LAB

Algoritmos y Programación

Manejo de Arrays

Dados 2 arrays, llamados Padre y Madre, generar un nuevo array mediante el *cruce de orden* de ambos que consiste en lo siguiente:

$(h, k, c, e, f, d, b, l, a, i, g, j)$ Padre
$(a, b, c, d, e, f, g, h, i, j, k, l)$ Madre

Cruce de orden

1. Se eligen dos puntos de corte aleatoriamente.
2. Entre estos dos puntos de corte se sitúan los elementos del padre.
3. El resto se van eligiendo de la madre siempre que no hayan sido seleccionados previamente. Se comienza a partir del segundo punto de corte.

Si los puntos de corte son el sexto y el noveno, el hijo será:

$$(d, e, f, g, h, i\,|b, l, a|\,j, k, c).$$

Hacer una implementación de este algoritmo, en el método llamado ***order_crossover*** dentro de ***code.py.***

La interfaz es la siguiente:
def order_crossover(parent1, parent2, lower_bound, upper_bound):

Por ejemplo:

Entrada:

parent1= [8,11,3,5,6,4,2,12,1,9,7,10] #padre

parent2= [1,2,3,4,5,6,7,8,9,10,11,12] #madre

lower_bound= 6 #limite inferior

upper_bound= 9 #limite superior

Salida esperada:

[4, 5, 6, 7, 8, 9, 2, 12, 1, 10, 11, 3]

1 2 3

main.py

```python
from solve_tsp import *

string1= input()
parent1 = [int(k) for k in string1.split(',')]

string2= input()
parent2 = [int(k) for k in string2.split(',')]

lower_bound = int(input())
upper_bound = int(input())

solution = order_crossover(parent1, parent2, lower_bound, upper_bound)

print(solution)
```

solve_tsp.py

```python
def order_crossover(parent1, parent2, lower_bound, upper_bound):
    child1 = []

    return child1

```