Trabajo Práctico N°1

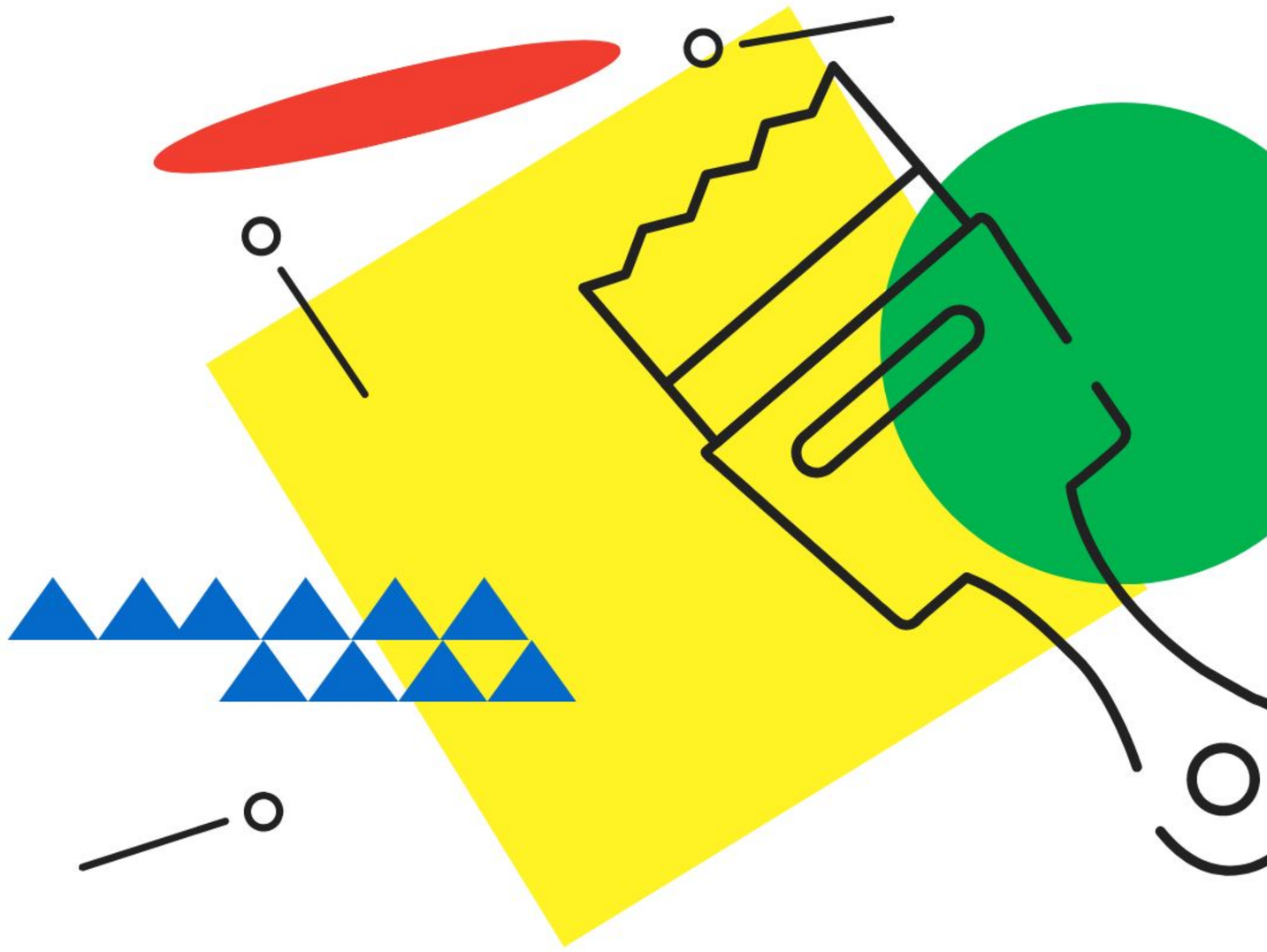
Métodos de Búsqueda + Algoritmos Genéticos

GRUPO 6

Luciana Diaz Kralj
Gonzalo Nicolas Rossin
João Nuno Diegues Vasconcelos
Mafalda Colaço Parente Morais Da Costa

Lado A: Métodos de Búsqueda.

