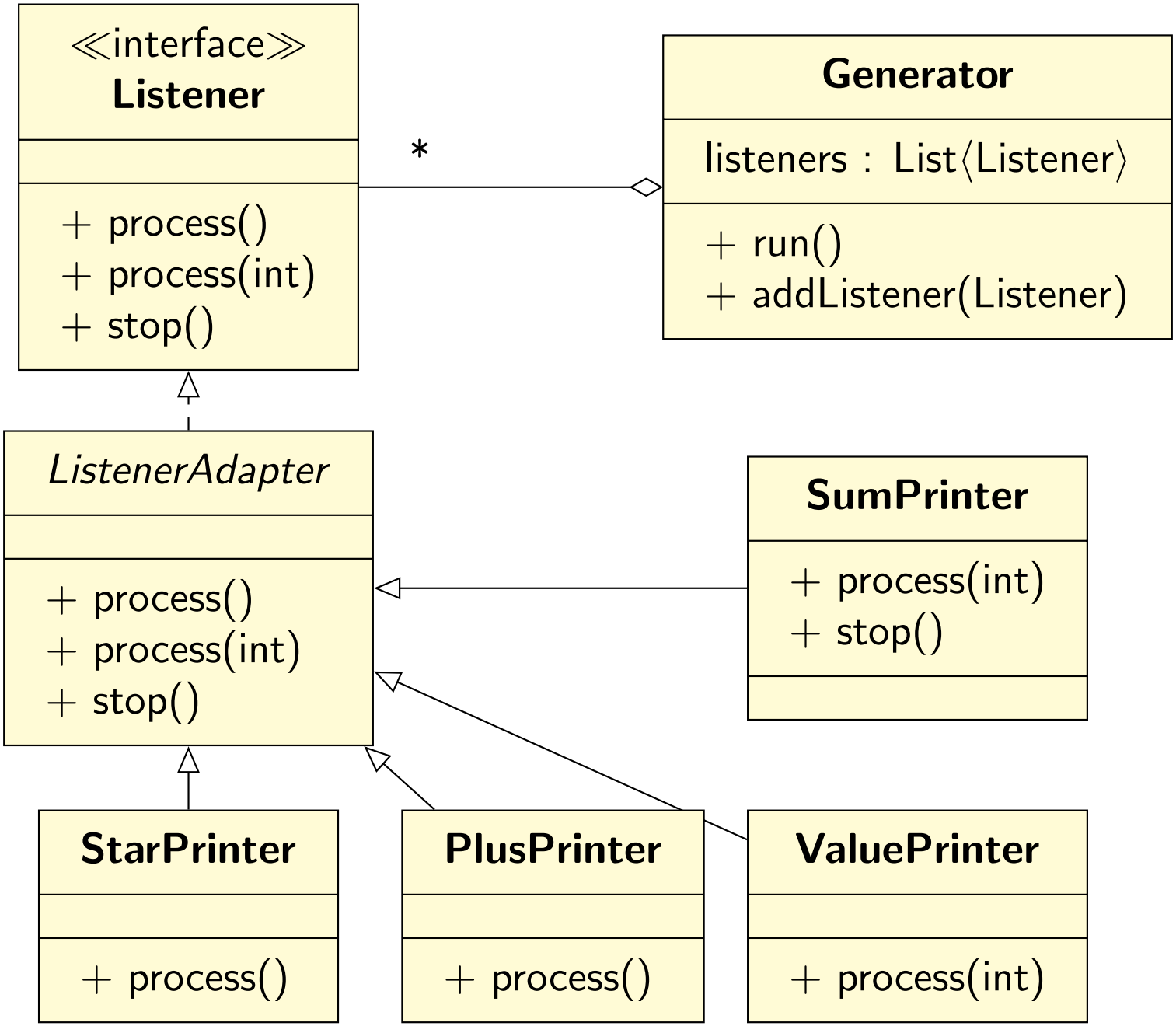
Listen

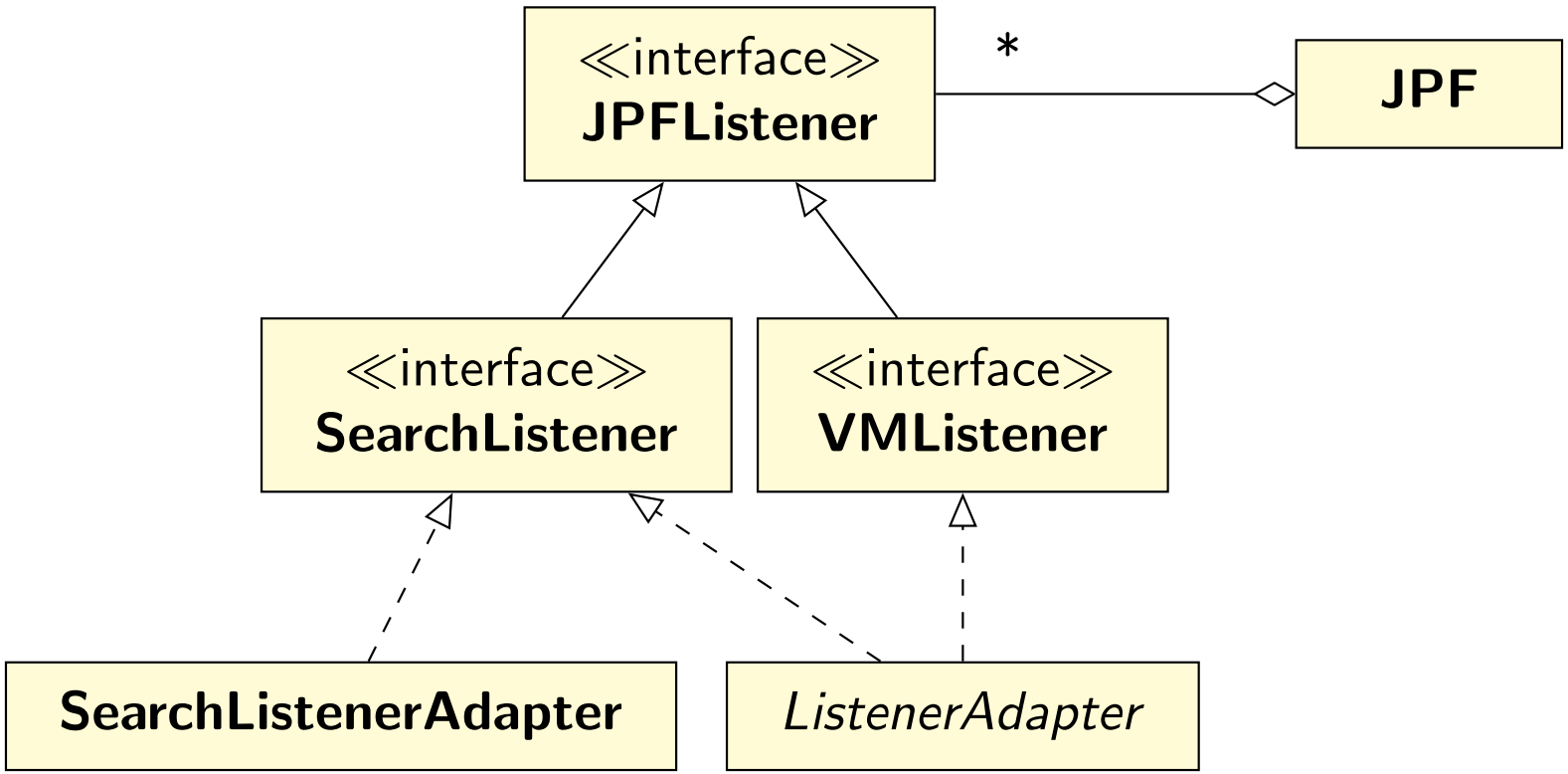
EECS 4315

www.eecs.yorku.ca/course/4315/

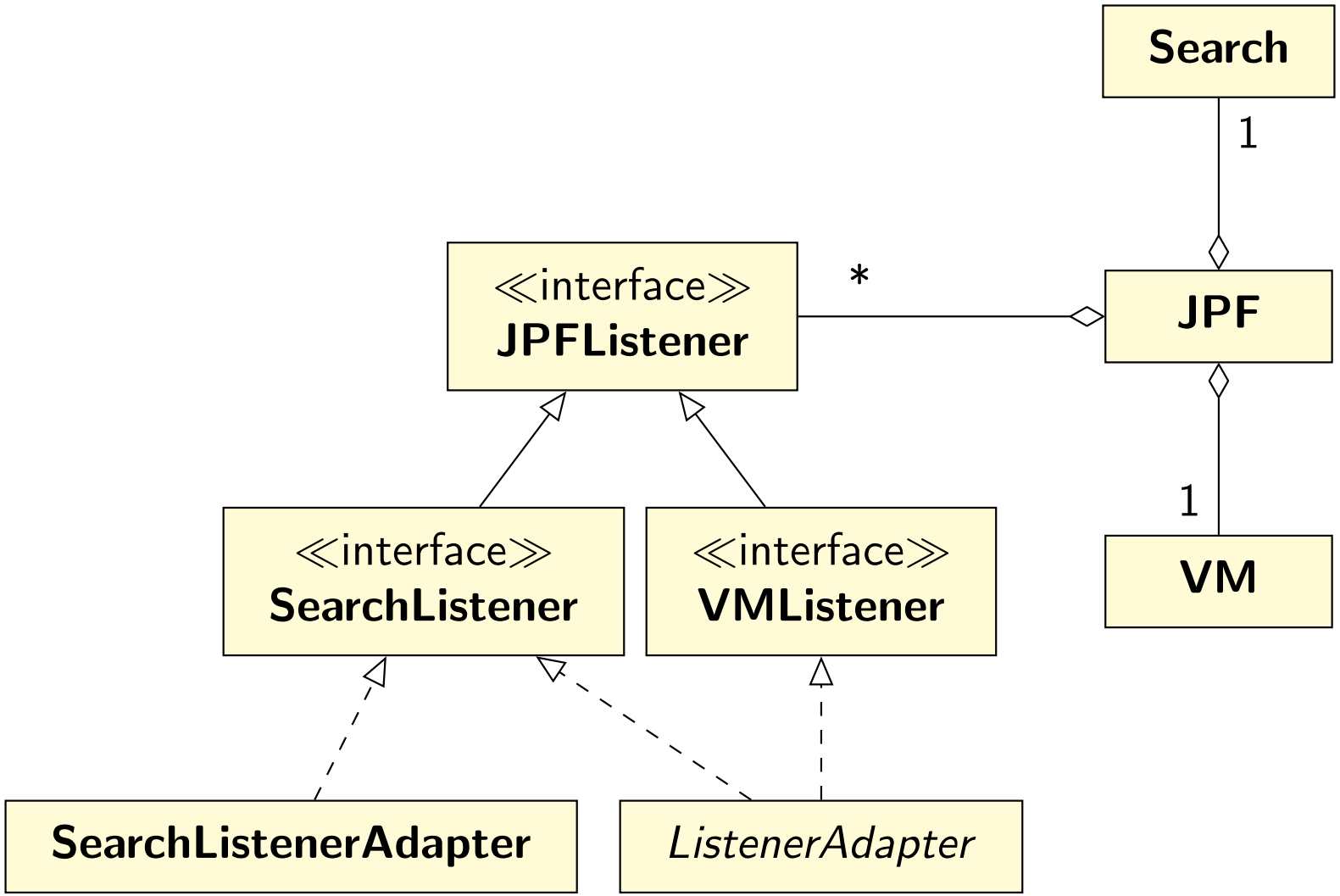
Generator and listeners



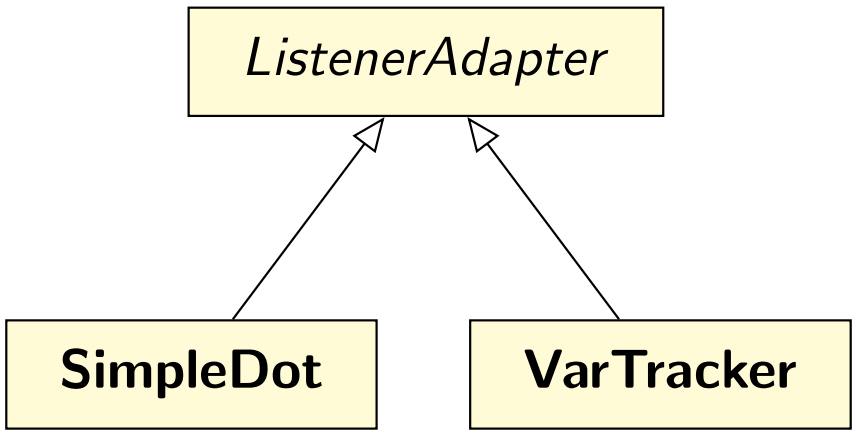
JPF and listeners



JPF and listeners



JPF and listeners



JPFListener

The interface `JPFListener` is empty.

JPFListener

The interface `JPFListener` is empty.

Question

Why introduce an empty interface?

JPFListener

The interface `JPFListener` is empty.

Question

Why introduce an empty interface?

Answer

JPF has a collection of `JPFListener`s, some can be `SearchListener`s and others can be `VMListener`s.

SearchListener

```
public interface SearchListener extends JPFListener {
  void stateAdvanced(Search search);
  void stateBacktracked(Search search);
  void stateProcessed(Search search);
  void statePurged(Search search);
  void stateRestored(Search search);
  void stateStored(Search search);

  void searchProbed(Search search);

  void propertyViolated(Search search);

  void searchConstraintHit(Search search);

  void searchStarted(Search search);
  void searchFinished(Search search);
}
```

SearchListener

The API of the `SearchListener` interface can be found here.

The Javadoc was added by Yahya Ismail as last year's EECS 4315 project.

Search events

Implement a listener which prints the sequence of events that have been triggered during JPF's search. For each event, it prints the sort of event and the ID of the state of the search in which the event occurs.

Search events

Question

Which methods of the `SearchListener` interface are relevant?

Search events

Question

Which methods of the `SearchListener` interface are relevant?

Answer

All of them.

Search events

Question

The method `stateAdvanced` has a parameter of type `Search`. Does the interface `Listener` of the previous lecture contain a similar type of method? If so, which one?

Search events

Question

The method `stateAdvanced` has a parameter of type `Search`. Does the interface `Listener` of the previous lecture contain a similar type of method? If so, which one?

Answer

Yes, the method `process` has a parameter of type `int`.

Search events

Question

The method `stateAdvanced` has a parameter of type `Search`. Does the interface `Listener` of the previous lecture contain a similar type of method? If so, which one?

Answer

Yes, the method `process` has a parameter of type `int`.

Question

What is the role of the parameter?

Search events

Question

The method `stateAdvanced` has a parameter of type `Search`. Does the interface `Listener` of the previous lecture contain a similar type of method? If so, which one?

Answer

Yes, the method `process` has a parameter of type `int`.

Question

What is the role of the parameter?

Answer

It provides information about the event: the integer produced by the generator and the current state of JPF's search.

Search events

Implement a listener which prints the sequence of events that have been triggered during JPF's search. For each event, it prints the sort of event and the ID of the state of the search in which the event occurs.

Question

How to implement the method `searchAdvanced(Search search)`? The class `Search` has a method `getStateId` that returns the ID of the current state of the search.

