Kahbod Aeini - 98101209

1.
    a.
        i. As far as insects can hop to and off the walls, they have no difference with empty paths. So state-space is the whole map coordinates with no respect for each square to be a wall or empty path and we assume an MxN two-dimensional array as our feasible path and state-space. And Insect coordination is (x, y) where 0 ≤ i ≤ M-1 and 0 ≤ j ≤ N-1. Our actions in each state (except squares on the edges) would be Left, Right, Up and Down.
        ii. Since insects can move on any square, our state-space size equals the number of all the squares which is MxN.
    b. Since insects can not hop on or off the walls in this subproblem, our state-space decreases from MxN to the number of empty squares.
        i. We consider W as number of walls and E as number of empty squares so we have: E = MN - W
        ii. Since we have no data about the insects' coordinations, we assume that in each empty square is or is not an insect spotted. So our state-space size would be $|S| = 2^{E}$
2. For ease we consider each city as a letter and hence our path as a string made by these letters.
    a. Since we have 10 cities and we want to generate a path from one of these cities to another and meet all other cities exactly once in it, we model it as a genetic problem where we have **10** genes to make a DNA of.
    b. Assume we have two strings S_1 and S_2. Now we choose a random number within the number of their genes (i.e. 4 in our problem). Then we divide strings by the random index and replace each gene one by one in two strings and if there are repeated letters we substitute them with the initial letter. For example we have
    S_1= ABCDEFGHIJ and S_2= AGHEFCBDJI. Now let's separate strings
    S_1= ABCDEFGHIJ => ABCD + EFGHIJ
    S_2= AGHEFCBDJI => AGHE + FCBDJI
    First gene of both DNAs are equal so we process the second one.
    To generate first new string from replacing S_1 letters in S_2, we replace G in S_2 with B and put G in place of initialed B in S_2, so so far C_1(first child) is ABHEFCGDJI. And in the end we have below new strings as children.
    C_1 = ABCDFHGEJI
    C_2 = AGHEDFBCIJ
    c. We can choose two random numbers within [0, 9] and substitute letters in those indexes with each other.
3.

a. f(x_1) = 7 + 6 + 2 * 5 - 3 - 8 + 4 = 16

   f(x_2) = 9 + 0 + 2 * 3 - 6 - 4 + 2 = 7

   f(x_3) = 9 + 2 + 2 * 8 - 3 - 1 + 3 = 26

   f(x_4) = 2 + 3 + 2 * 3 - 3 - 8 + 4 = 4

b. A

   i. Most fitted chromosomes are x_1 and x_3.

      Now we generate new chromosomes by one-point crossing them over.
      x_5 = 765313

      x_6 = 928384

   ii. Most unfitted chromosomes are x_2 and x_4.

      Now we generate new chromosomes by two-point crossing them over.
      x_5 = 902342

      x_6 = 233684

c. f(x_5) = 7 + 6 + 2 * 5 - 3 - 1 + 3 = 22

   f(x_6) = 9 + 2 + 2 * 8 - 3 - 8 + 4 = 20

   f(x_7) = 9 + 0 + 2 * 2 - 3 - 4 + 2 = 8

   f(x_8) = 2 + 3 + 2 * 3 - 6 - 8 + 4 = 1

d. Most optimal chromosome has the maximum fitness value. So we can generate it by maximizing genes with positive coefficients and minimizing genes with negative ones.
   x_{Best}= 999009

e. It is not possible to generate the most optimal chromosome with this population because there is no 999 substring in this population or there is no 00 substring in 4 to 5 index of the given chromosomes. So without mutation and only with crossover we can not generate the most optimal chromosome, but with another population, there is a rare possibility to achieve it because in these situations we need mutations and with only crossover it is unlikely to achieve it.