TP N°1



01

Ejercicio 1:

8-Puzzle

Del tablero al azar al tablero solución

8-puzzle

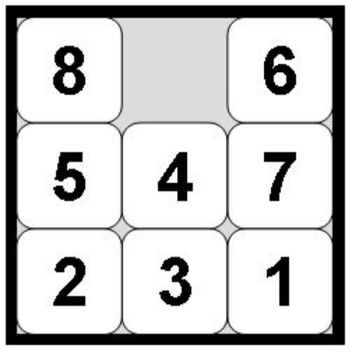
8-puzzle consiste de un tablero de 3*3 con ocho fichas numeradas y un espacio en blanco, donde queremos llegar al tablero solución.

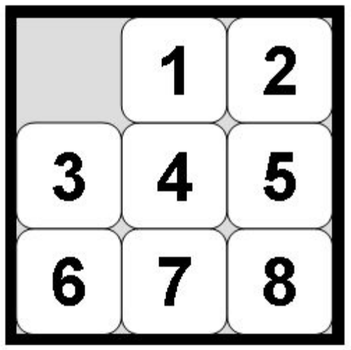
Nº de movimientos posibles en caso de espacio blanco		
2	3	2
3	4	3
2	3	2

Acciones: Moviendo los números adyacentes al espacio vacío.
Izquierda, derecha, arriba, abajo, dependiendo de la ubicación del mosaico.
Branching factor de 2,66.

Estados: Hay 9! estados posibles, pero solo 9!/2 estados desde donde es posible llegar a la solución.

Condición de terminación: Los números en el tablero están ordenados.





8-puzzle

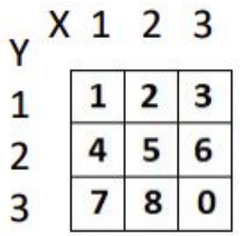
Estructura:

Una matriz 3*3, la posición es un espacio que tiene el número y un tuplo (Xs,Ys), la posición en la matriz.

Guardar la posición del espacio vacío como forma de mejorar la eficiencia.

Condiciones para poder moverse:

Nombre de movimiento	Condicíon prévia	Costo
Arriba	Ys > 1	1
Abajo	Ys < 3	1
Izquierda	Xs > 1	1
Derecha	Xs < 3	1



Heuristica 1

Fuera de Lugar

Cantidad de números que están fuera de su posición correcta.

Admissible una vez que no sobreestima es costo real.

5	7	3
8	2	
1	6	4

Solo el número 3 se encuentra en la posición correcta

Entonces el valor de la heurística = 8 - 1 = 7

Heuristica 2

Distancia de Manhattan

Distancia de los números que están fuera de su posición correcta: distancia horizontal + distancia vertical

Admissible una vez que no sobreestima es costo real.

1	7	3
8	2	4
	6	5

El número 7 y 2 se encuentran en la posición incorrecta.

Heuristica de 7: horizontal 1
vertical 2

Total de 7:
horizontal + vertical = 3

Heurística total: 4

Heuristica 3

Números incorrectos en línea y columna.

Admissible una vez que no sobreestima es costo real.

1	7	2
8	3	4
	6	5

El número 7, 2 y 3 se encuentran en la posición incorrecta.

Heurística 2ª columna : 2
Heurística 3ª columna: 1
Heurística 1ª línea: 2
Heurística 2ª línea: 1

Heurística total: 6

Metodos de busqueda

Métodos de búsqueda informados

Para este problema, dado que conocemos el estado objetivo y hemos desarrollado heurísticas, creemos que la mejor opción es utilizar métodos de búsqueda informados, de modo que deje atrás el uso de BFS y DFS, métodos de búsqueda desinformados.

Greedy search

Con el Greedy Search simplemente elegimos el nodo que tiene el menor costo, eligiendo el Optimal Local Path.

