



**INSTITUTO POLITÉCNICO
NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**



ALGORITMOS GENÉTICOS

PRÁCTICA 7 OPERADORES DE CRUZA 2

Profesora: Sandra Luz Morales Guitrón

Alumno: Alba Díaz Diego Samuel

Grupo: 3CM5

ÍNDICE DE CONTENIDO

Introducción	1
Contenido	1
Order Crossover (OX1)	1
Partially Mapped Crossover (PMX)	2
Position-Based Crossover (POS)	2
Order-Based Crossover (OX2)	3
Cycle Crossover (CX)	3
Práctica 7	4
Descripción	4
Implementación de Order Crossover (OX1)	4
Compilación y Ejecución	4
Implementación de Partially Mapped Crossover (PMX)	5
Compilación y Ejecución	5
Implementación de Position-Based Crossover (POS)	6
Compilación y Ejecución	6
Implementación de Order-Based Crossover (OX2)	7
Compilación y Ejecución	7
Implementación de Cycle Crossover (CX)	8
Compilación y Ejecución	8
Conclusión	9

Introducción

En algunos AG, no todos los cromosomas representan soluciones válidas. En algunos casos, es necesario usar operadores de cruce y mutación diseñados para evitar violar las limitaciones del problema.

Por ejemplo, un AG para resolver el problema del vendedor ambulante puede usar una lista ordenada de ciudades para representar una ruta de solución. Tal cromosoma solo representa una solución válida si la lista contiene todas las ciudades que el vendedor debe visitar. El uso de los operadores de cruce vistos en la *Práctica no. 6* a menudo resultara en cromosomas inválidos. Los AG que optimizan el ordenamiento de una lista dada requieren diferentes operadores de cruce que eviten generar soluciones invalidas.

En esta práctica se estudiarán los siguientes operadores de cruce para la representación de permutaciones:

- Order crossover (OX1).
- Partially mapped crossover (PMX).
- Position-based crossover (POS).
- Order-based crossover (OX2).
- Cycle crossover (CX).

Contenido

Order Crossover (OX1)

Procedimiento:

1. Crear dos puntos de cruce aleatorios en el primer padre.
2. Producir un hijo copiando la sub-cadena limitada por los puntos de cruce generados en el paso anterior en las posiciones correspondientes con el primer padre. Las posiciones restantes se dejan en blanco.
3. Borrar del segundo padre los valores que se encuentren en la sub-cadena. La secuencia resultante contiene los valores restantes.
4. Colocar los valores restantes en las posiciones en blanco del primer hijo de izquierda a derecha.
5. Para obtener el segundo hijo se repiten los pasos del 1 al 4, pero invirtiendo los roles de los padres.

1) P1:

1	4	2	3
---	---	---	---

P2:

4	3	2	1
---	---	---	---

sub-cadena:

4	2
---	---

2) H1:

X	4	2	X
---	---	---	---

3) P2':

X	3	X	1
---	---	---	---

4) H1:

3	4	2	1
---	---	---	---

Ilustración 1. Ejemplo de Order Crossover.

Partially Mapped Crossover (PMX)

Procedimiento:

1. Elegir aleatoriamente dos puntos de cruce.
2. Intercambiar los segmentos generados entre los padres para formar los hijos.
3. El resto de los valores que conforman los hijos se obtienen haciendo mapeo entre los dos padres: si un valor está contenido en el segmento intercambiado, se sustituye por el valor que tenga en el otro segmento.