Compiling a listener

```
3 import gov.nasa.jpf.search.Search;
4 import gov.nasa.jpf.search.SearchListener;
5
```

Compiling a listener

To compile the listener, make sure that `jpf.jar` is part of the classpath.

Using a listener

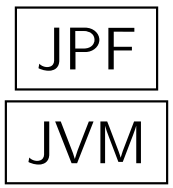
```
target = Traversal
classpath = /courses/4315/workspace/4315/bin
cg.enumerate_random = true
listener = listener.SearchEvents
```

Using a listener

```
target = Traversal
classpath = /courses/4315/workspace/4315/bin
cg.enumerate_random = true
listener = listener.SearchEvents
```

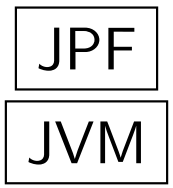
```
[SEVERE] JPF configuration error: class not found listener.SearchEvents
[SEVERE] JPF terminated
```

JPF as a JVM



- JPF is a JVM.
- Since JPF is written in Java, it runs on a JVM.

JPF as a JVM

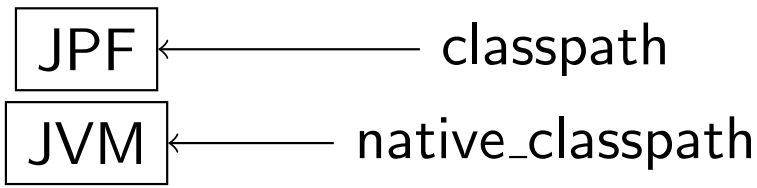


- JPF is a JVM.
- Since JPF is written in Java, it runs on a JVM.

- JPF model checks Java bytecode.
- JVM executes Java bytecode.

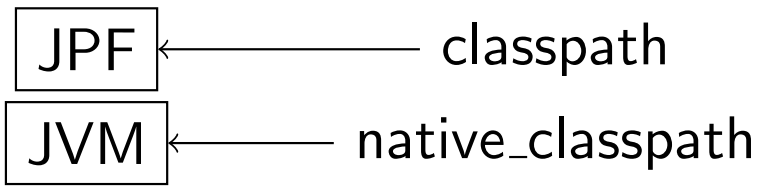
JPF as a JVM

Each JVM has a classpath which tells the JVM where to look for classes.



JPF as a JVM

Each JVM has a classpath which tells the JVM where to look for classes.



classpath of JPF: where JPF looks for classes to model check

native_classpath of JPF: where the JVM looks for classes to execute (as part of JPF)

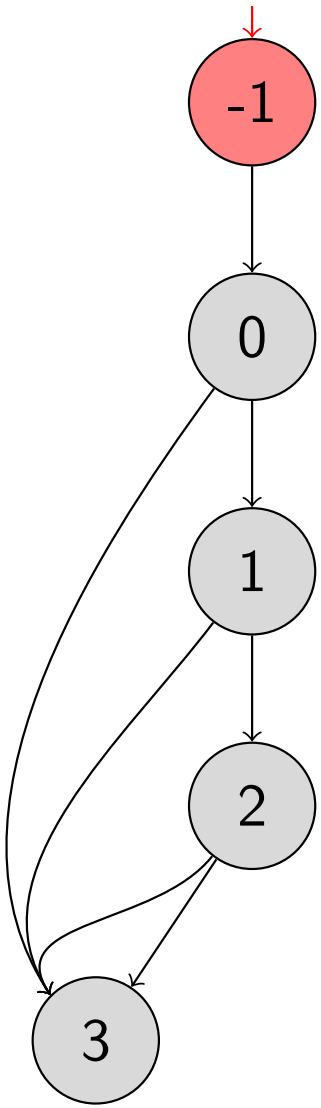
Using a listener

```
target = Traversal
classpath = /courses/4315/workspace/4315/bin
cg.enumerate_random = true
listener = listener.SearchEvents
native_classpath = /courses/4315/workspace/4315/bin
```

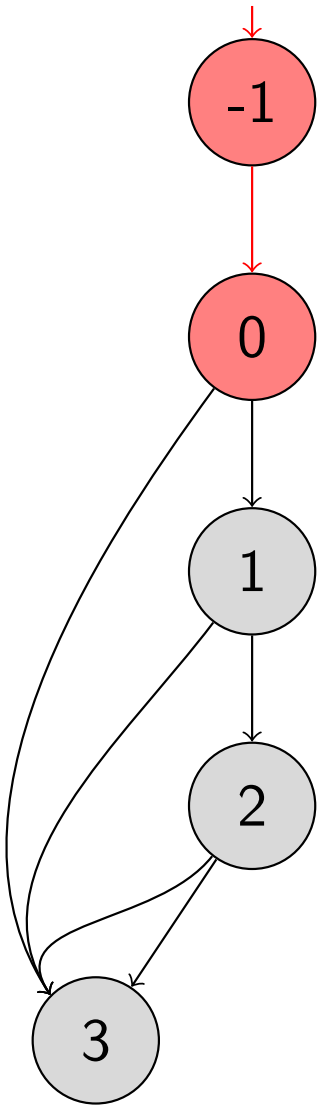
Traversal

```
Random random = new Random();
  System.out.println("0");
  if (random.nextBoolean()) {
    System.out.println("2");
  } else {
    System.out.println("1");
    if (random.nextBoolean()) {
      System.out.println("4");
    } else {
      System.out.println("3");
    if (random.nextBoolean()) {
      System.out.println("6");
    } else {
      System.out.println("5");
    }
  }
}
```

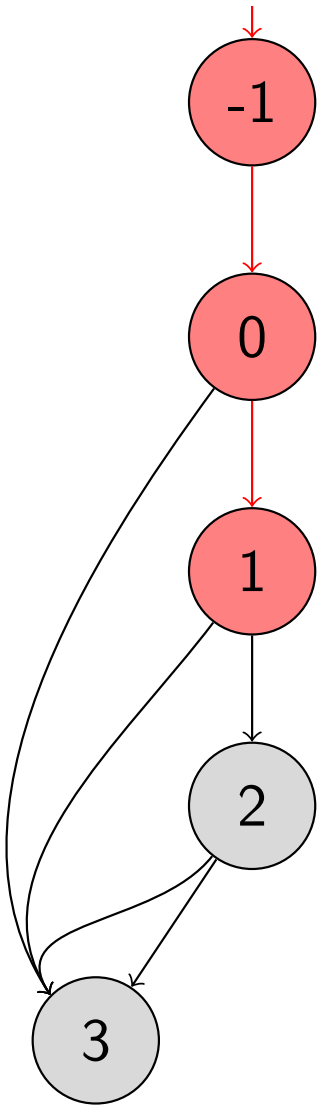
Traversal (DFS)



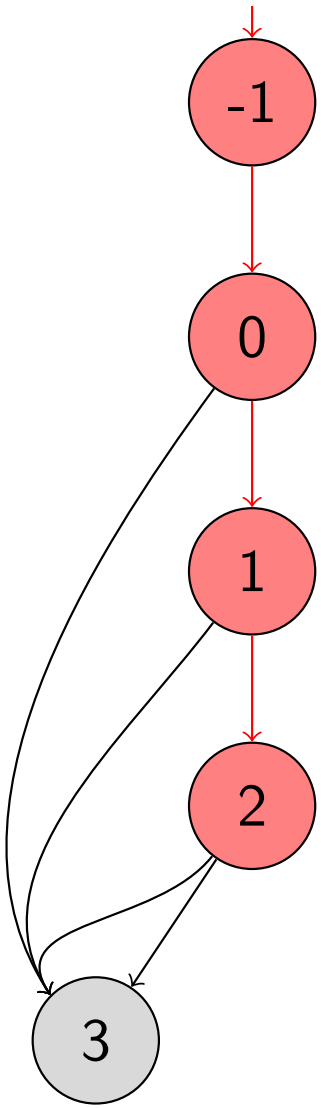
Traversal (DFS)



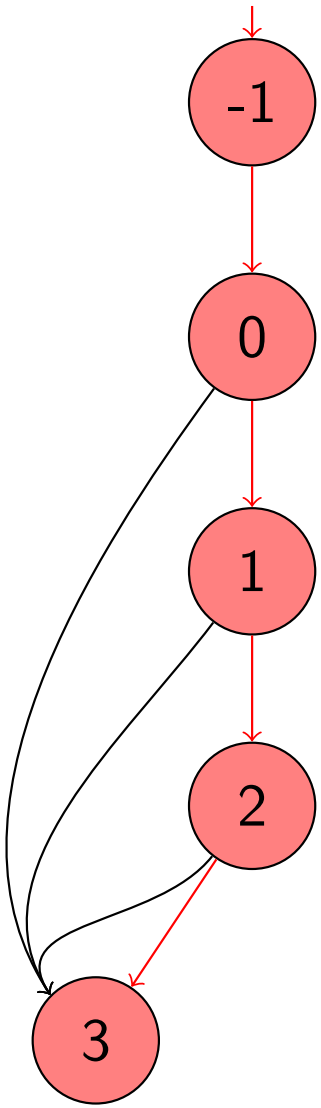
Traversal (DFS)



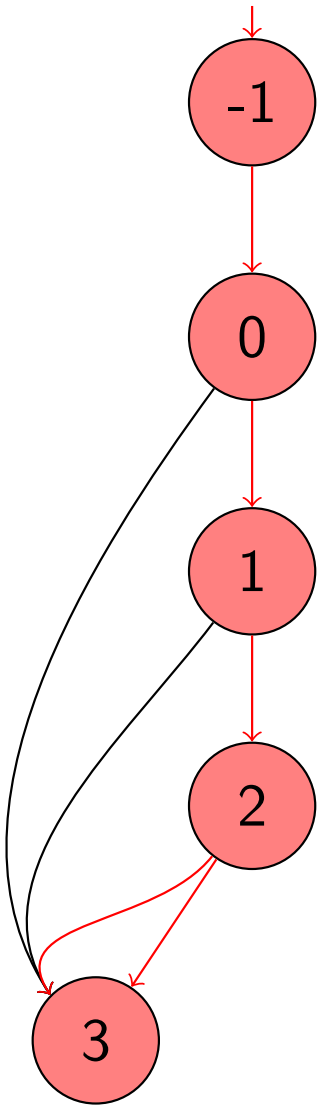
Traversal (DFS)



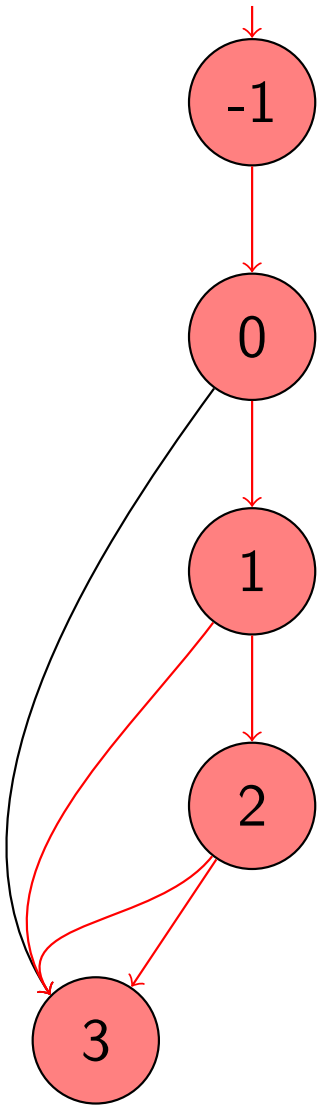
Traversal (DFS)



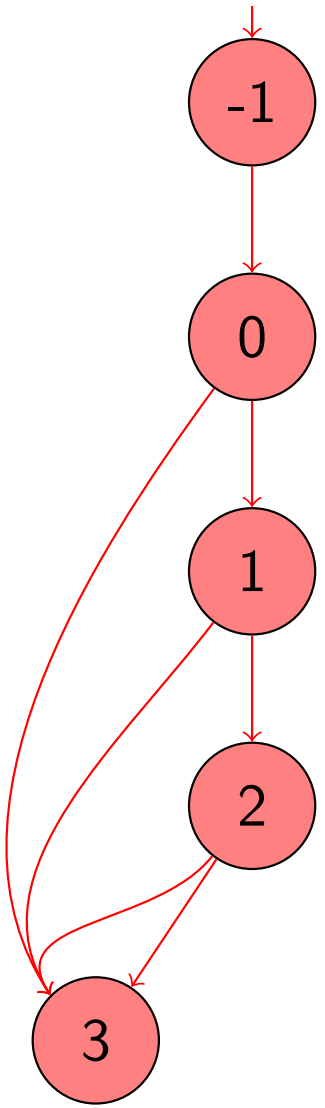
Traversal (DFS)



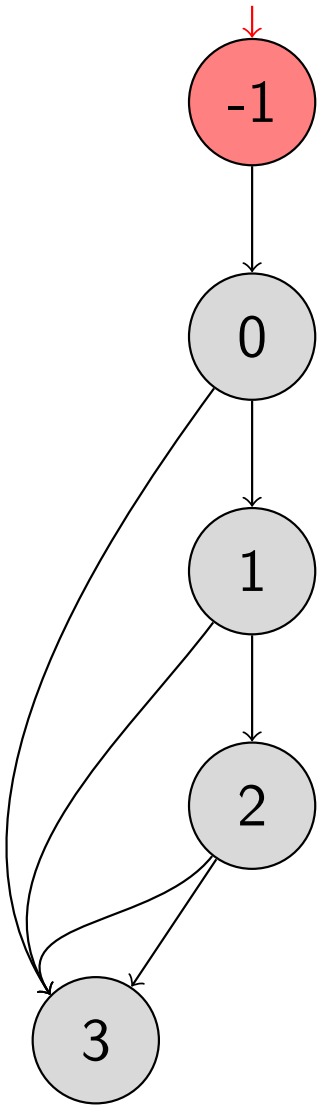
Traversal (DFS)



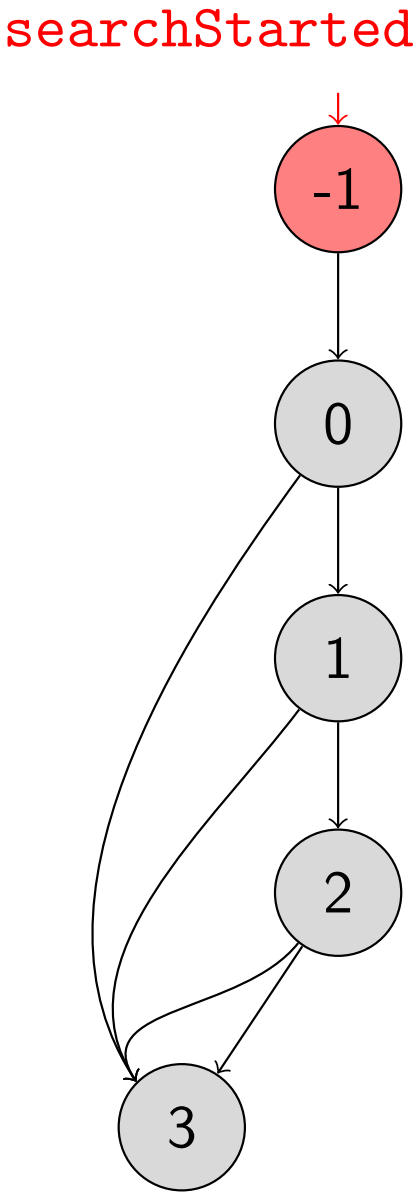
Traversal (DFS)