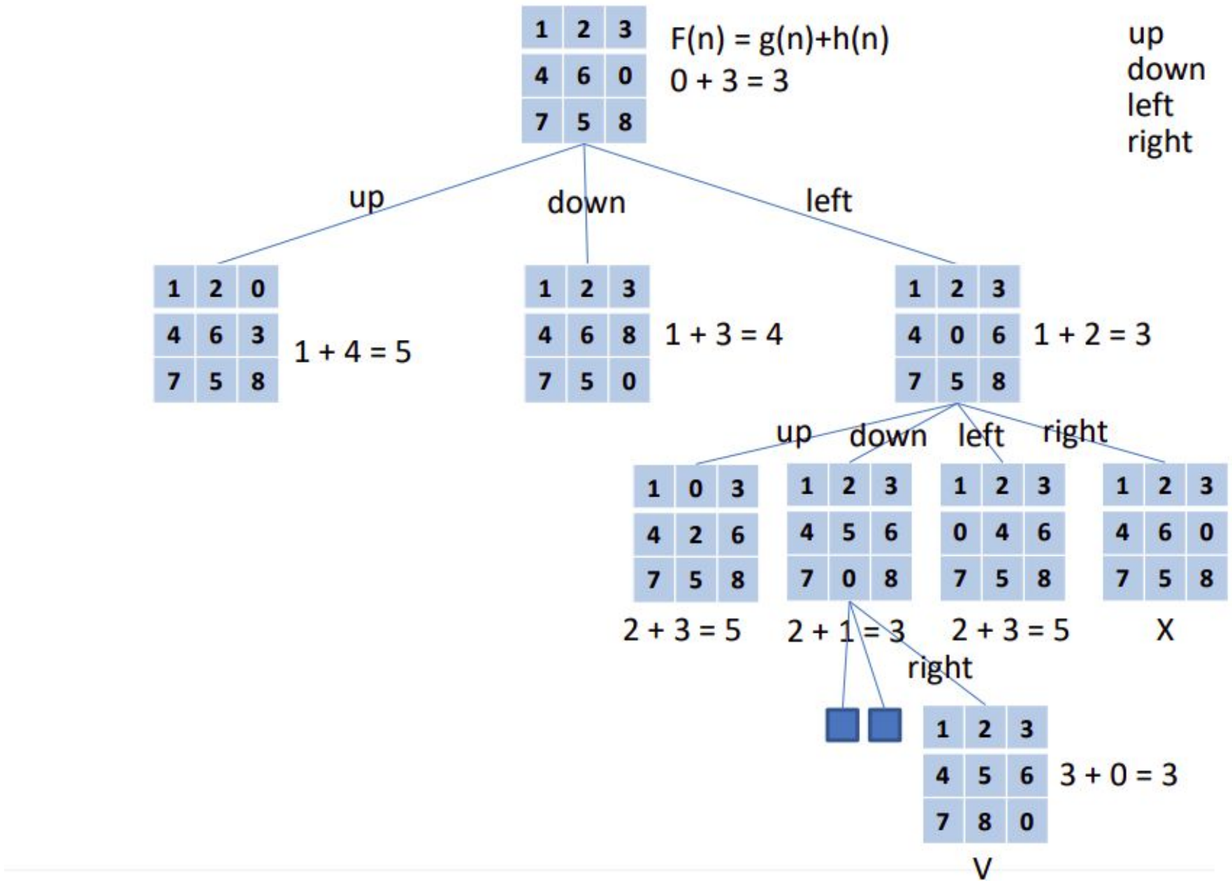
Metodos de busqueda

A*

Algoritmo que se utiliza ampliamente en la búsqueda de rutas y en el recorrido de grafos, el proceso de trazar una ruta transversal entre múltiples puntos, denominados nodos. Reconocido por su rendimiento y precisión, goza de un uso generalizado.

A* sería lo mejor porque considera el camino anterior. Para eso, usa un priority queue con la suma del costo a llegar a ese nodo más el costo establecido por la heurística.

