**Practical no 4**

**AIM:** Implement recursive best-first search algorithm for Romanian map problem.

**CODE:**

|  |
| --- |
| import queue as q  dict\_hn={  'A':336,  'B':0,  'C':160,  'D':242,  'E':161,  'F':176  }  dict\_gn={  'A':{'B':75,'C':118},  'B':{'A':85,'D':211,'E':90},  'C':{'A':120,'F':146},  'D':{'B':75},  'E':{'B':86},  'F':{'C':99}  }  def get\_fn(citystr):  cities=citystr.split(',')  hn=0  gn=0  ctr=0  while ctr!=len(cities)-1:  gn=gn+dict\_gn[cities[ctr]][cities[ctr+1]]  ctr+=1  hn=dict\_hn[cities[len(cities)-1]]  return hn+gn  def expand(mycities,cityq,goal):  tot,citystr=mycities  cities=citystr.split(',')  city2expand=cities[len(cities)-1]  if(city2expand==goal):  ans="The RBST Path is "+citystr+"with the value as "+str(tot);  while not cityq.empty():  cityq.get()  return ans  print("Expanded City -------",city2expand)  tempq=q.PriorityQueue()  for city in dict\_gn[city2expand]:  tempq.put((get\_fn(citystr+','+city),citystr+','+city))  print('First Best and Second Best inserted into tempq')  ctr=1  if(cityq.empty()):  while not tempq.empty():  if ctr==1 or ctr==2:  tempgn,tempcitystr=tempq.get()  print('Inserting into cityqueue :',tempgn,tempcitystr)  cityq.put((tempgn,tempcitystr))  ctr=ctr+1  else:  #pass  tempq.get()  else:  fn=0  citystr=""  fn=getSecondBest(cityq,fn,citystr)  while not tempq.empty():  if ctr==1 or ctr==2:  tempgn,tempcitystr=tempq.get()  if tempgn>ctr:  if ctr==1:  print('Inserting into cityqueue :',tempgn,tempcitystr)  cityq.put((tempgn,tempcitystr))  ctr=3  continue  else:  #break  print("Inserting into CityQueue:",tempgn,citystr)  cityq.put((tempgn,tempcitystr))  ctr+=1  else:  tempq.get()  while not tempq.empty():  tempq.get()  def getSecondBest(cityq,fn,citystring):  fn,citystring=cityq.get()  cityq.put((fn,citystring))  return fn  def main():  start="A"  goal="F"  cityq=q.PriorityQueue()  cityq.put((get\_fn(start),start))  while not cityq.empty():  mycities=cityq.get()  ans=expand(mycities,cityq,goal)  print(ans)  print('performed by krunal 713')  main() |