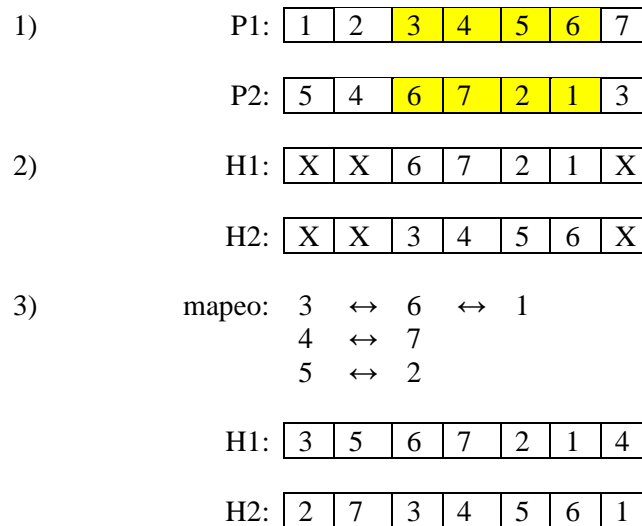


Ilustración 2. Ejemplo de Partially Mapped Crossover.

Position-Based Crossover (POS)

Procedimiento:

1. Seleccionar al azar un conjunto de posiciones del primer padre (no necesariamente consecutivas).
2. Producir un hijo borrando del primer padre las posiciones que no estén en el conjunto formado.
3. Borrar del segundo padre los valores del conjunto. La secuencia de valores resultante se utilizará para completar el hijo.
4. Colocar en los espacios en blanco del hijo la secuencia de valores resultante, de izquierda a derecha.
5. Para formar al segundo hijo se repiten los pasos del 1 al 4, pero intercambiando los roles de los padres.

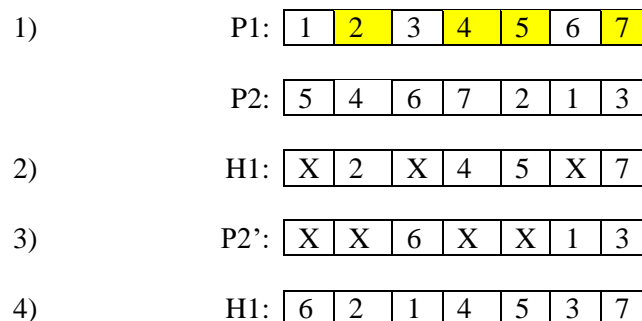


Ilustración 3. Ejemplo de Position-Based Crossover.

Order-Based Crossover (OX2)

Procedimiento:

1. Seleccionar al azar un conjunto de posiciones del primer padre (no necesariamente consecutivas).
2. Generar un hijo removiendo del segundo padre los valores que estén en el conjunto formado.
3. Completar el hijo con los valores del conjunto, insertando de izquierda a derecha.

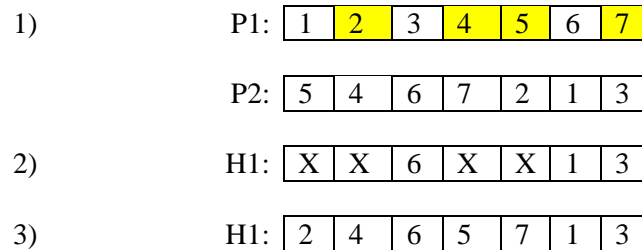


Ilustración 4. Ejemplo de Order-Based Crossover.

Cycle Crossover (CX)

Procedimiento:

1. Encontrar un ciclo que se define mediante las posiciones correspondientes de los valores entre los padres.
2. Generar el primer hijo borrando del primer padre los elementos que no estén en el ciclo.
3. Borrar del segundo padre los elementos que estén en el ciclo. La secuencia de valores resultante se utilizará para rellenar al hijo.
4. Completar al hijo utilizando la secuencia resultante, sustituyendo de izquierda a derecha.
5. Repetir los pasos del 1 al 4 para generar al segundo hijo, pero intercambiando los roles de los padres.