A* ejemplo



Usando la primera heurística

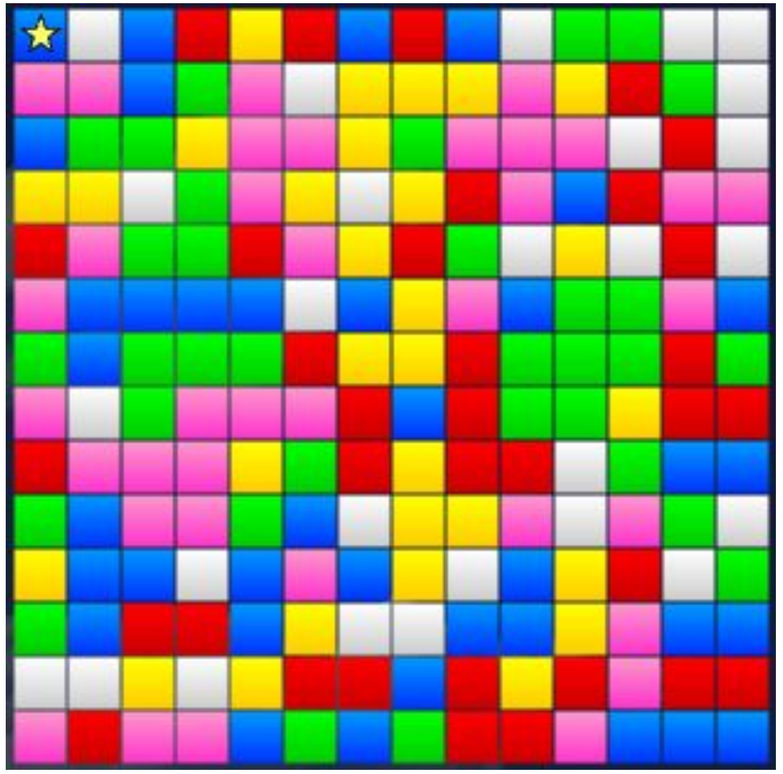
02

Ejercicio 2: Fill-Zone

La forma rápida de pintar toda la grilla

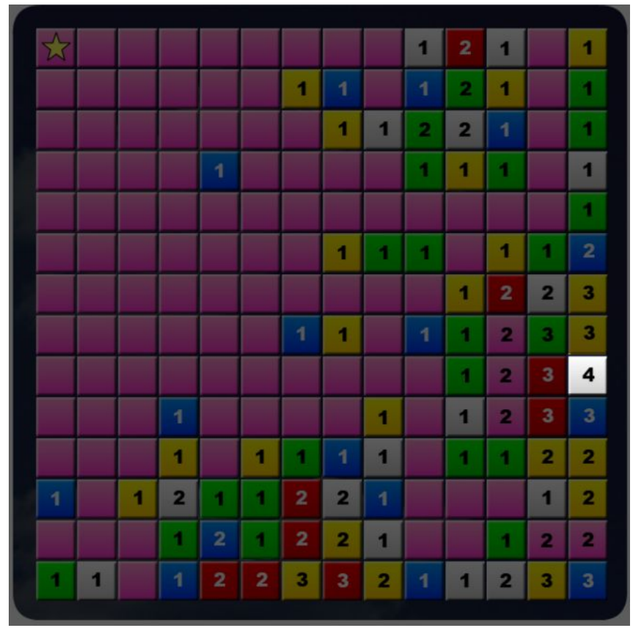
Fill-Zone

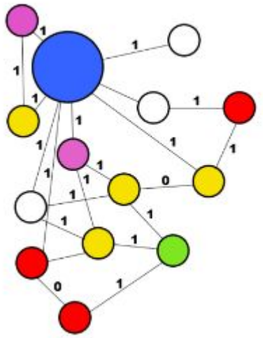
- El juego consiste en lograr que toda la grilla tenga el mismo color.
- Para esto, debemos cambiar el color de las celdas controladas por el jugador.
- Serán unidas todas las celdas adyacentes que sean del mismo color.



Heurística 1: Bronson Distance

- Estima la cantidad de pasos restantes para terminar el tablero en base a la cantidad de turnos necesarios para alcanzar la celda más alejada del tablero.
- Para conseguir este valor se realiza un grafo de las celdas no controladas y por medio de Dijkstra se obtiene el valor.
- Es admisible dado que para terminar el tablero se deberá llegar a este casillero y la cantidad de pasos para hacerlo no podrá ser inferior al valor calculado.

