Started on	Friday, 4 June 2021, 9:10 AM
State	Finished
Completed on	Friday, 4 June 2021, 9:24 AM
Time taken	14 mins 15 secs
Marks	55.00/60.00
Grade	**91.67** out of 100.00

Question **1**
Partially correct
Mark 5.00 out of 10.00

Check what is TRUE about uninformed search.

Select one or more:

- [✓ The search can iterate through all states] ✓
- Is a kind of technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement
- It is the type of search that we can call an intelligent search
- The search does know how far is a current state from the start
- [✓ The search can distinguish a non-goal state from a goal state] ✓
- The search does know how close is a current state to the goal

Your answer is partially correct.

You have correctly selected 2.
The correct answers are: Is a kind of technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement
, The search can iterate through all states, The search can distinguish a non-goal state from a goal state, The search does know how far is a current state from the start

Question **2**
Correct
Mark 10.00 out of 10.00

Check what is TRUE about Breadth-First Search.

Select one or more:

- [✓ All nodes are expanded at a given depth before any nodes at the next level are expanded] ✓
- [✓ Implemented by using Queue] ✓
- Implemented by using Stack
- It visits nodes into the deepest level before going back and visiting the siblings in the tree

Your answer is correct.

The correct answers are: All nodes are expanded at a given depth before any nodes at the next level are expanded, Implemented by using Queue

Question **3**

Correct

Mark 10.00 out of 10.00

Check what is TRUE about Depth-First Search.

Select one or more:

- [✓ Implemented by using Stack] ✓
- All nodes are expanded at a given depth before any nodes at the next level are expanded
- Implemented by using Queue
- [✓ It visits nodes into the deepest level before going back and visiting the siblings in the tree] ✓

Your answer is correct.

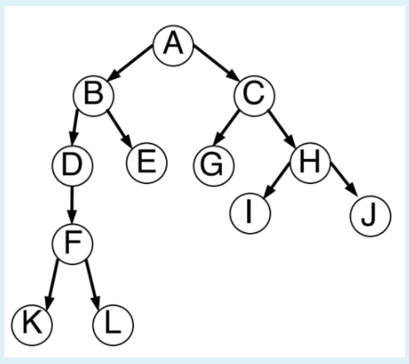
The correct answers are: It visits nodes into the deepest level before going back and visiting the siblings in the tree, Implemented by using Stack

Question **4**

Correct

Mark 10.00 out of 10.00

Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state G. What order would the states be expanded by Breadth-First Search?



Select one:

- [✓ ABCDEG] ✓
- ABDFKLECG
- A ABC ABDECG

Your answer is correct.

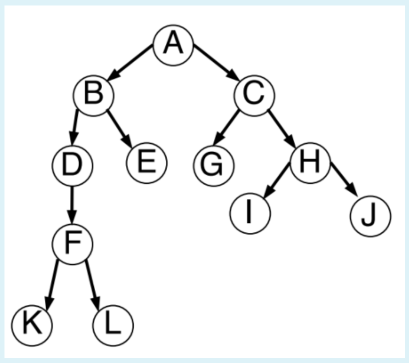
The correct answer is: ABCDEG

Question **5**

Correct

Mark 10.00 out of 10.00

Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state G. What order would the states be expanded by Depth-First Search?



Select one:

- A ABC ABDECG
- [✓ ABDFKLECG] ✓
- ABCDEG

Your answer is correct.

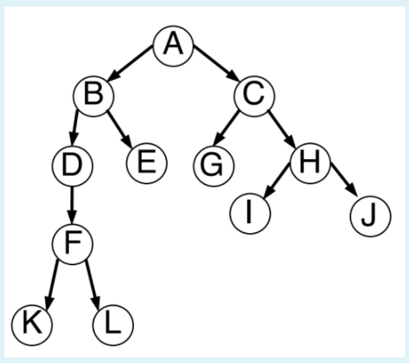
The correct answer is: ABDFKLECG

Question **6**

Complete

Not graded

Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state G. What order would the states be expanded by Iterative Deepening Search?



Select one:

- [✓ ABCDEG]
- ABDFKLECG
- A ABC ABDECG

Your answer is incorrect.

The correct answer is: A ABC ABDECG

Question **7**

Correct

Mark 10.00 out of 10.00

Given a few fragments of code. Which one is from the correct implementation of BFS/DFS search?

Select one:

●
```
openlist.push(starting);
while(true) {
  if (openlist.empty())  { current = null; break; }
  current = openlist.pop();
  closedlist.push(current);
  if (current == target)  break;
  for (var adjacent of getAdjacents(current))
    if (!closedlist.find(adjacent) && !openlist.find(adjacent))
        openlist.push(adjacent);
}
```

✓

○
```
closedlist.push(starting);
while(true) {
  if (closedlist.empty())  { current = null; break; }
  current = closedlist.pop();
  openlist.push(current);
  if (current == target)  break;
  for (var adjacent of getAdjacents(current))
    if (!openlist.find(adjacent) && !closedlist.find(adjacent))
        closedlist.push(adjacent);
}
```

○
```
openlist.push(starting);
while(true) {
  if (!openlist.empty())  { current = null; break; }
  current = openlist.pop();
  closedlist.push(current);
  if (current != target)  break;
  for (var adjacent of getAdjacents(current))
    if (closedlist.find(adjacent) || openlist.find(adjacent))
        openlist.push(adjacent);
}
```

○
```
closedlist.push(starting);
while(true) {
  if (!closedlist.empty())  { current = null; break; }
  current = closedlist.pop();
  openlist.push(current);
  if (current != target)  break;
  for (var adjacent of getAdjacents(current))
    if (openlist.find(adjacent) && closedlist.find(adjacent))
        closedlist.push(adjacent);
}
```

Your answer is correct.

The correct answer is:
```
openlist.push(starting);
while(true) {
  if (openlist.empty())  { current = null; break; }
  current = openlist.pop();
  closedlist.push(current);
  if (current == target)  break;
  for (var adjacent of getAdjacents(current))
    if (!closedlist.find(adjacent) && !openlist.find(adjacent))
        openlist.push(adjacent);
}
```

◄ Attendance cs380su21-meta.sg Friday 4/6/2021 9:00am-12:20pm

Jump to...

Slides (Dijkstra's Search In Explicit Graph) ►