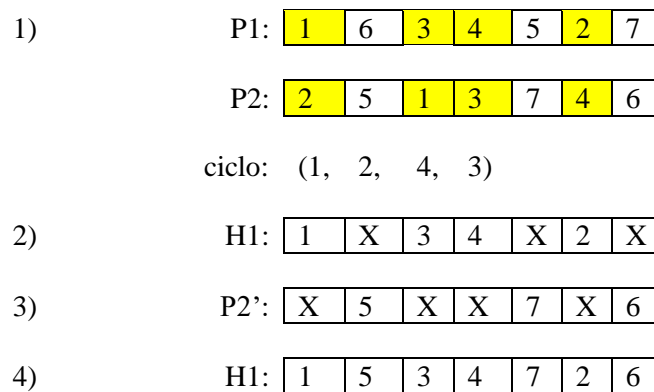


Ilustración 5. Ejemplo de Cycle Crossover.

Práctica 7

Descripción

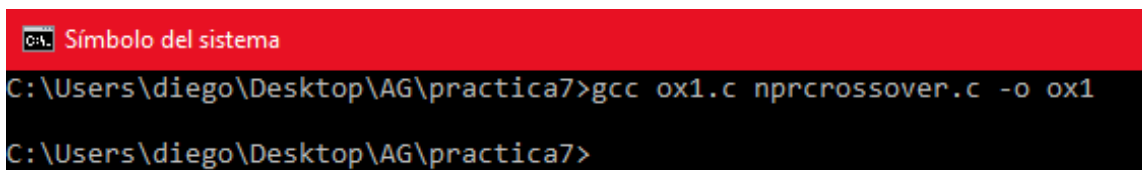
Programar los operadores de cruce descritos en esta práctica, sin importar el tamaño del cromosoma ni de la población.

Implementación de Order Crossover (OX1)

Compilación y Ejecución

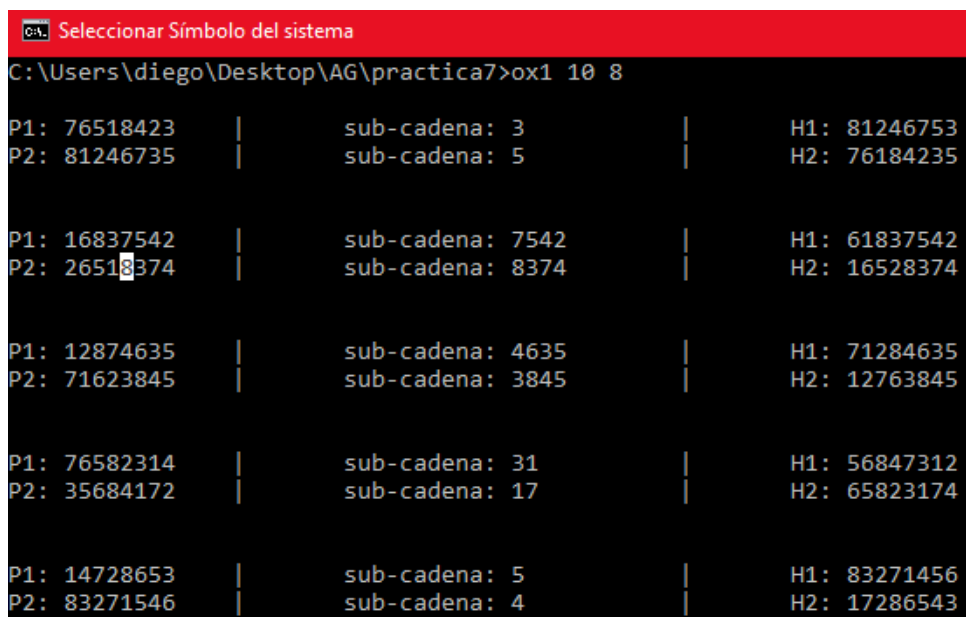
En la *Figura 1* y *Figura 2* se anexan capturas de pantalla con la compilación y ejecución del código de programa desarrollado para el operador OX1. La ejecución se realiza para una población con tamaño igual a 10 y la longitud del cromosoma es igual a 8.

