```
import random
def score(parent1, parent2):
    # doing crossover
    for i in range(len(parent1)-1, len(parent1)-4, -1):
        parent1[i], parent2[i] = parent2[i], parent1[i]
    #doint mutation by randomly selecting the genes
    mutation_index = [random.randint(0, len(parent1)-1) for i in range(len(parent1)//2)]

    for i in mutation_index:
        if parent1[i] == '0':
            parent1[i] = '1'
        else:
            parent1[i] = '0'

        if parent2[i] == '0':
            parent2[i] = '1'
        else:
            parent2[i] = '0'

    score1 = parent1.count('1')
    score2 = parent2.count('1')
    #checking which child is better with more gene of type1
    if score1 > score2:
        return [''.join(parent1), score1]
    else:
        return [''.join(parent2), score2]

def genetic_algo():
    # Taking input as no. of parents
    n = int(input('Enter the number of parents: '))
    parents = []
    #taking parents genes as input 1 by 1


    for i in range(n):
        parents.append(list(input(f'Enter the parent{i+1}: ')))
    results = []


    #finding the score and storing it in results
    for i in range(len(parents)):
        for j in range(i+1, len(parents)):
            arr = [parents[i].copy(), parents[j].copy()]
            scores = score(parents[i], parents[j])
            results.append(scores + arr)


    # finding the best score among all combination of parents
    results.sort(key=lambda x: x[1], reverse=True)
    print(f'The best offspring among the parents is : {results[0][0]} and the parents are 
{"".join(results[0][2])} and {"".join(results[0][3])}')


genetic_algo()
```