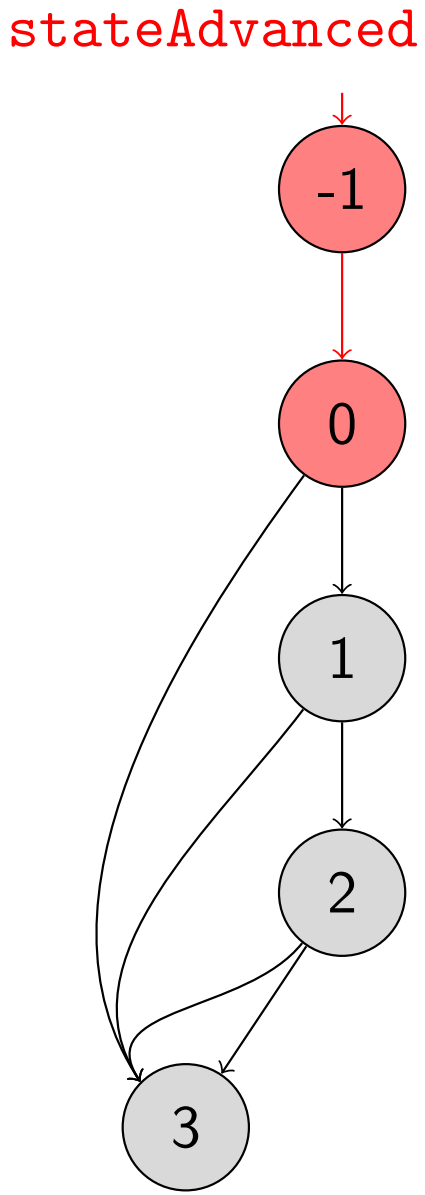
Search events (DFS)



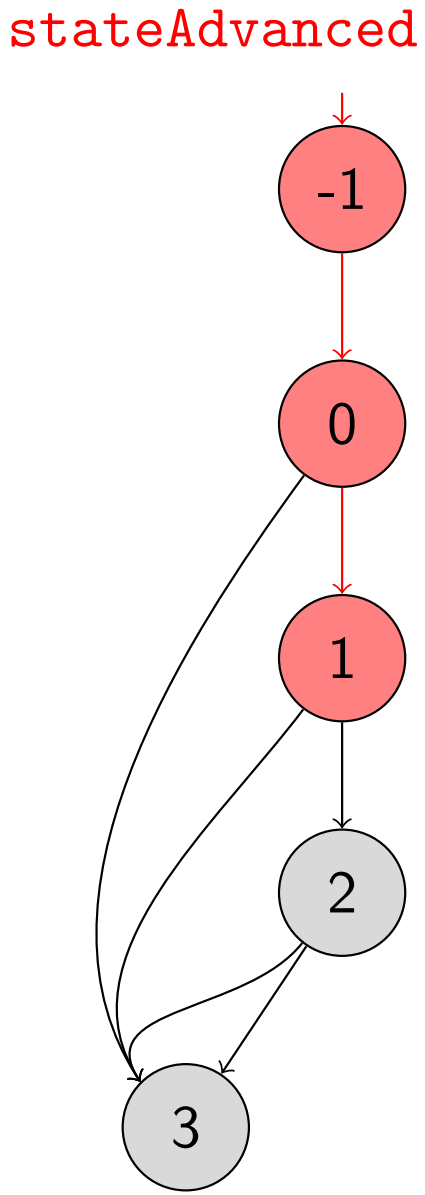
Search events (DFS)



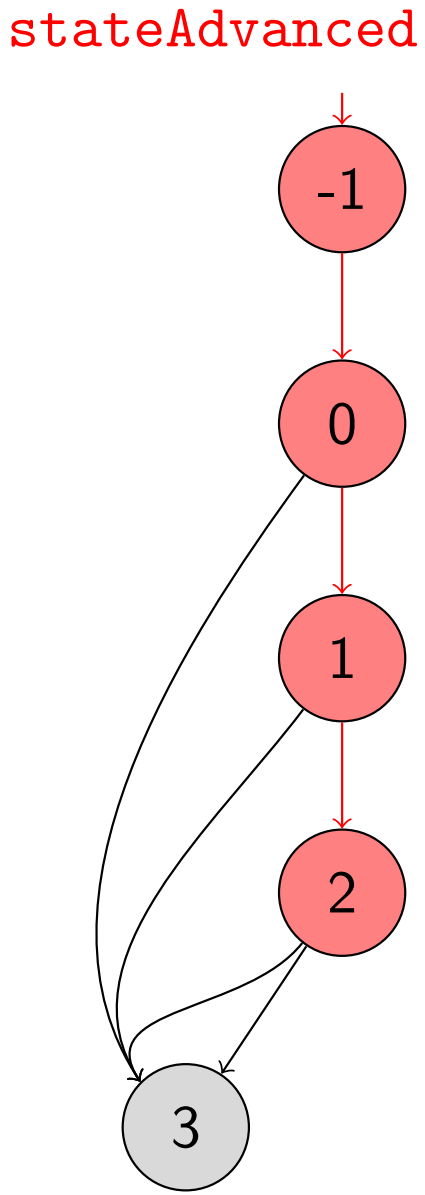
Search events (DFS)



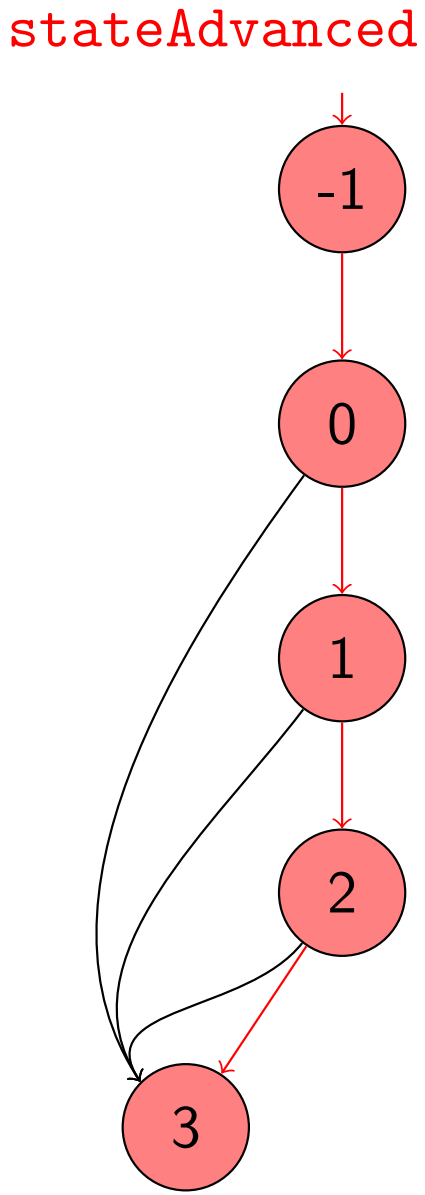
Search events (DFS)



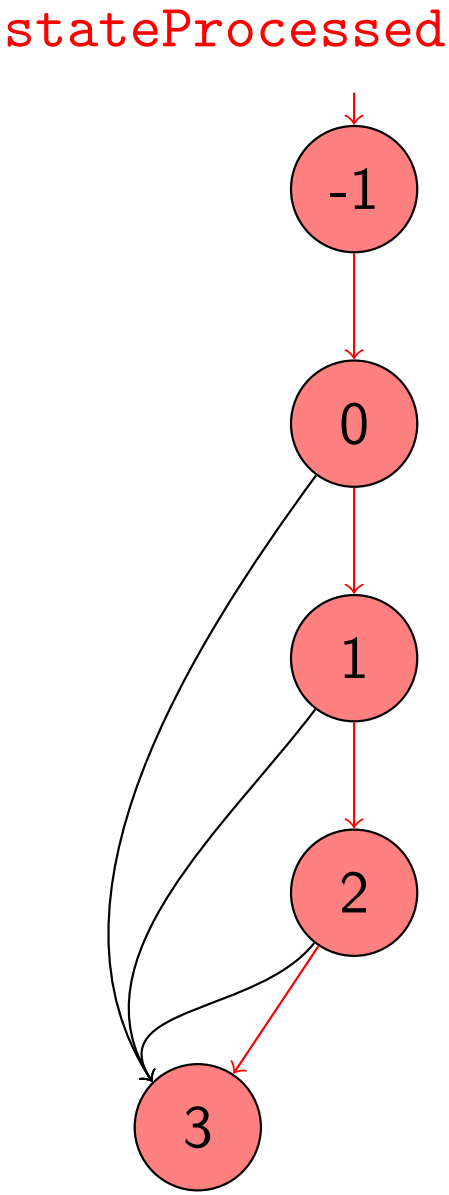
Search events (DFS)



Search events (DFS)

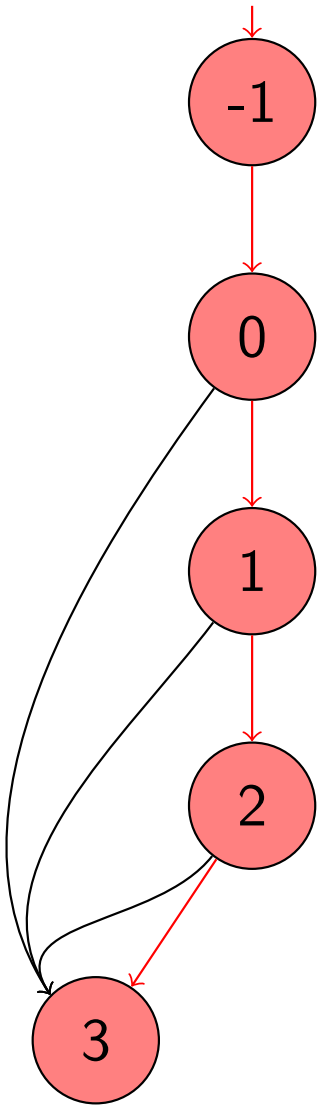


Search events (DFS)

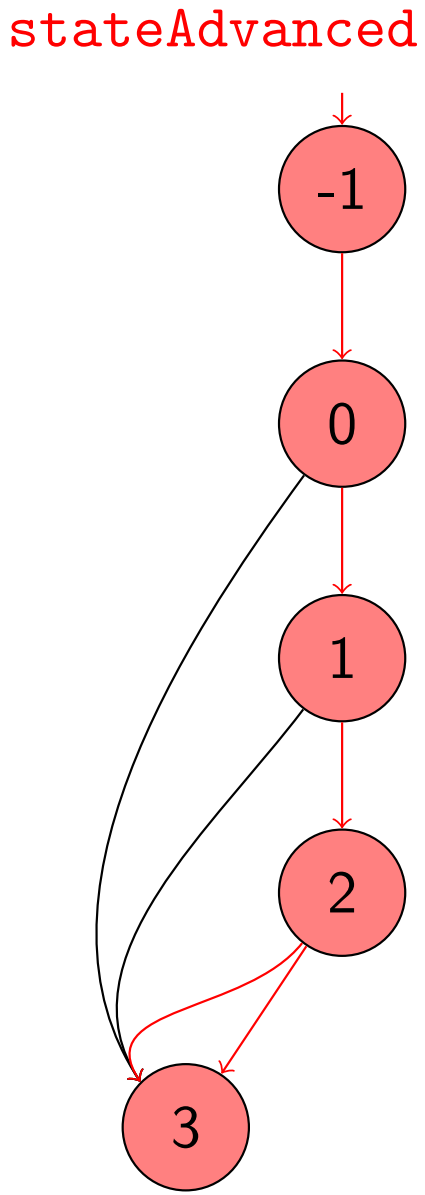


Search events (DFS)

`stateBacktracked`

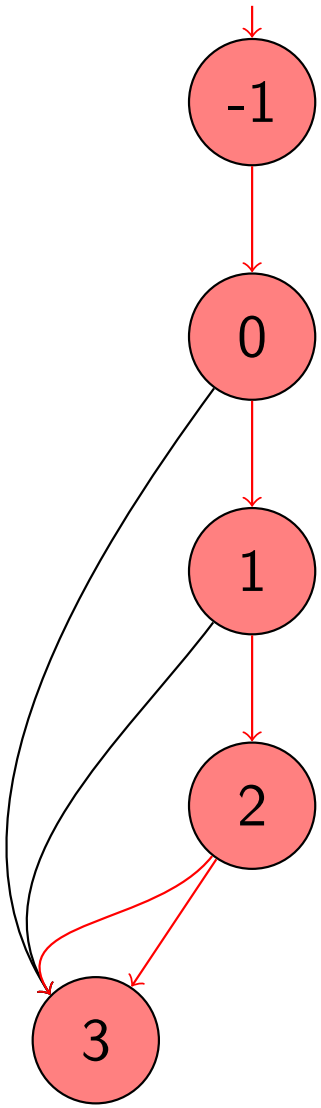


Search events (DFS)

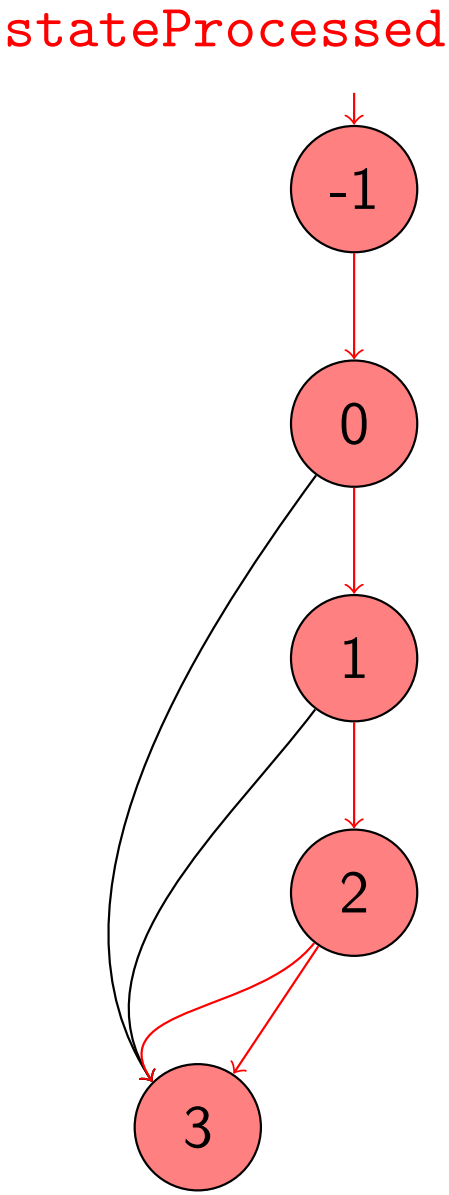


Search events (DFS)