En la *Figura 2* se resumen los resultados obtenidos en 3 columnas. En la primera columna se observan los padres generados de forma aleatoria. La segunda columna contiene la sub-cadena seleccionada aleatoriamente. Y en la tercera columna se observa la descendencia obtenida mediante el método de cruce implementado.



```
C:\Users\diego\Desktop\AG\practica7>gcc ox1.c nprcrossover.c -o ox1  
C:\Users\diego\Desktop\AG\practica7>
```

Figura 1. Compilación – OX1.



```
C:\Users\diego\Desktop\AG\practica7>ox1 10 8
```

P1: 76518423		sub-cadena: 3		H1: 81246753
P2: 81246735		sub-cadena: 5		H2: 76184235
P1: 16837542		sub-cadena: 7542		H1: 61837542
P2: 26518374		sub-cadena: 8374		H2: 16528374
P1: 12874635		sub-cadena: 4635		H1: 71284635
P2: 71623845		sub-cadena: 3845		H2: 12763845
P1: 76582314		sub-cadena: 31		H1: 56847312
P2: 35684172		sub-cadena: 17		H2: 65823174
P1: 14728653		sub-cadena: 5		H1: 83271456
P2: 83271546		sub-cadena: 4		H2: 17286543

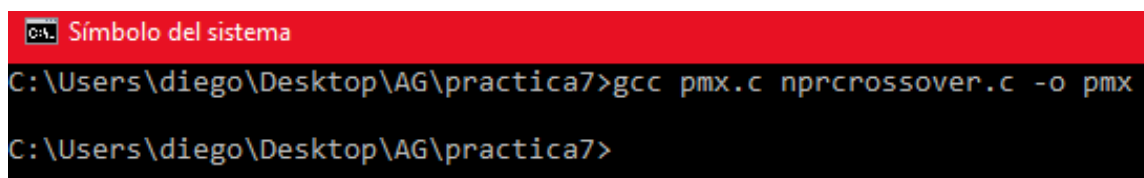
Figura 2. Ejecución – OX1.

Implementación de Partially Mapped Crossover (PMX)

Compilación y Ejecución

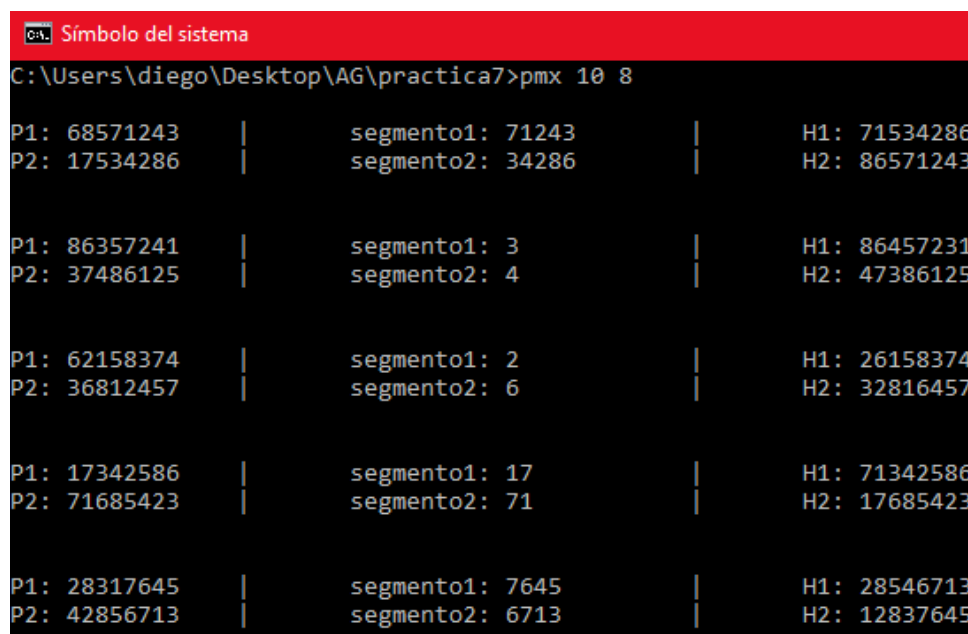
En la *Figura 3* y *Figura 4* se anexan capturas de pantalla con la compilación y ejecución del código de programa desarrollado para el operador PMX. La ejecución se realiza para una población con tamaño igual a 10 y la longitud del cromosoma es igual a 8.

