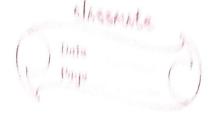
DFS: Depth first Sea. Emp. NO:3 [water Jug] Date: AIM: create a DFS priogram to Solve the mater Jug problem usin python code, Algorithm :-1) Pinibialize the queue: Step 1: ... create a queue q' for BFS step 2:- crease a set visited to keep track of visited grates to avoid yeles. Step 3: .. Enqueue the initial state (0,0) where the both jugs are empty. 2) BFS loop Step 4: .. while queue is not empty · Dequeue the front state (x, y) where x is the amount of water is jug (and if is the amount of water is Jug 2. · If either x == target br. y = = target then solution is found. · If the State x, y has been visited

skip to the nesse



Hank the State xiy shan willbed for the certainent state (x14) Jananasa igel possible come States By applying. & fice And, (Ind 12) - five thing a (thing an, x)) emply sing 1 1 (0,1) -) employ ding 2. 1 (x,9) 2) pour warer from jug 1 cojug 2 with dapacity for Jugo A pour mater from jugo tojugo > mish readerests of jug 1.

check for solution

Step 5: . If the queue is enhauted and the starget been reached print "solution in not possible".

step 6: Otherwise, print the sequence of operation leading to the Solution.



from collection impost dequere
def solution (a, b, target):
m= {3
is solvable = False
path: CJ
q = dequeue ()
grappend ((0,0))
while den (q)>0:
u=q.popletco
if (v (o), v (i)) inm;
continue
if ip [0] > a ox ([1] > b ox ((0) < 0 ox
U [1] < 0
Contine ([[
path. append ([u[o],u[i]))
1-([[1]]) m
if v(o)==tanget or v(i)==tanget: is solvable = true
ig U[1] 1 = 0:
path. append (lu[0].0])
"SI : len (path)
for i in range (SI):
print ("(", path[i][o],",", path[i]
break
break [i]")") 9. append ([u[07,b])
σ. append ([u[o], b]) σ. append ([u[i], a])



on u(o) +ap ge-[1] u = b if c== a or (d == 0 and d>=0); q append ([c,d]) 6=2[=U[0]-ap c=v[i]+ap if (c==0 and c>=0) or d==b: q append (cc,d) q. append ([a; o]) g. append ([0,b]) if not is solvable print ("solution not possible") uj -- name -- - mais -Jug 1= wit (input ("Enter th capacing of jugs")) juig d = int [input ["Enter The target amount")) print ("path from initial state to solution State") solution (jug 1, jug2, target) output:

Enter the capacity of Jug 1:4
Enter the capacity of Jug 2:3
Enter the tanget amount: 2
Enter the tanget amount: 2
path from initial state to solution
-State

(0,0) (1,3) (0,3)

(410) (412) (413) (0,2)

(310)

Program is executed, and output is verified successfully.