`stateBacktracked`

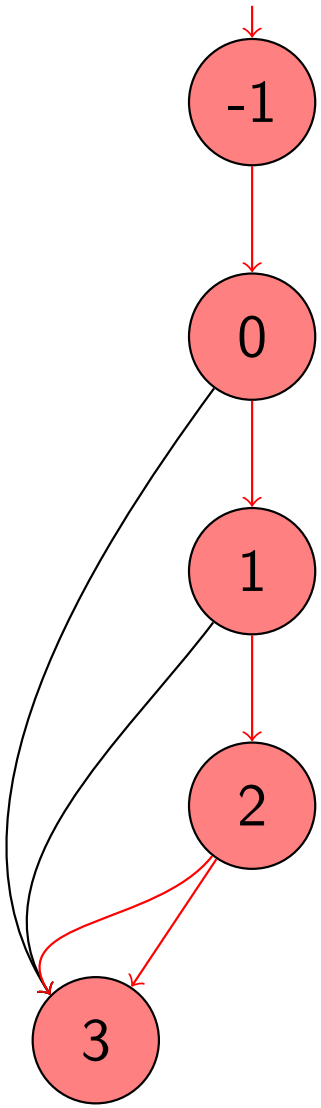


Search events (DFS)

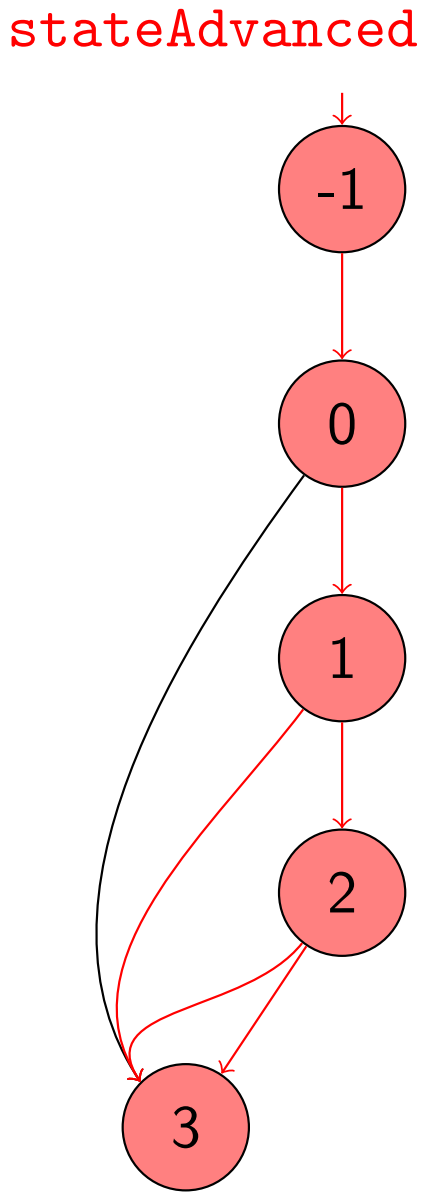


Search events (DFS)

`stateBacktracked`

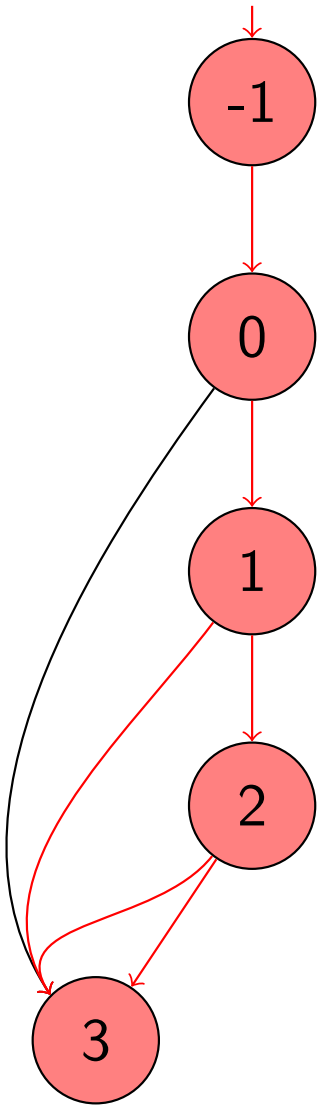


Search events (DFS)

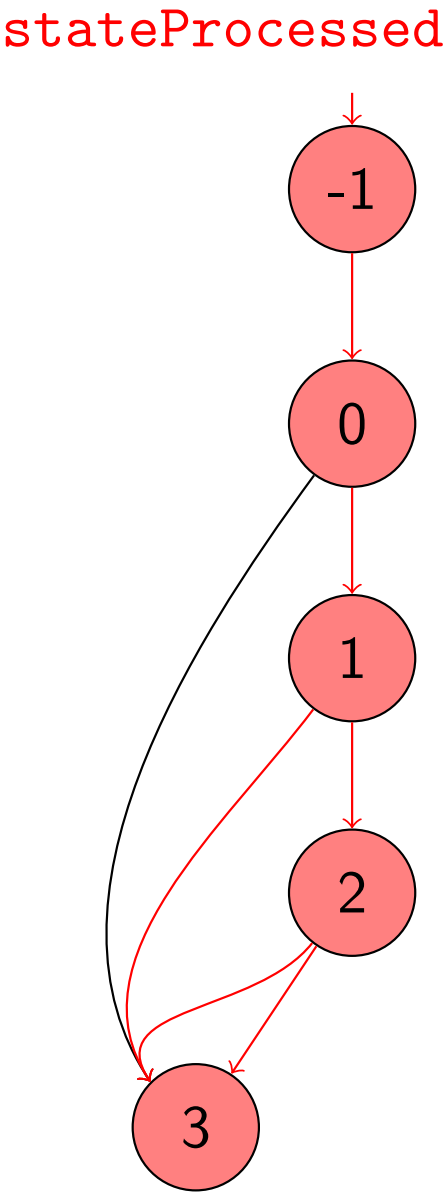


Search events (DFS)

`stateBacktracked`

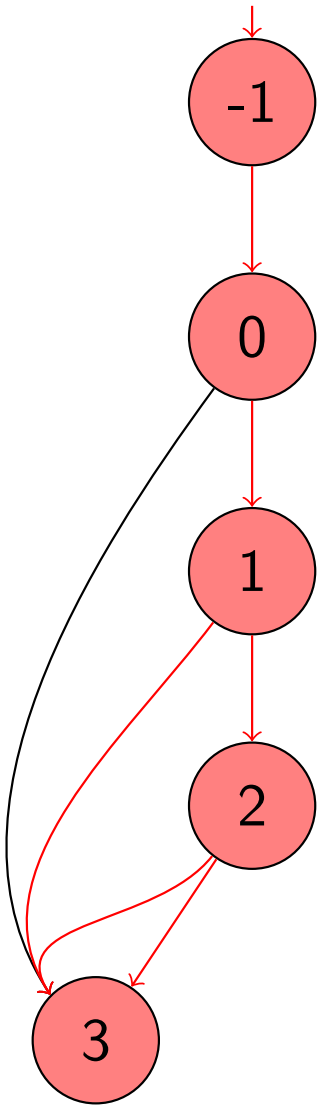


Search events (DFS)

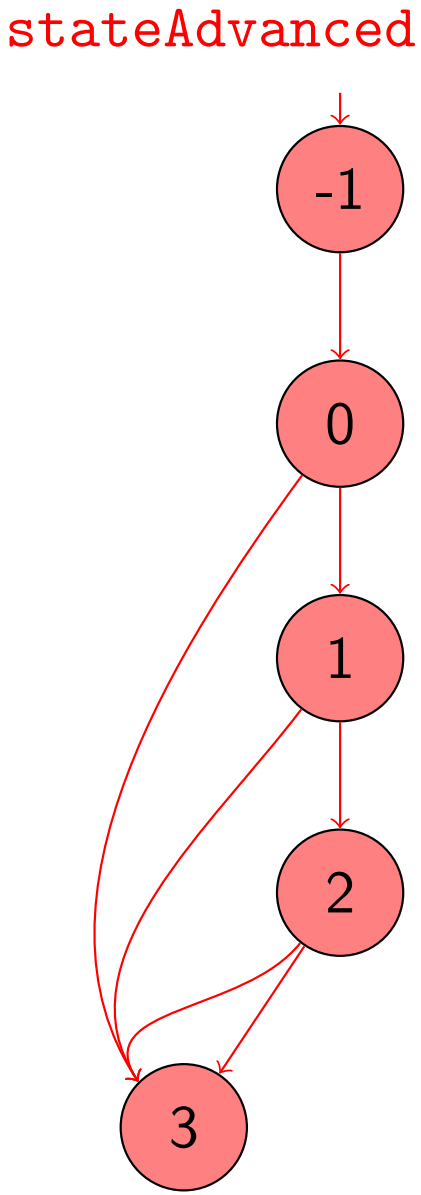


Search events (DFS)

`stateBacktracked`

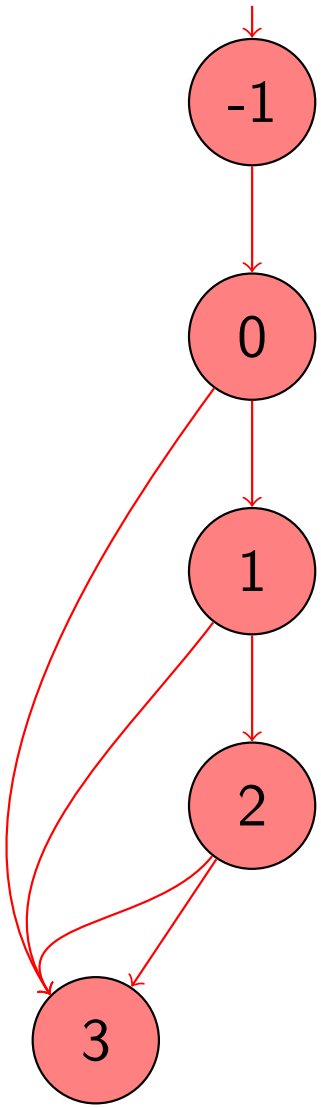


Search events (DFS)

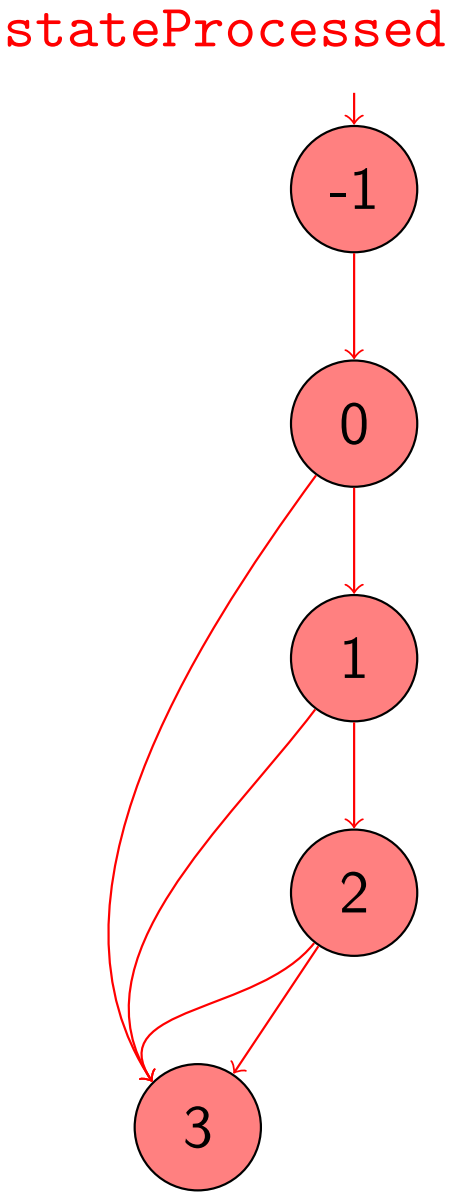


Search events (DFS)

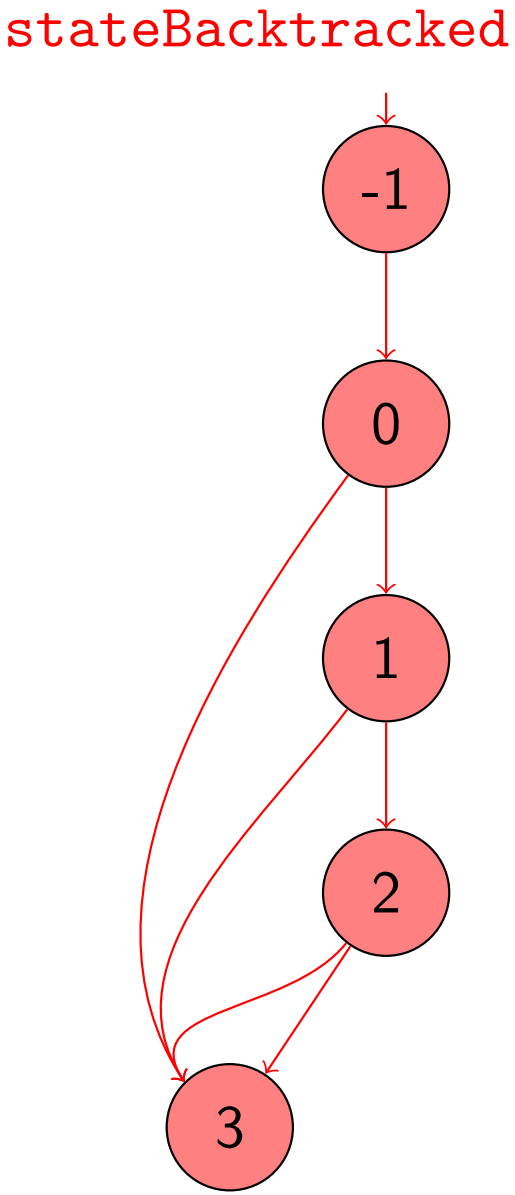
`stateBacktracked`



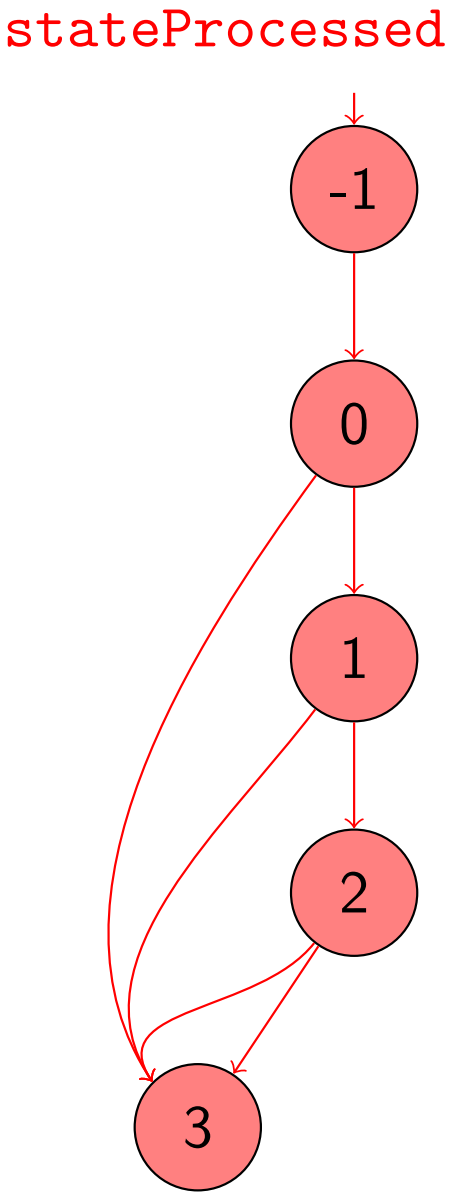
Search events (DFS)