En la *Figura 4* se resumen los resultados obtenidos en 3 columnas. En la primera columna se observan los padres generados de forma aleatoria. La segunda columna contiene los segmentos delimitados por los puntos de cruce generados aleatoriamente, segmentos utilizados para el mapeo. Y en la tercera columna se observa la descendencia obtenida mediante el método de cruce implementado.



```
C:\Users\diego\Desktop\AG\practica7>gcc pmx.c nprcrossover.c -o pmx
C:\Users\diego\Desktop\AG\practica7>
```

Figura 3. Compilación - PMX.



```
C:\Users\diego\Desktop\AG\practica7>pmx 10 8

P1: 68571243 | segmento1: 71243 | H1: 71534286
P2: 17534286 | segmento2: 34286 | H2: 86571243

P1: 86357241 | segmento1: 3 | H1: 86457231
P2: 37486125 | segmento2: 4 | H2: 47386125

P1: 62158374 | segmento1: 2 | H1: 26158374
P2: 36812457 | segmento2: 6 | H2: 32816457

P1: 17342586 | segmento1: 17 | H1: 71342586
P2: 71685423 | segmento2: 71 | H2: 17685423

P1: 28317645 | segmento1: 7645 | H1: 28546713
P2: 42856713 | segmento2: 6713 | H2: 12837645
```

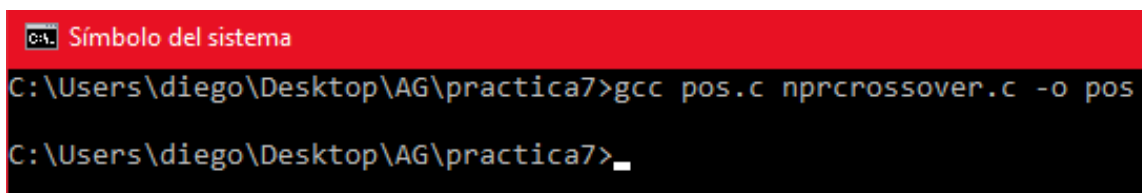
Figura 4. Ejecución - PMX.

Implementación de Position-Based Crossover (POS)

Compilación y Ejecución

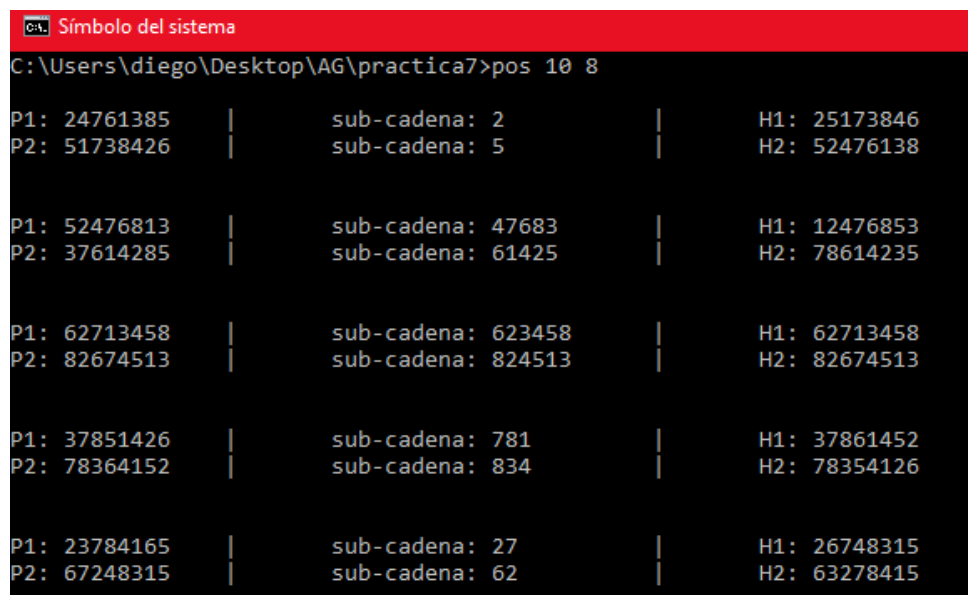
En la *Figura 5* y *Figura 6* se anexan capturas de pantalla con la compilación y ejecución del código de programa desarrollado para el operador POS. La ejecución se realiza para una población con tamaño igual a 10 y la longitud del cromosoma es igual a 8.

