```python
from random import randint
import random
from collections import Counter


class Minesters ():

  population=[]            # Society population
  first_Population=[]      # first generations
  cross_Populations=[]     # cross over populations
  mutation_Populations=[] # mutation level population
  evaluated_populations=[] # Evaluation level population
  results =[]                  #cost = 0 and answers

  def __init__(self,param1=20):
    self.pop_Size=param1

  def make_Population(self):
    '''Create first generation Random'''
    gen_List=[]
    pop_List=[]     # Population
    #---start----
    for _ in range(self.pop_Size):
      for i in range(8) :
        n=randint(0,7)  # Make random minister.
        gen_List.append(n)
      #print(f'{gen_List}\n')        # print First minister.
      pop_List.append(gen_List) # add gens to population list
      gen_List=[]                # Clear gen string .
    self.first_Population=pop_List
    print(f'First population has generated {len(self.first_Population)} strings.')


  def make_Population_unfreq(self):
    '''Create first generation !! Unfrequented'''
    done_C=0        # Counter of doing Tasks
    gen_List=[]
    pop_List=[]     # Population
    #---start----
    for _ in range(self.pop_Size):
      n=randint(0,7)  # Make first random minister.
      gen_List.append(n)
      #print(f'{n},{gen_List}\n')        # print First minister.
      while len(gen_List)<8 :            # do until making 8 unfrequented gen
        n =randint(0,7)
        #print(n)
        frqgen_C= 0                       # counter of frequented gen
        for x in range(0,len(gen_List)):
          if n==gen_List[x]:             # if find frequented gen set flag
            frqgen_C +=1
        if frqgen_C ==0 :                 #if is not frequented append it
          gen_List.append(n)
        frqgen_C=0
        done_C +=1
      pop_List.append(gen_List) # add gens to population list
      gen_List=[]
    #print(gen_List,done_C)
    #print(pop_List)
    self.first_Population=pop_List
    return pop_List

  def threat_Cost(self):
    ''' Score threat of ministers
        then you should less score gen
        for next step '''

    #print(self.first_Population)
    for gen in self.first_Population:
      print(gen)
      temp=list(enumerate(gen))
      print(temp)

      print('--------------------------')

  def cross_Over(self):
```

```python
        cross_Population=[] #list of cross level child =16 children
        parents_Index=[]    #Random Parent list , each pairs of parents will make 2 children
        child1=[]
        child2=[]
        for _ in range(16) :    # We want to cross new 16 Children over previous Parents
          parents_Index.append(randint(0,19))   # make Random Parents List
        #Dam
        #parents_Index=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
        print(f'Cross Parents list index are= {parents_Index}')
        break_Pos = randint(0,6)  # Set random break point index of gen Do not Set end index 7 !!!
        #Dam
        #break_Pos = 3
        print(f'Cross break position={break_Pos}')
        #print('paranet&child1 \t \t \t parent&chil2 ')
        for i in range (0,15,2): #i =Pair Of parents index
          #print(f'{self.first_Population[parents_Index[i]]}\t{self.first_Population[parents_Index[i+1]]}')
          child1=self.first_Population[parents_Index[i]][:break_Pos]+self.first_Population[parents_Index[i+1]][break_Pos:]
          child2=self.first_Population[parents_Index[i+1]][:break_Pos]+self.first_Population[parents_Index[i]][break_Pos:]
          #print(f'{child1}\t{child2}')
          #print('--------------------------------------')
          cross_Population.append(child1)
          cross_Population.append(child2)
          child1=[]
          child2=[]
        print(f'Cross population genarated ! len={len(cross_Population)}')
        self.cross_Populations=cross_Population   # copy cross population in public variable

    def mutaion(self):
        '''Select Random 6 Gens and Random Index mutation'''
        mutation_Population=[]
        parents_Index=[]         #random list of selected gens fo mutate
        mutated_child=[]
        #mutation_Pos = randint(0,7)   # Set Mutation Index in gen string
        for _ in range(6):
          parents_Index.append(randint(0,15))   # select 6 random gen
        #print('Main Gens \t\t\t\t  Murated Gens \t Pos')
        for i in range(6) :
          mutated_child=self.cross_Populations[parents_Index[i]]
          #print(f'{mutated_child}')
          mutation_Pos = randint(0,7)          # set Mutation Position
          mutated_child[mutation_Pos]=randint(0,7)
          #print(f'\t {mutated_child} \t pos={mutation_Pos}')
          mutation_Population.append(mutated_child)
          mutated_child=[]
        print(f'{len(mutation_Population)} Mutated Gens string has made .')