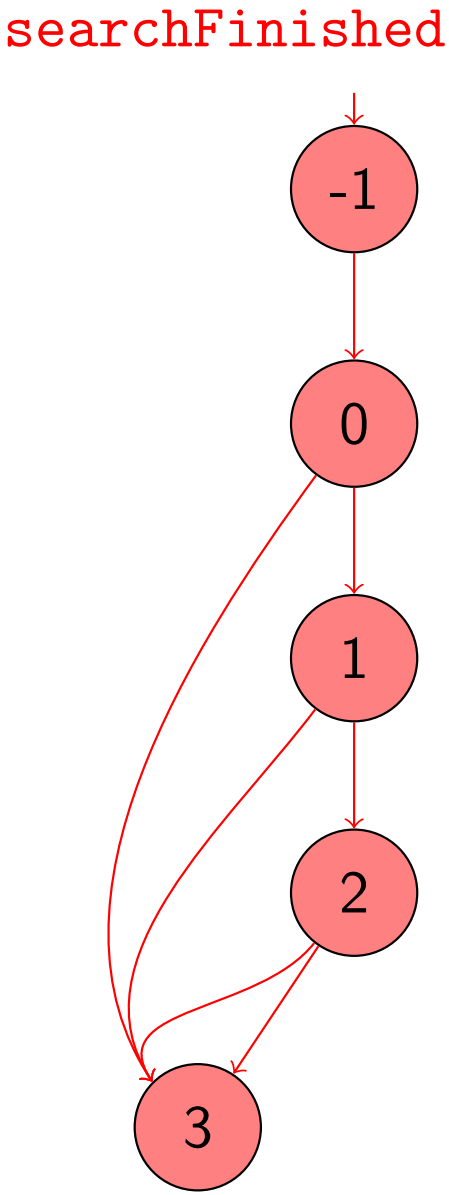
Search events (DFS)



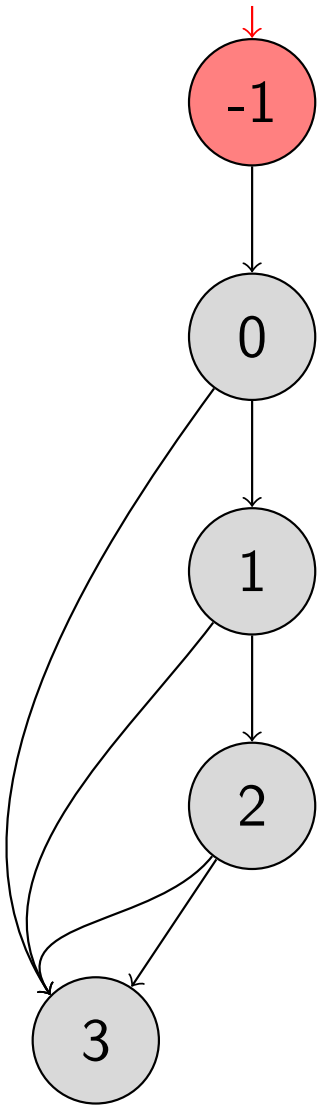
Search events (DFS)



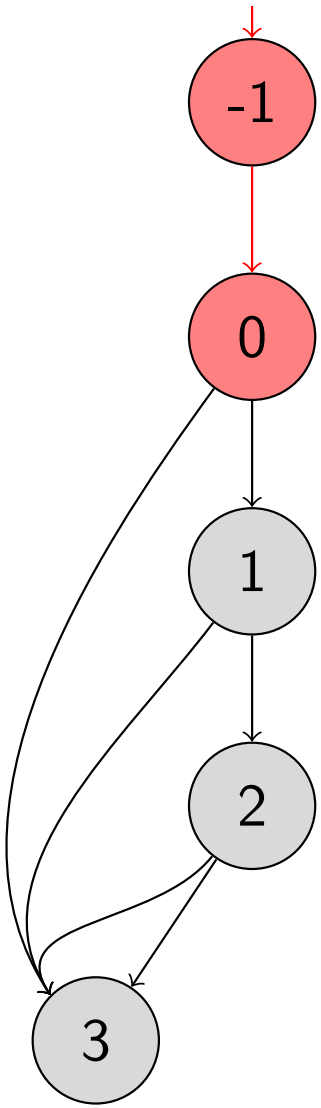
Search events (DFS)



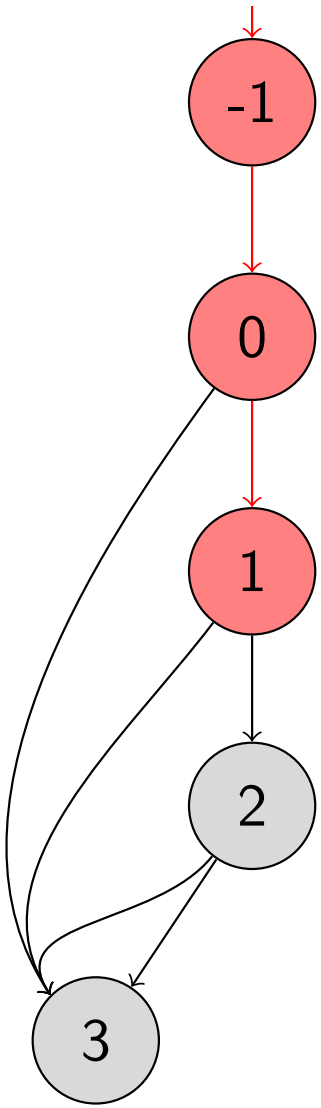
Traversal (BFS)



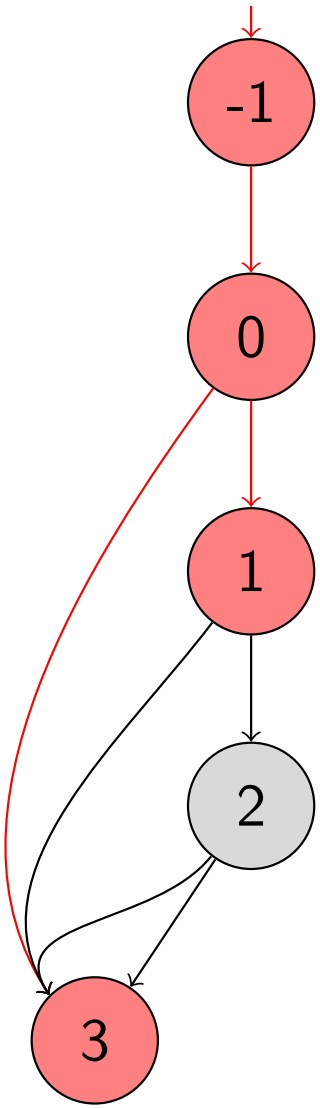
Traversal (BFS)



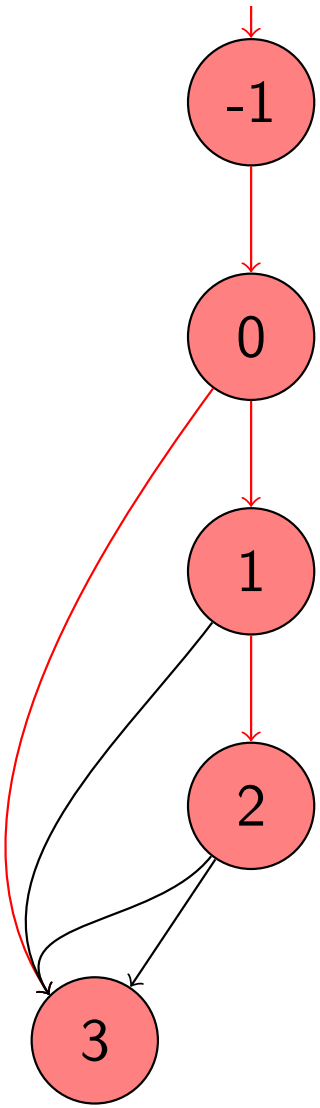
Traversal (BFS)



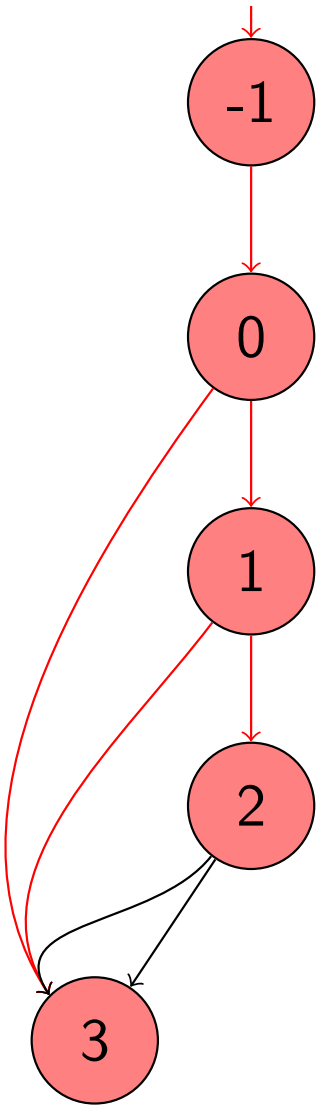
Traversal (BFS)



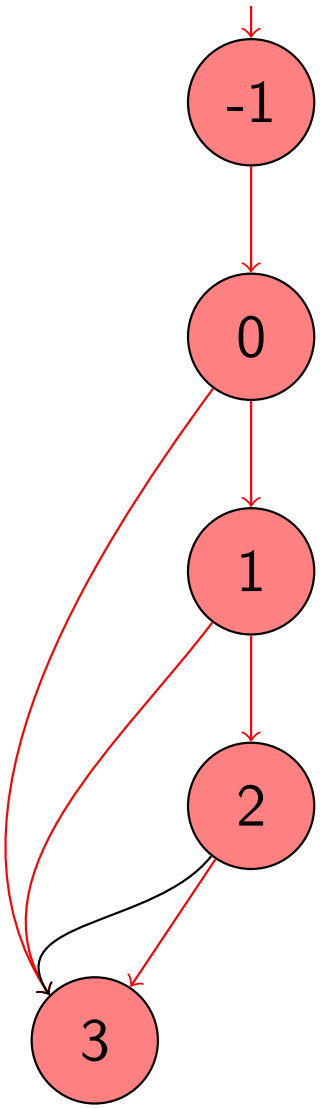
Traversal (BFS)



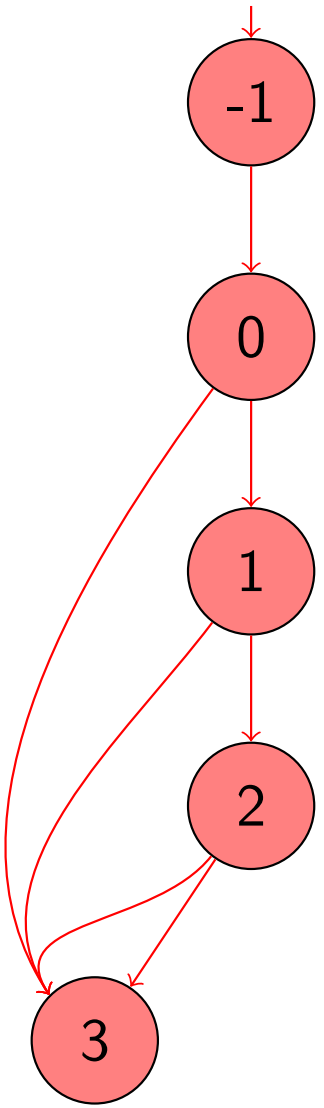
Traversal (BFS)



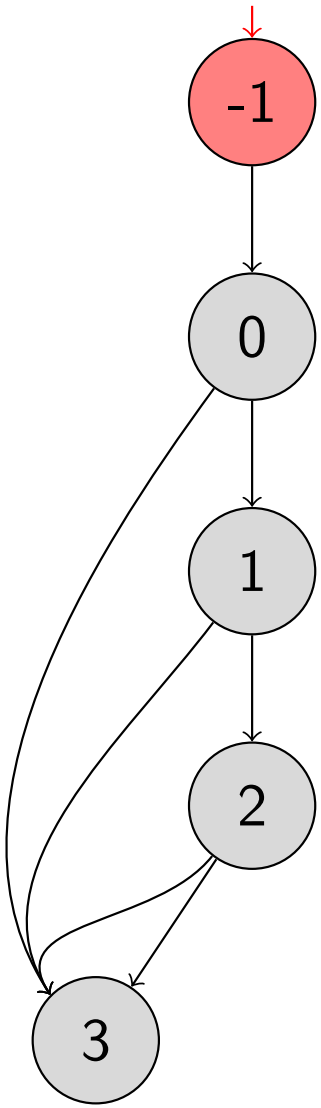
Traversal (BFS)



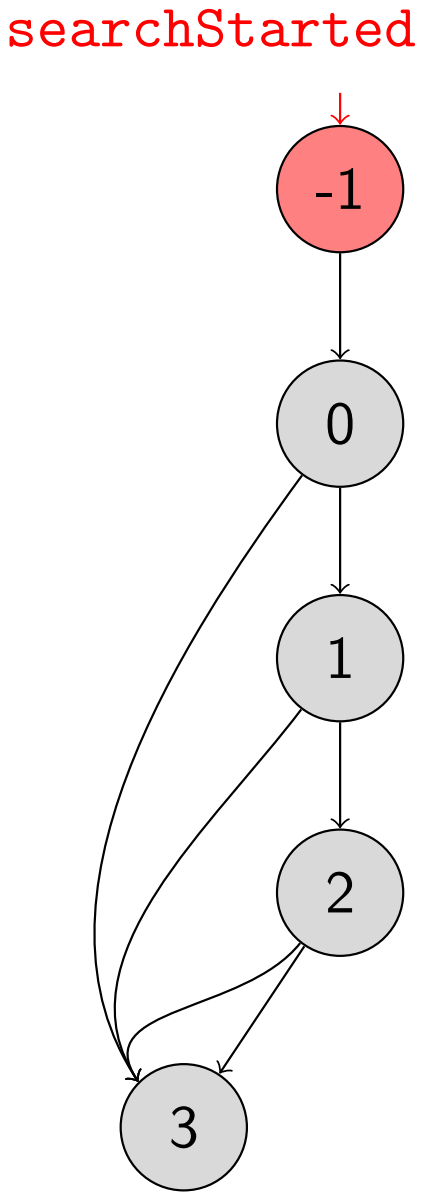
Traversal (BFS)