En la *Figura 6* se resumen los resultados obtenidos en 3 columnas. En la primera columna se observan los padres generados de forma aleatoria. La segunda columna contiene los elementos pertenecientes a las posiciones elegidas aleatoriamente. Y en la tercera columna se observa la descendencia obtenida mediante el método de cruce implementado.



```
C:\Users\diego\Desktop\AG\practica7>gcc pos.c nprcrossover.c -o pos
C:\Users\diego\Desktop\AG\practica7>
```

Figura 5. Compilación - POS.



```
C:\Users\diego\Desktop\AG\practica7>pos 10 8

P1: 24761385 | sub-cadena: 2 | H1: 25173846
P2: 51738426 | sub-cadena: 5 | H2: 52476138

P1: 52476813 | sub-cadena: 47683 | H1: 12476853
P2: 37614285 | sub-cadena: 61425 | H2: 78614235

P1: 62713458 | sub-cadena: 623458 | H1: 62713458
P2: 82674513 | sub-cadena: 824513 | H2: 82674513

P1: 37851426 | sub-cadena: 781 | H1: 37861452
P2: 78364152 | sub-cadena: 834 | H2: 78354126

P1: 23784165 | sub-cadena: 27 | H1: 26748315
P2: 67248315 | sub-cadena: 62 | H2: 63278415
```

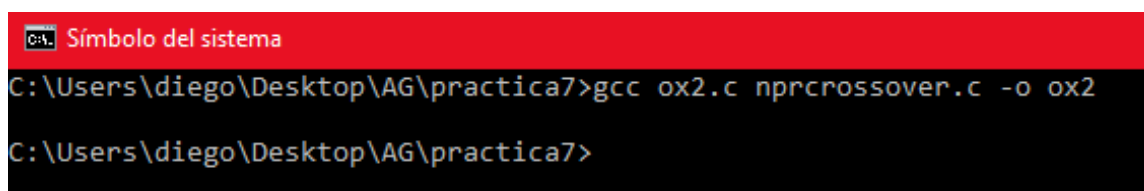
Figura 6. Ejecución - POS.

Implementación de Order-Based Crossover (OX2)

