```
EXP NO:3
            WATERJUG ...
 :MIA
   To find the shortest path between a
Offert node & goal node in a graph or grid,
exploring only the most promising paths.
CODE
  from Queue import posiority Queue.
    def a - Ostart- search (graph, sslart, goal):
   open- list = priority Queue()
Open - list put ((o) start))
       g_&core = { start : 0).
     | scrop = S start: heuristic (start, god)
        came - from festart] = none.
    while not open-list empty ():
      airrent = open - list get () []
    if current = = geal:
    reconstruct = path (come-from, current)
    fror neighbour, cost in graph [aurent]:
     tentative - g - score = q score [ current f cos
     if neighborar not ing Brove or tentative
          L 9 - scrove [neighbour]: -g-score
```

come - from [neighbour] = current. g-score [neighbour] = tendative -g-score. - fre scoref neighbour] = tentative - 9 - core etter prizing to houristic (neighborgoal) open - list. put (() - score [neighbour]. du con du continuis de la continuis de continuis. : (pap (propa etima) pouso. def heuristic (node, goal): return abs (node (0) - goal (0) + abs((1) hours g'node [i] - goal [i]). des reconstruct pats (one-from, current): total path = [current] while current in come from and come-from [aurent] is not none: (home man current = come = from fairment) total-path. append (airent). to often of total - path reverse () or taline in sweturn total - path. orin C. Crimillain mrs - C.

graph = { (0,0): {((0,1),1)(21,0),j7 (0/1): [[[1]0/0)/1] (120): [1(0,0)1) , ((1,1)2/1)] of (1,1): [((1,0)/1)], ((0,1)/1)-((2/2))i (2/2), []. Plogdi troqui @tart = (0,0) path = a - star - search (graph, start, goal) point (if " path jourd: ¿ path g ") heapy heappush ((trota, o Solution [10] &] = mord - soms Eq. From = smay B Econo = gatant · Handard) e p - diamo - Thus the water jug program is succenfully executed and output is verified.