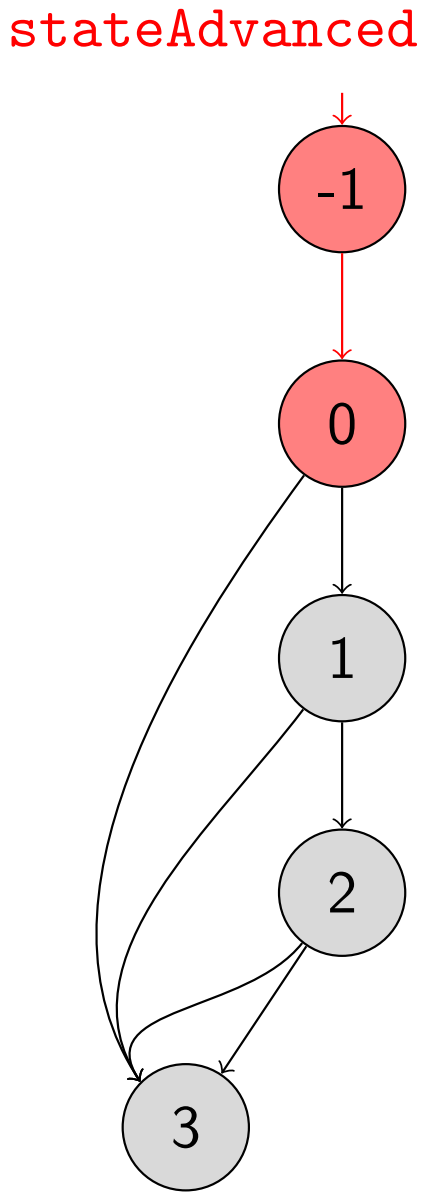
Search events (BFS)



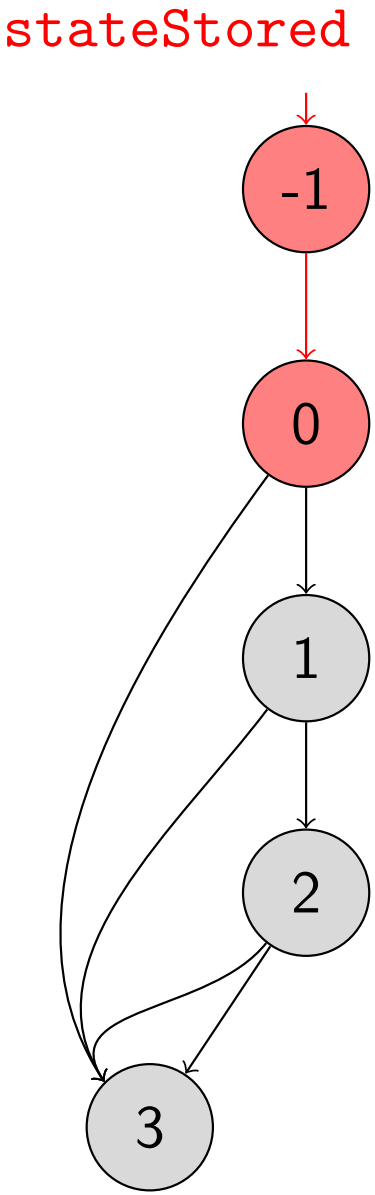
Search events (BFS)



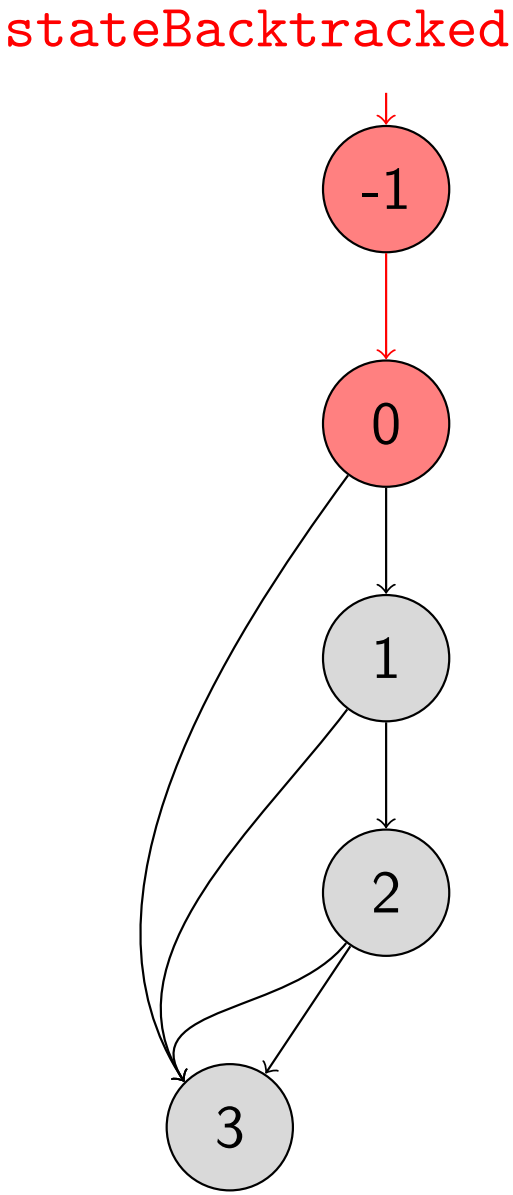
Search events (BFS)



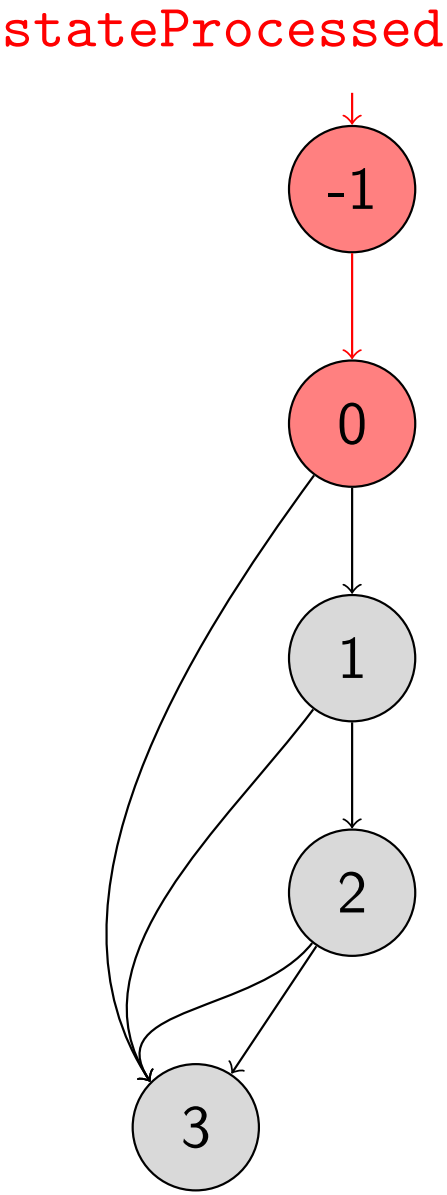
Search events (BFS)



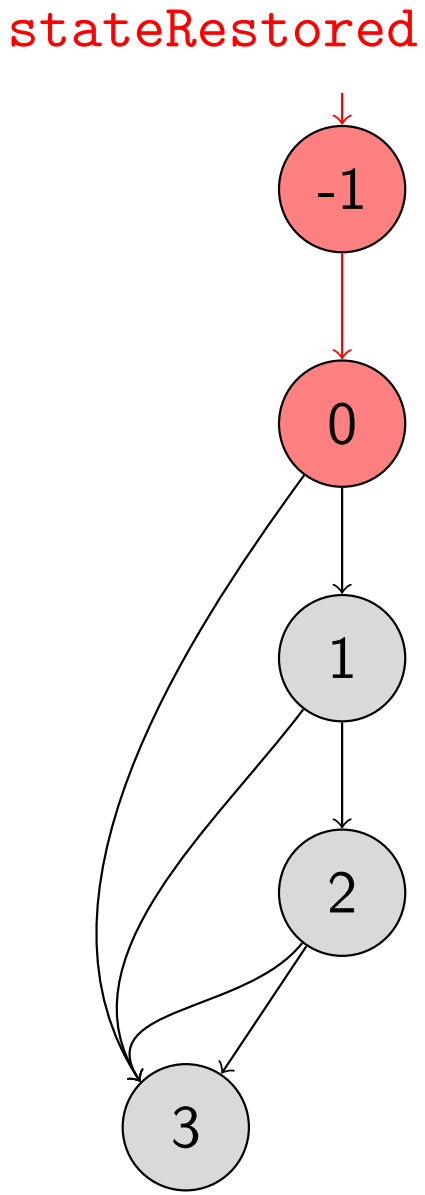
Search events (BFS)



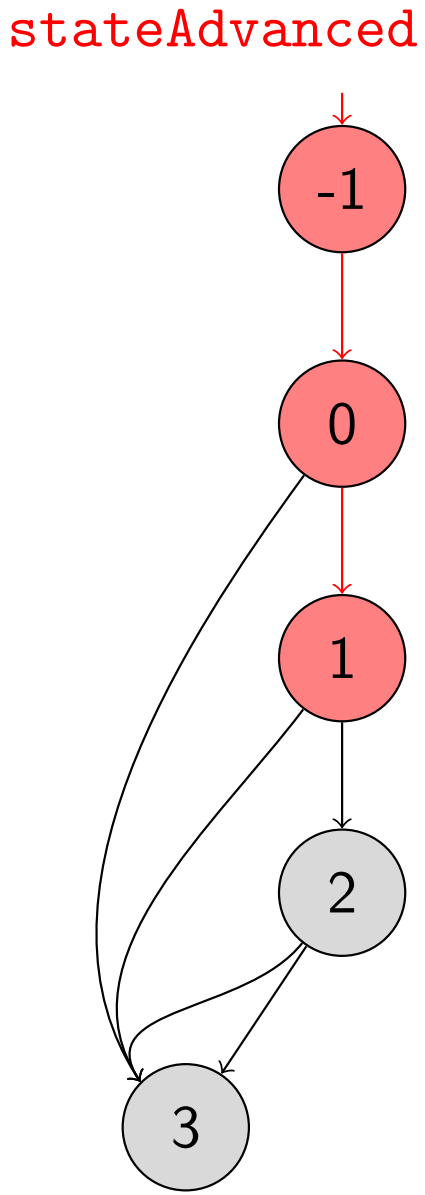
Search events (BFS)



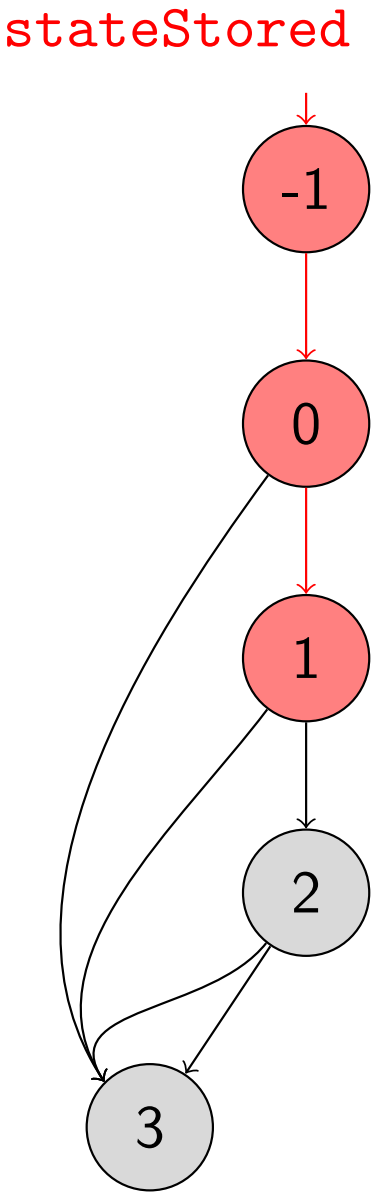
Search events (BFS)



Search events (BFS)

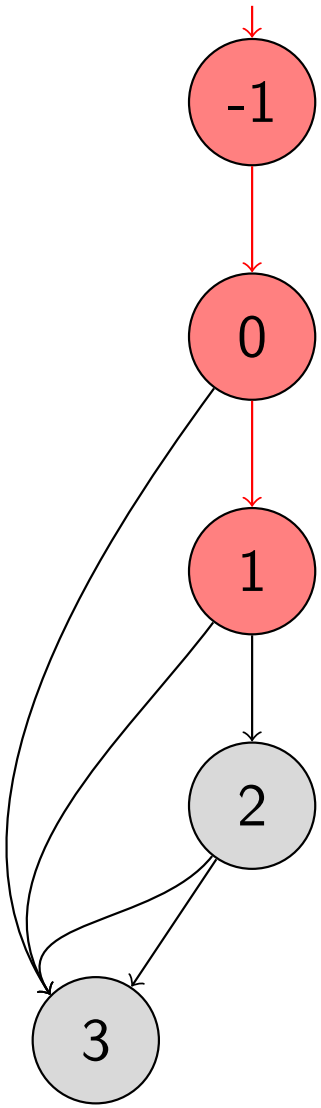


Search events (BFS)

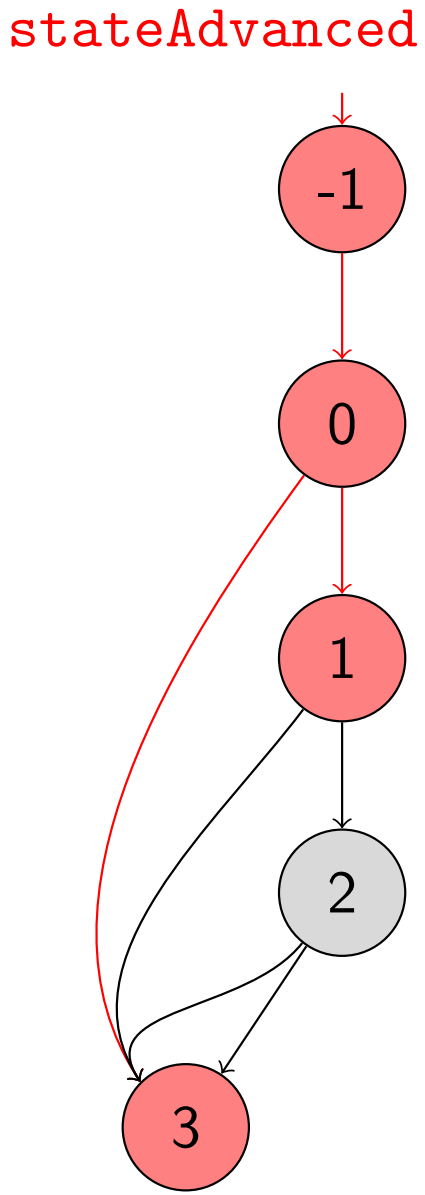


Search events (BFS)

stateBacktracked

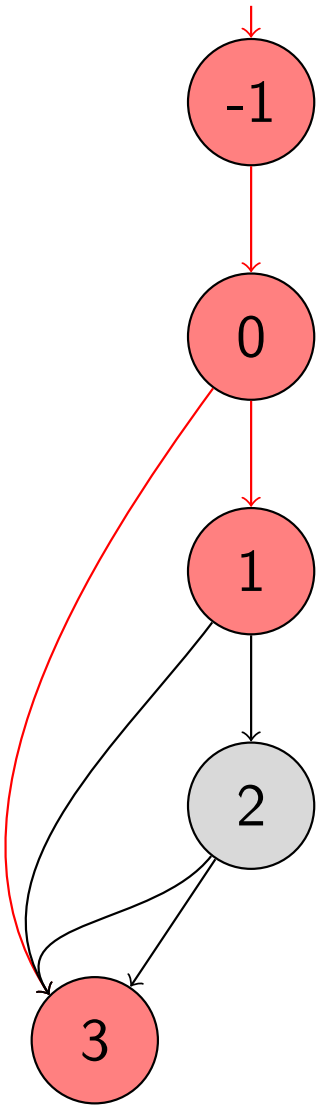


Search events (BFS)