Compilación y Ejecución

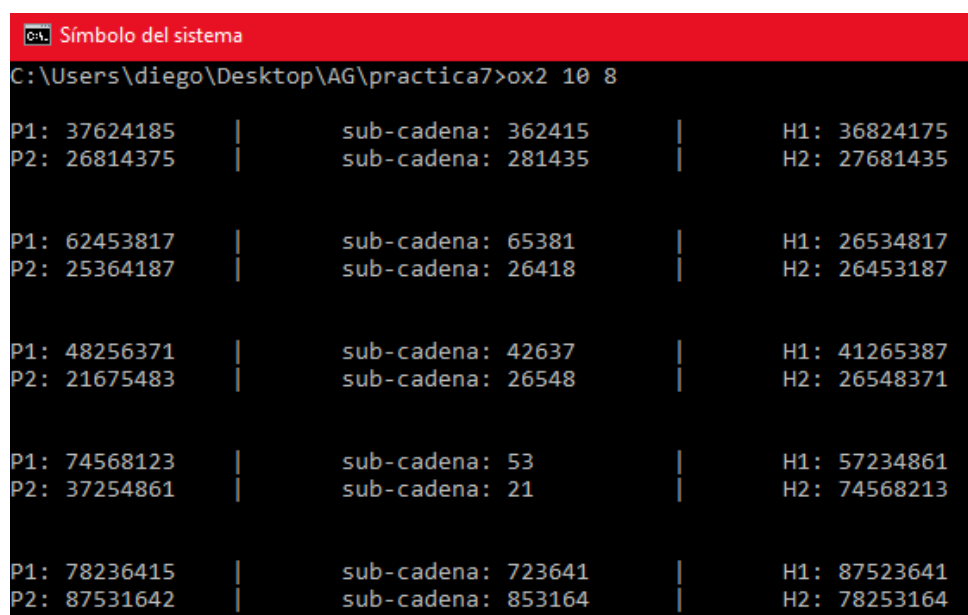
En la *Figura 7* y *Figura 8* se anexan capturas de pantalla con la compilación y ejecución del código de programa desarrollado para el operador OX2. La ejecución se realiza para una población con tamaño igual a 10 y la longitud del cromosoma es igual a 8.

En la *Figura 8* se resumen los resultados obtenidos en 3 columnas. En la primera columna se observan los padres generados de forma aleatoria. La segunda columna contiene los valores pertenecientes a las posiciones elegidas de forma aleatoria. Y la tercera columna contiene la descendencia obtenida.



```
C:\Users\diego\Desktop\AG\practica7>gcc ox2.c nprcrossover.c -o ox2  
C:\Users\diego\Desktop\AG\practica7>
```

Figura 7. Compilación – OX2.



```
C:\Users\diego\Desktop\AG\practica7>ox2 10 8  
  
P1: 37624185 | sub-cadena: 362415 | H1: 36824175  
P2: 26814375 | sub-cadena: 281435 | H2: 27681435  
  
P1: 62453817 | sub-cadena: 65381 | H1: 26534817  
P2: 25364187 | sub-cadena: 26418 | H2: 26453187  
  
P1: 48256371 | sub-cadena: 42637 | H1: 41265387  
P2: 21675483 | sub-cadena: 26548 | H2: 26548371  
  
P1: 74568123 | sub-cadena: 53 | H1: 57234861  
P2: 37254861 | sub-cadena: 21 | H2: 74568213  
  
P1: 78236415 | sub-cadena: 723641 | H1: 87523641  
P2: 87531642 | sub-cadena: 853164 | H2: 78253164
```

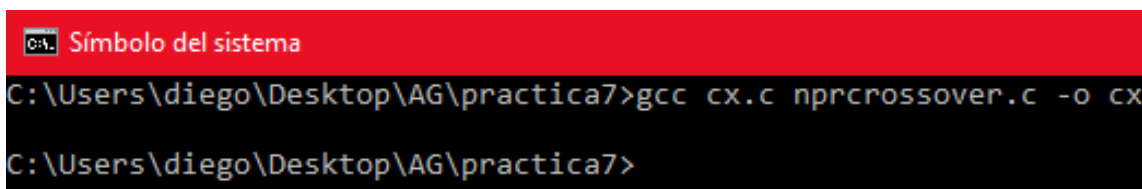
Figura 8. Ejecución – ox2.