        #print(mutation_Population)
        self.mutation_Populations=mutation_Population

    def evaluation(self):
        '''Calculaes Cost score
        threat score (Ministers that are in diagonal positions) add with
        frequnted score (ministers who repeated in gen string)'''

        def frequent(g_list):
          ''' Calculates scores of frequented ministers in gen string'''
          f_score=0
          freq_D = Counter(g_list)    # dict of minister's requention
          for d in freq_D.values():
            if d>1:                    # frequntion Condition
              f_score +=d
            else:
              pass
          #print(f'*{g_list} \t freq={f_score}*')
          return f_score

        def threat(g_list):
          ''' Calculates diagonal threat'''
          t_score=0
          for i in range(8):
            for j in range(i+1,8):
              if abs(g_list[i]-g_list[j]) == abs(i-j) :
                t_score +=1
              else:
                pass
          #print(f'*{g_list} \t threat={t_score}*')
          return t_score
```

```python
    evaluation_Population=[]  # after calculate summery cost we select 20 low cost gen
    society=self.first_Population+self.cross_Populations+self.mutation_Populations
    cost_List=[]             # cost's list
    gens_Cost_dict={}        # dict for save Gens index:Cost

    print(f'Society has {len(society)} Gen strings .')
    for position in society :
      cost_score=frequent(position)+threat(position)     # sumery of Cost
      if cost_score== 0:
        self.results.append(position)                    # add zero cost to resualt list
      else:
        pass
      cost_List.append(cost_score)                       # add to cost list
      #print(f' {position} \t F={frequent(position)} \t T={threat(position)} \t cost={cost_score}')
    gens_Cost_dict=dict(zip (range(len(society)),cost_List))
    #print(f'dict len={len(gens_Cost_dict)}')
    sorted_gens_Cost_dict=sorted(gens_Cost_dict.items(),key=lambda x:x[1],reverse=False)
    #print(sorted_gens_Cost_dict)
    for i in range(20):
      evaluation_Population.append(society[sorted_gens_Cost_dict[i][0]])
    print(f'{len(evaluation_Population)} gen string has evaluated !')
    self.evaluated_populations=evaluation_Population
    print(f'{len(self.results)} answers has found .')
    print('_____')

  def run(self):
    self.population=[]           # Society population
    self.first_Population=self.evaluated_populations     # first generations
    #print(self.first_Population)
    self.cross_Populations=[]    # cross over populations
    self.mutation_Populations=[] # mutation level population
    self.evaluated_populations=[]
    self.cross_Over()
    self.mutaion()
    self.evaluation()
```

Run Area

```python
flag=0
b=Minesters(20)
b.make_Population_unfreq()
b.cross_Over()
b.mutaion()
b.evaluation()
for i in range(3000):
  b.run()
```

Streaming output truncated to the last 5000 lines.
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11000 answers has found .
_____
Cross Parents list index are= [11, 7, 3, 1, 3, 18, 5, 19, 15, 7, 4, 10, 14, 4, 13, 11]
Cross break position=0
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11036 answers has found .
_____
Cross Parents list index are= [10, 5, 9, 15, 1, 2, 11, 2, 15, 4, 17, 3, 12, 9, 8, 13]
Cross break position=0
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11066 answers has found .
_____
Cross Parents list index are= [18, 13, 5, 16, 11, 15, 16, 7, 13, 14, 14, 7, 1, 5, 3, 1]
Cross break position=0
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11098 answers has found .
_____
Cross Parents list index are= [3, 7, 19, 17, 4, 5, 10, 7, 2, 11, 17, 3, 12, 1, 12, 2]
Cross break position=2
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11130 answers has found .
_____
Cross Parents list index are= [0, 12, 5, 0, 7, 2, 2, 9, 14, 9, 16, 7, 1, 14, 6, 7]
Cross break position=3
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11162 answers has found .
_____
Cross Parents list index are= [7, 6, 8, 17, 13, 10, 3, 7, 4, 5, 3, 8, 1, 8, 6, 11]
Cross break position=2
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11196 answers has found .
_____
Cross Parents list index are= [10, 17, 7, 15, 14, 17, 19, 17, 8, 17, 4, 8, 10, 12, 9, 18]
Cross break position=3
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11228 answers has found .
_____
Cross Parents list index are= [19, 1, 1, 13, 4, 19, 5, 8, 17, 11, 15, 18, 19, 12, 19, 18]
Cross break position=4
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11262 answers has found .
_____
Cross Parents list index are= [14, 8, 15, 12, 5, 14, 3, 8, 4, 17, 17, 17, 19, 6, 3, 9]
Cross break position=2
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !
11297 answers has found .
_____
Cross Parents list index are= [2, 17, 2, 5, 5, 11, 2, 5, 15, 17, 5, 0, 9, 1, 6, 4]
Cross break position=0
Cross population genarated ! len=16
6 Mutated Gens string has made .
Society has 42 Gen strings .
20 gen string has evaluated !