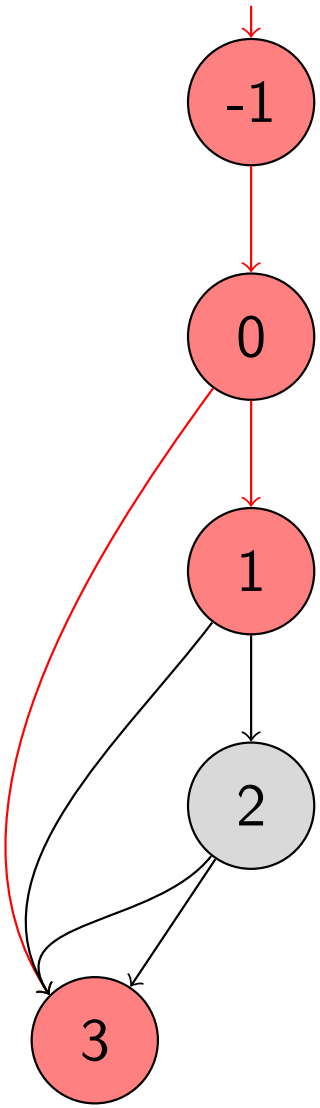
Search events (BFS)

Final states are not stored

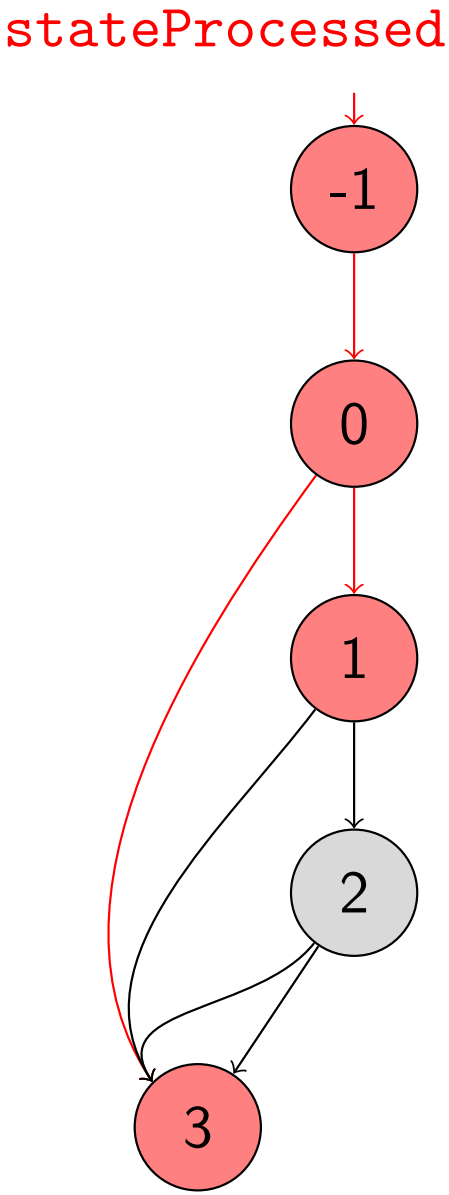


Search events (BFS)

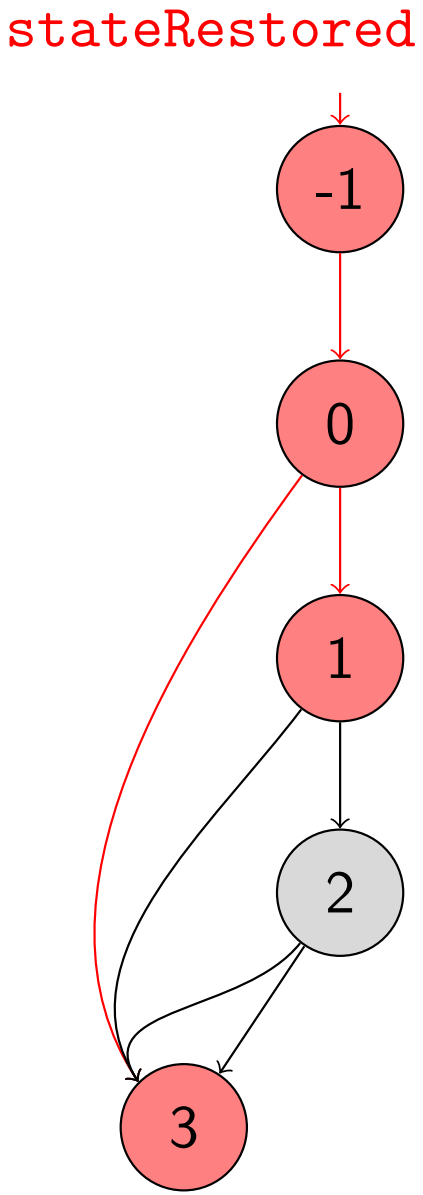
`stateBacktracked`



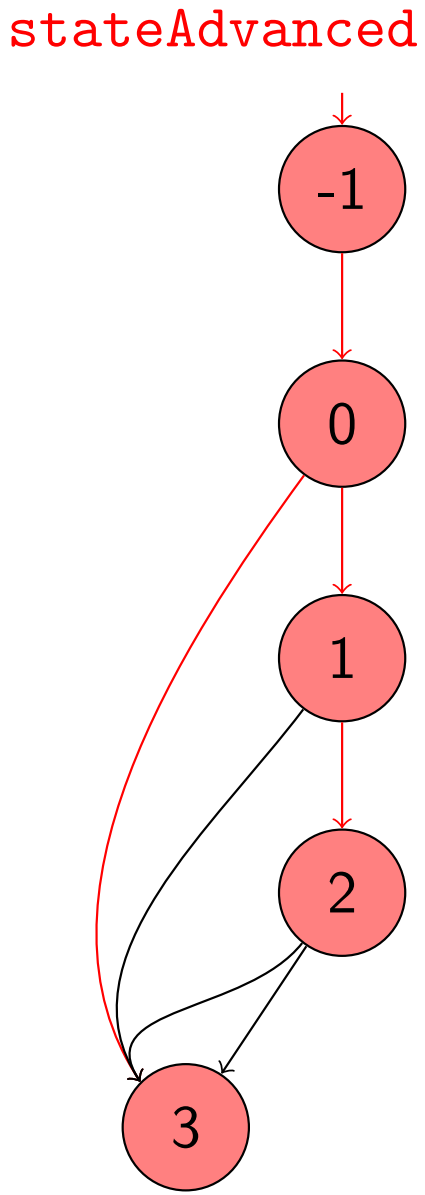
Search events (BFS)



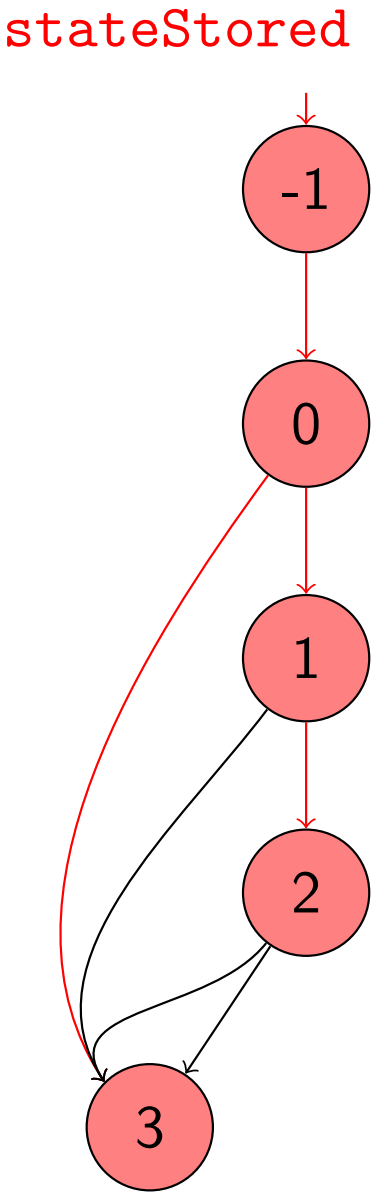
Search events (BFS)



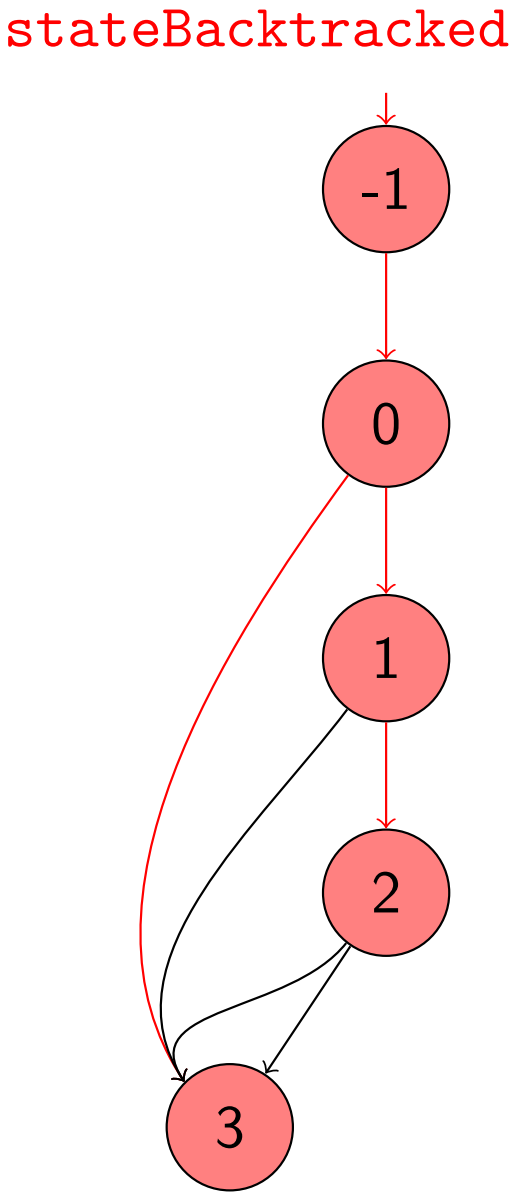
Search events (BFS)



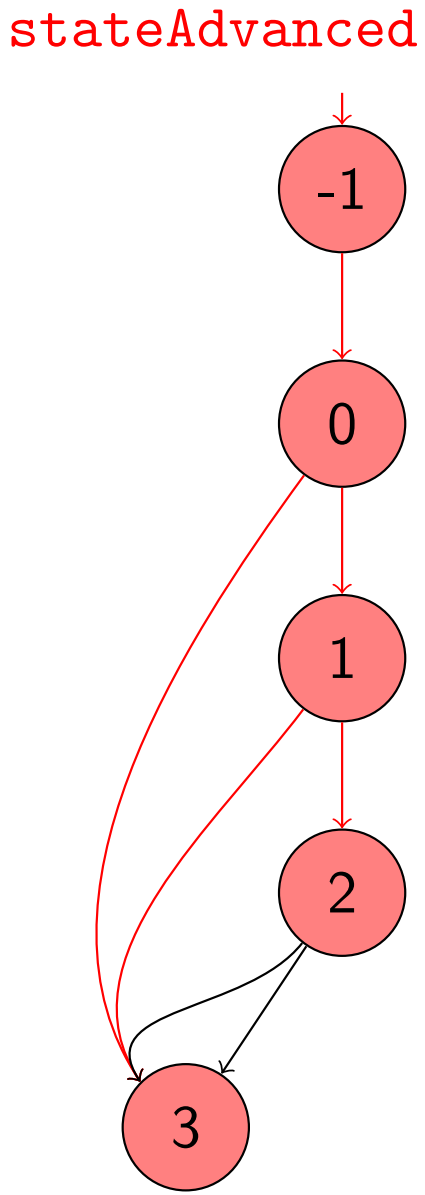
Search events (BFS)



Search events (BFS)

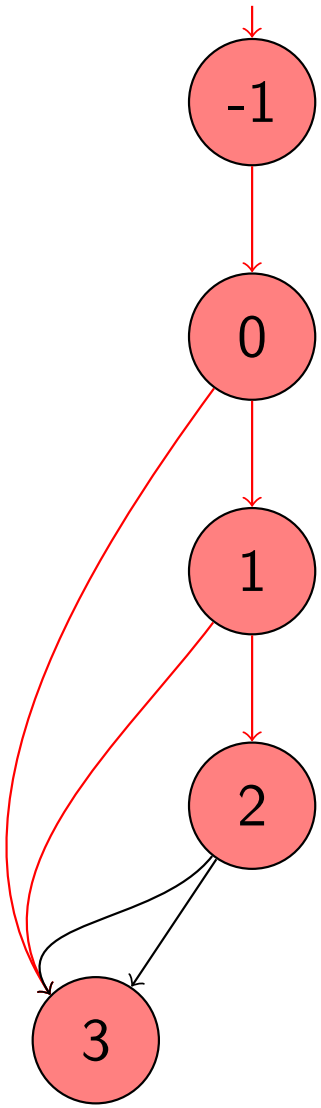


Search events (BFS)

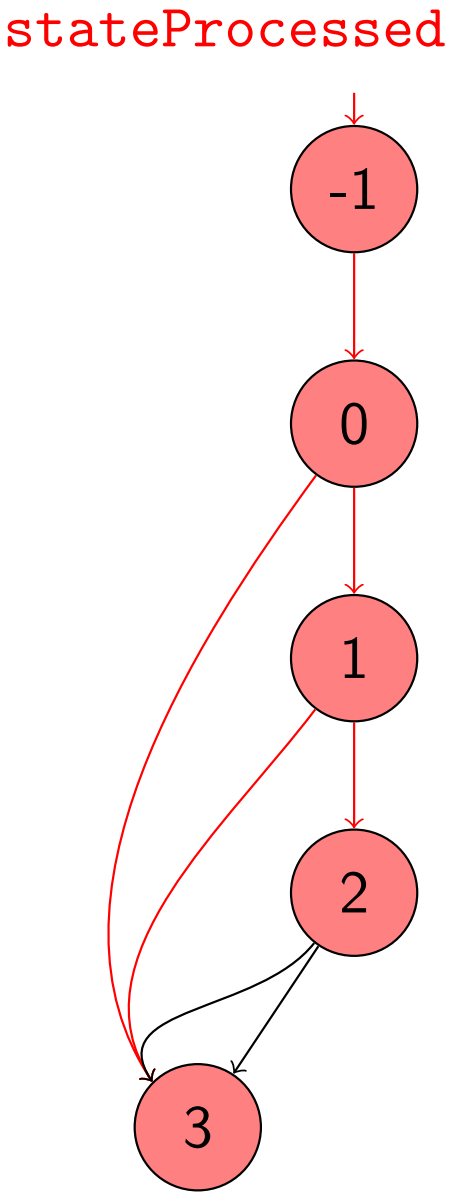


Search events (BFS)