Implementación de Cycle Crossover (CX)

Compilación y Ejecución

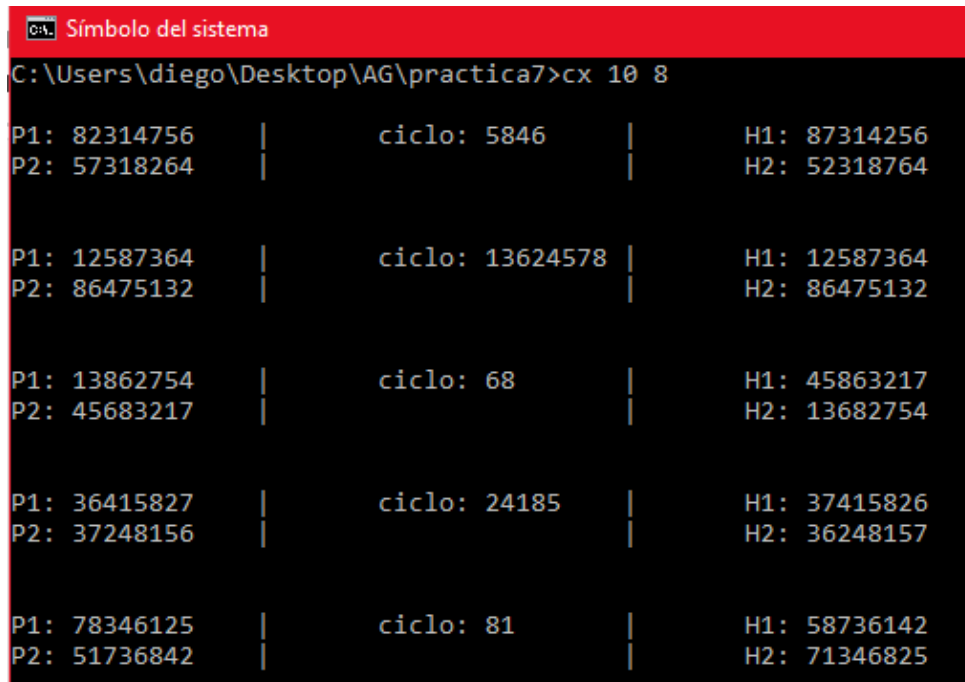
En la *Figura 9* y *Figura 10* se anexan capturas de pantalla con la compilación y ejecución del código de programa desarrollado para el operador CX. La ejecución se realiza para una población con tamaño igual a 10 y la longitud del cromosoma es igual a 8.

En la *Figura 10* se resumen los resultados obtenidos en 3 columnas. En la primera columna se observan los padres generados de forma aleatoria. La segunda columna contiene los elementos pertenecientes al ciclo generado mediante una posición inicial aleatoria. Y la tercera columna contiene la descendencia obtenida.



```
C:\Users\diego\Desktop\AG\practica7>gcc cx.c nprcrossover.c -o cx
C:\Users\diego\Desktop\AG\practica7>
```

Figura 9. Compilación - CX.



```
C:\Users\diego\Desktop\AG\practica7>cx 10 8

P1: 82314756 | ciclo: 5846 | H1: 87314256
P2: 57318264 | | H2: 52318764

P1: 12587364 | ciclo: 13624578 | H1: 12587364
P2: 86475132 | | H2: 86475132

P1: 13862754 | ciclo: 68 | H1: 45863217
P2: 45683217 | | H2: 13682754

P1: 36415827 | ciclo: 24185 | H1: 37415826
P2: 37248156 | | H2: 36248157

P1: 78346125 | ciclo: 81 | H1: 58736142
P2: 51736842 | | H2: 71346825
```

Figura 10. Ejecución - CX.

Conclusión

A través de esta práctica se ha resaltado la necesidad de implementar una representación de cromosoma diferente a la vista en prácticas anteriores dado ciertos problemas en particular. Se ha mencionado, además, que al implementar ciertos tipos de representación de cromosoma los operadores de cruce vistos en la *Práctica No. 6* pueden resultar en soluciones inválidas, por lo que se requieren operadores de cruce específicos para la representación a utilizar que entreguen soluciones válidas. Por ejemplo, en el caso específico de esta práctica, se implementaron cuatro operadores de cruce para representación de permutaciones.