

Ex. No: 1

## Problem Solving- Using State Space Search

27/07/2022

### Uninformed Search Strategies

---

#### BFS - Code:

```
from collections import deque

def BFS(a, b, target):
    pathMap = {}
    isSolvable = False
    path = []

    q = deque()

    q.append((0, 0))

    while (len(q) > 0):

        curr = q.popleft()

        if ((curr[0], curr[1]) in pathMap):
            continue

        if ((curr[0] > a or curr[1] > b or
            curr[0] < 0 or curr[1] < 0)):
            continue

        path.append([curr[0], curr[1]])

        pathMap[(curr[0], curr[1])] = 1

        if (curr[0] == target or curr[1] == target):
            isSolvable = True

            if (curr[0] == target):
                if (curr[1] != 0):
                    path.append([curr[0], 0])
            else:
                if (curr[1] != 0):
                    path.append([0, curr[1]])

            sz = len(path)
            for i in range(sz):
                print("(", path[i][0], ", ",
                    path[i][1], ")")
            break

        q.append([curr[0], b])
        q.append([a, curr[1]])

    for ap in range(max(a, b) + 1):

        c = curr[0] + ap
        d = curr[1] - ap
```

```
if (c == a or (d == 0 and d >= 0)):  
    q.append([c, d])
```

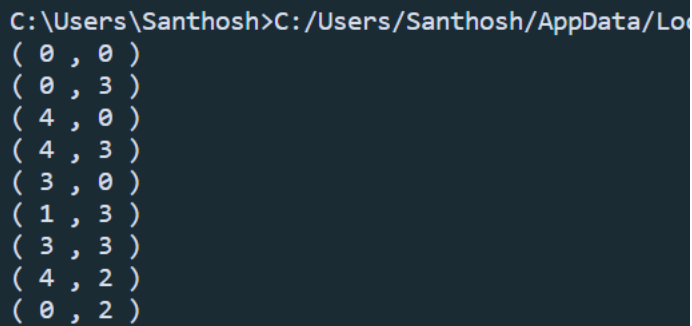
```
c = curr[0] - ap  
d = curr[1] + ap
```

```
if ((c == 0 and c >= 0) or d == b):  
    q.append([c, d])
```

```
q.append([a, 0])  
q.append([0, b])
```

```
if (not isSolvable):  
    print("No solution")
```

```
if __name__ == '__main__':  
    Jug1, Jug2, target = 4, 3, 2  
    BFS(Jug1, Jug2, target)
```

A terminal window with a dark background showing the output of a BFS algorithm. The output consists of a list of coordinate pairs (c, d) representing states in a water jug problem. The pairs are: (0, 0), (0, 3), (4, 0), (4, 3), (3, 0), (1, 3), (3, 3), (4, 2), and (0, 2).

```
C:\Users\Santhosh>C:/Users/Santhosh/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe  
( 0 , 0 )  
( 0 , 3 )  
( 4 , 0 )  
( 4 , 3 )  
( 3 , 0 )  
( 1 , 3 )  
( 3 , 3 )  
( 4 , 2 )  
( 0 , 2 )
```

### DFS – code:

```
def DFS(a, b, target):  
    pathMap = {}  
    isSolvable = False  
    path = []  
  
    stack = []  
  
    stack.append((0, 0))  
  
    while (len(stack) > 0):  
  
        curr = stack.pop()  
  
        if ((curr[0], curr[1]) in pathMap):  
            continue  
  
        if ((curr[0] > a or curr[1] > b or  
            curr[0] < 0 or curr[1] < 0)):  
            continue  
  
        path.append([curr[0], curr[1]])  
  
        pathMap[(curr[0], curr[1])] = 1
```

```
if (curr[0] == target or curr[1] == target):
    isSolvable = True
```

```
    if (curr[0] == target):
        if (curr[1] != 0):
            path.append([curr[0], 0])
    else:
        if (curr[0] != 0):
            path.append([0, curr[1]])
```

```
    sz = len(path)
    for i in range(sz):
        print("(", path[i][0], ", ",
              path[i][1], ")")
    break
```

```
stack.append([curr[0], b])
stack.append([a, curr[1]])
```

```
for ap in range(max(a, b) + 1):
```

```
    c = curr[0] + ap
    d = curr[1] - ap
```

```
    if (c == a or (d == 0 and d >= 0)):
        stack.append([c, d])
```

```
    c = curr[0] - ap
    d = curr[1] + ap
```

```
    if ((c == 0 and c >= 0) or d == b):
        stack.append([c, d])
```

```
stack.append([a, 0])
stack.append([0, b])
```

```
if (not isSolvable):
    print("No solution")
```

```
if __name__ == '__main__':
    Jug1, Jug2, target = 4, 3, 2
    DFS(Jug1, Jug2, target)
```

```
C:\Users\Santhosh>C:/Users/Santh
( 0 , 0 )
( 0 , 3 )
( 4 , 0 )
( 1 , 3 )
( 4 , 3 )
( 3 , 0 )
( 3 , 3 )
( 4 , 2 )
( 0 , 2 )

C:\Users\Santhosh>
```