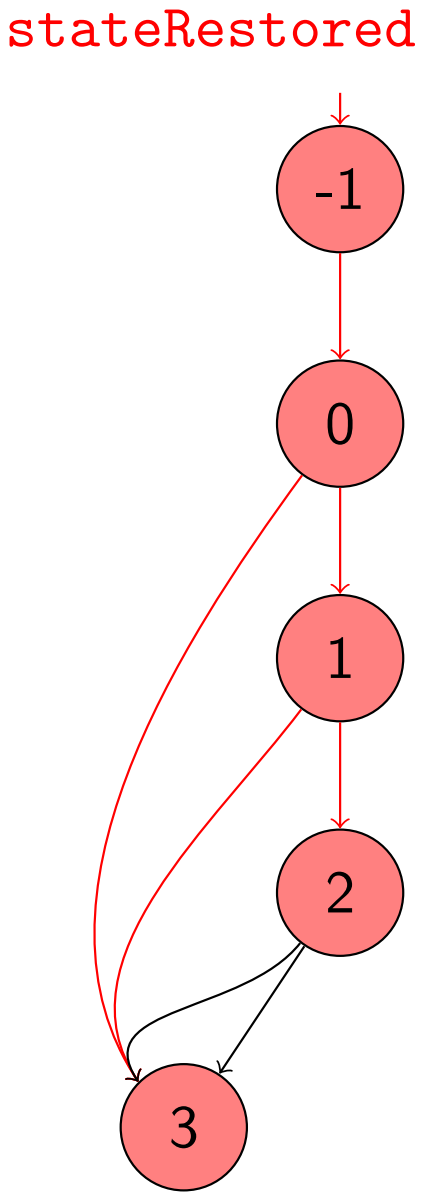
`stateBacktracked`



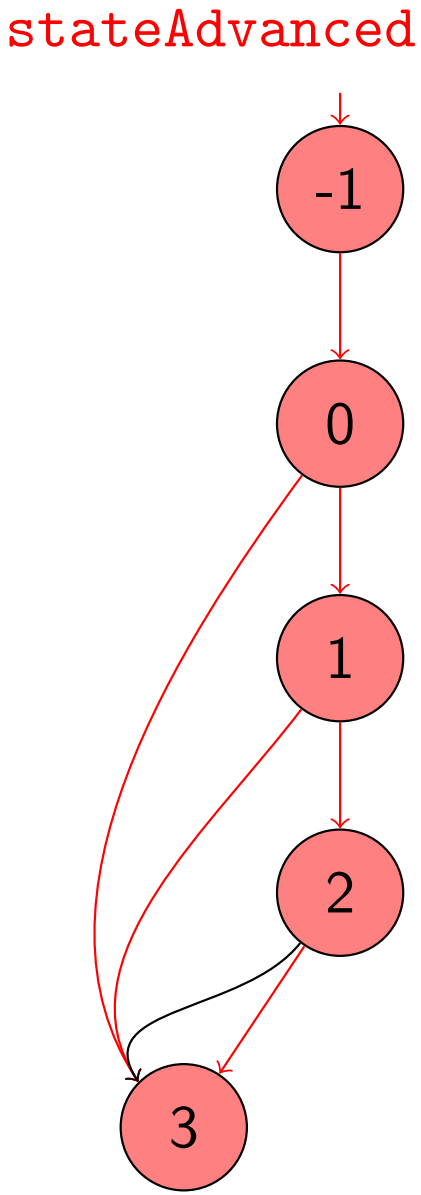
Search events (BFS)



Search events (BFS)

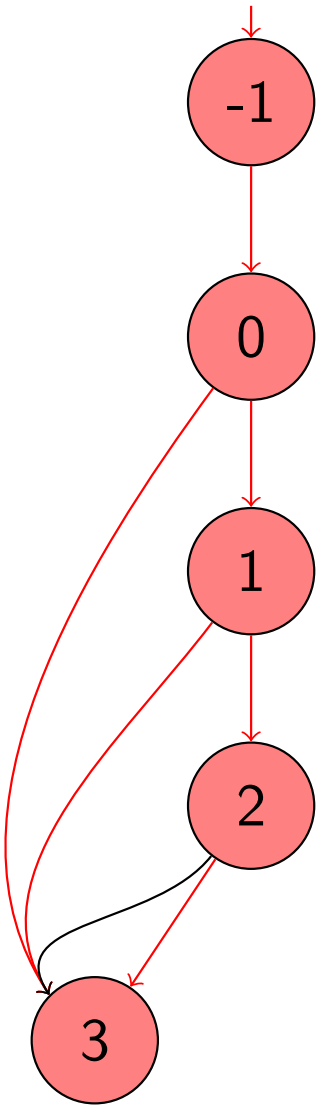


Search events (BFS)

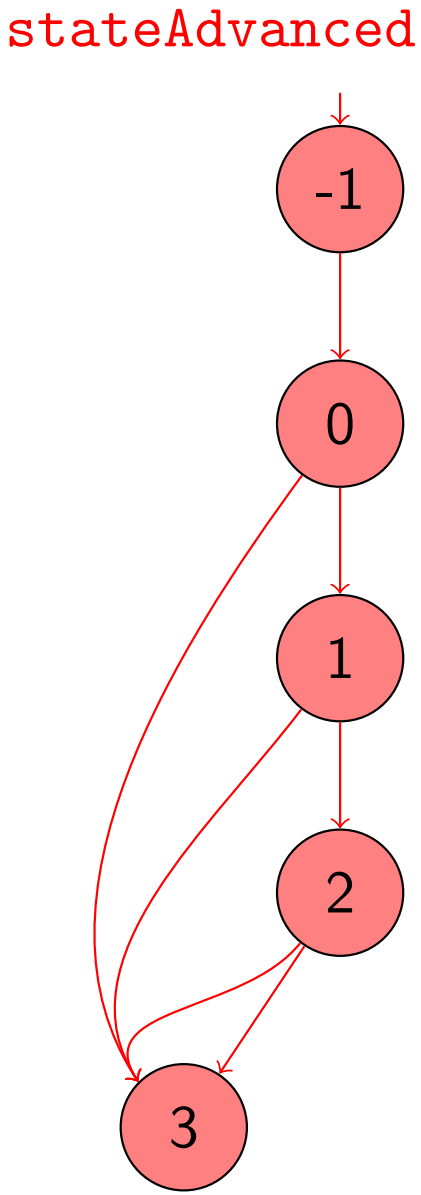


Search events (BFS)

`stateBacktracked`

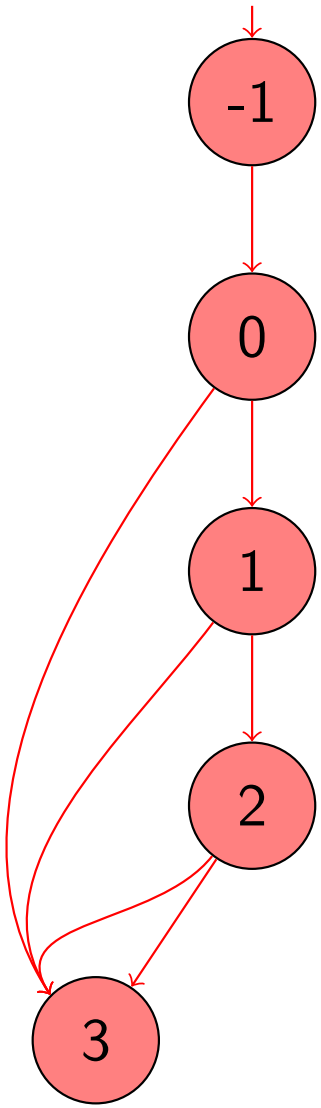


Search events (BFS)

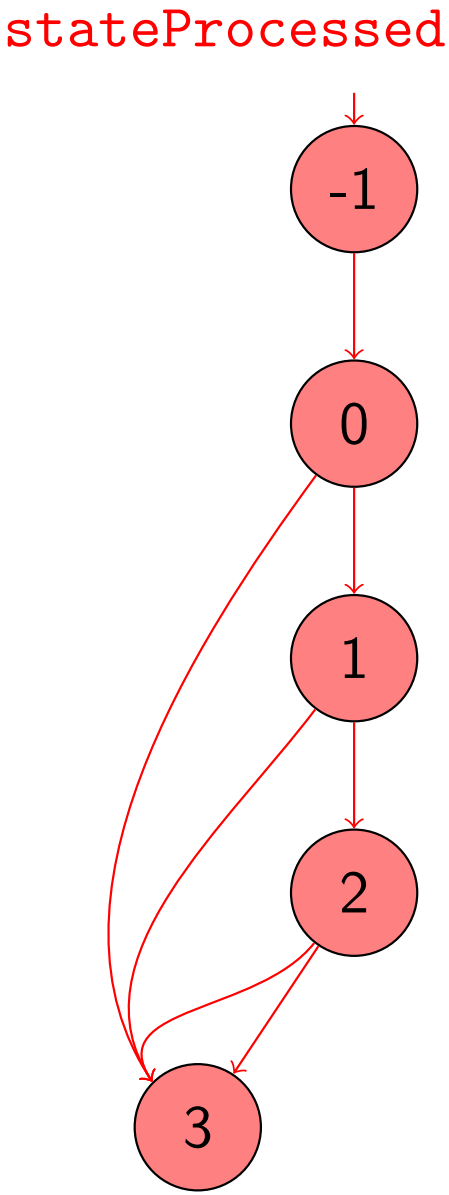


Search events (BFS)

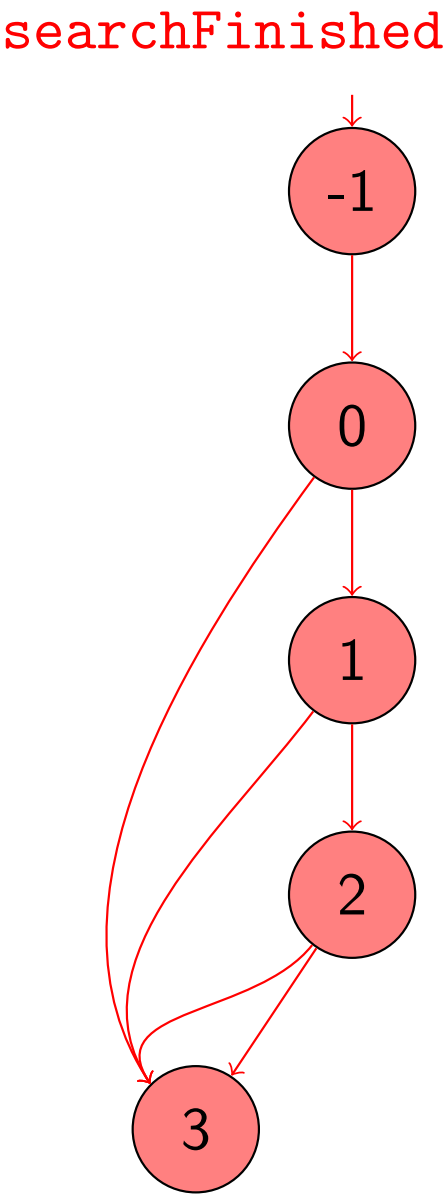
`stateBacktracked`



Search events (BFS)



Search events (BFS)



State space

Implement a listener which prints the states and transitions visited by the search in the following simple format:

```
-1 -> 0
0 -> 1
1 -> 2
0 -> 3
3 -> 4
4 -> 2
```

State space

Question

Which methods of the `SearchListener` interface are relevant?

State space

Question

Which methods of the `SearchListener` interface are relevant?

Answer

`stateAdvanced`, `stateBacktracked`, and `stateRestored`.

State space

Question

In order to print a transition, what information do we need?

State space

Question

In order to print a transition, what information do we need?

Answer

The ID of the source and target state.

State space

Question

In order to print a transition, what information do we need?

Answer

The ID of the source and target state.

Question

How do we store that information?

State space

Question

In order to print a transition, what information do we need?

Answer

The ID of the source and target state.

Question

How do we store that information?

Answer

As attributes.

State space

```
public void stateAdvanced(Search search) {
  this.previous = ???;
  this.current = ???;
}
```

Question

How do we update `this.previous`?

State space

```
public void stateAdvanced(Search search) {
  this.previous = ???;
  this.current = ???;
}
```

Question

How do we update `this.previous`?

Answer

`this.previous = this.current`.

State space

```
public void stateAdvanced(Search search) {
  this.previous = ???;
  this.current = ???;
}
```

Question

How can we use the `Search` parameter of the `stateAdvanced` method to update `this.current`?

State space

```
public void stateAdvanced(Search search) {
  this.previous = ???;
  this.current = ???;
}
```

Question

How can we use the `Search` parameter of the `stateAdvanced` method to update `this.current`?

Answer

Use a method of the `Search` class that returns the ID of the current state (`getStateId`).

State space

Question

Where do we initialize the attributes `current` and `previous`?

State space

Question

Where do we initialize the attributes `current` and `previous`?

Answer

In the constructor.

State space

Question

Where do we initialize the attributes `current` and `previous`?

Answer

In the constructor.

Question

How do we initialize the attributes `current` and `previous`?

State space

Question

Where do we initialize the attributes `current` and `previous`?

Answer

In the constructor.

Question

How do we initialize the attributes `current` and `previous`?

Answer

Set them to -1, the ID of the initial state.

State space

Question

How do we print the transition in `stateAdvanced`?

State space

Question

How do we print the transition in `stateAdvanced`?

Answer

```
System.out.printf("%d -> %d\n", this.previous, this.current
```

State space

Question

How do we implement `stateBacktracked`?

State space

Question

How do we implement `stateBacktracked`?

Answer

```
this.current = search.getStateId();
```

State space

Question

How do we implement `stateRestored`?

State space

Question

How do we implement `stateRestored`?

Answer

```
this.current = search.getStateId();
```

State space

Question

How do we implement `stateRestored`?

Answer

```
this.current = search.getStateId();
```

Question

Do we need the attribute `previous`?

State space

Question

How do we implement `stateRestored`?

Answer

```
this.current = search.getStateId();
```

Question

Do we need the attribute `previous`?

Answer

No.

State space

Implement a listener which creates a dot file representing the the states and transitions visited by the search.

```
digraph statespace {
-1 -> 0
0 -> 1
1 -> 2
0 -> 3
3 -> 4
4 -> 2
}
```

State space

Question

Where do we open a file for writing?

State space

Question

Where do we open a file for writing?

Answer

In the constructor or the method `searchStarted`.

State space

Question

Where do we print `digraph statespace {`?

State space

Question

Where do we print `digraph statespace {`?

Answer

In the constructor and the method `searchStarted`.

State space

Question

Where do we print `digraph statespace {`?

Answer

In the constructor and the method `searchStarted`.

Question

Where do we print the final `}`?

State space

Question

Where do we print `digraph statespace {`?

Answer

In the constructor and the method `searchStarted`.

Question

Where do we print the final `}`?

Answer

In the method `searchFinished`.

State space

Implement a listener which creates a dot file representing the the states and transitions visited by the search. Colour the initial state green and the final states red.

```
digraph statespace {
-1 [fillcolor=green]
-1 -> 0
0 -> 1
1 -> 2
2 [fillcolor=red]
0 -> 3
3 -> 4
4 -> 2
}
```

State space

Question

The initial state always has ID -1. Where do we print `-1 [fillcolor=green]`?

State space

Question

The initial state always has ID -1. Where do we print `-1 [fillcolor=green]`?

Answer

In the constructor or the method `searchStarted`.

State space

Question

The class `Search` has a method `isEndState`. How can this method be used?

State space

Question

The class `Search` has a method `isEndState`. How can this method be used?

Answer

To indicate that the final (end) states are red.