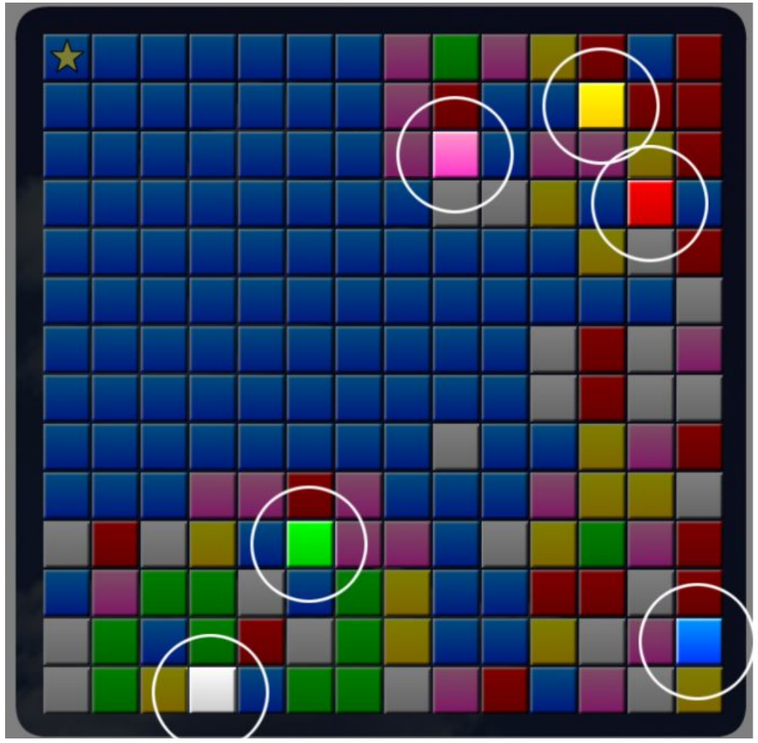




Heurística 2: Remaining colors

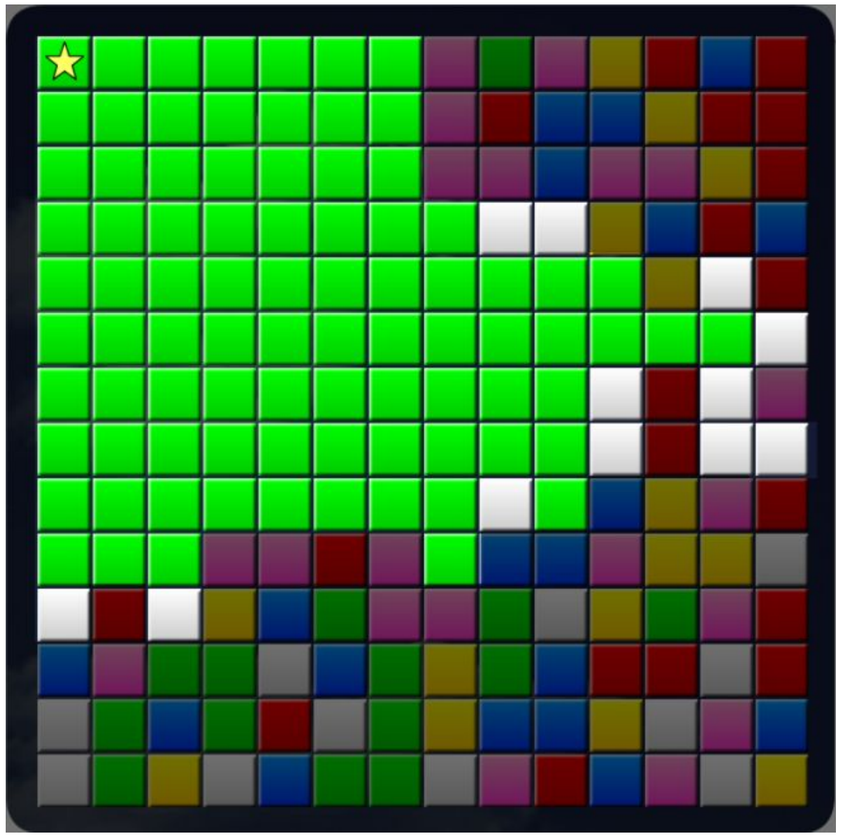
- Estima la cantidad de pasos restantes para terminar el tablero en base a la cantidad de colores diferentes no controlados restantes en el tablero.

- Es admisible ya que se debe cambiar al menos 1 vez a cada color no controlado para completar el juego.

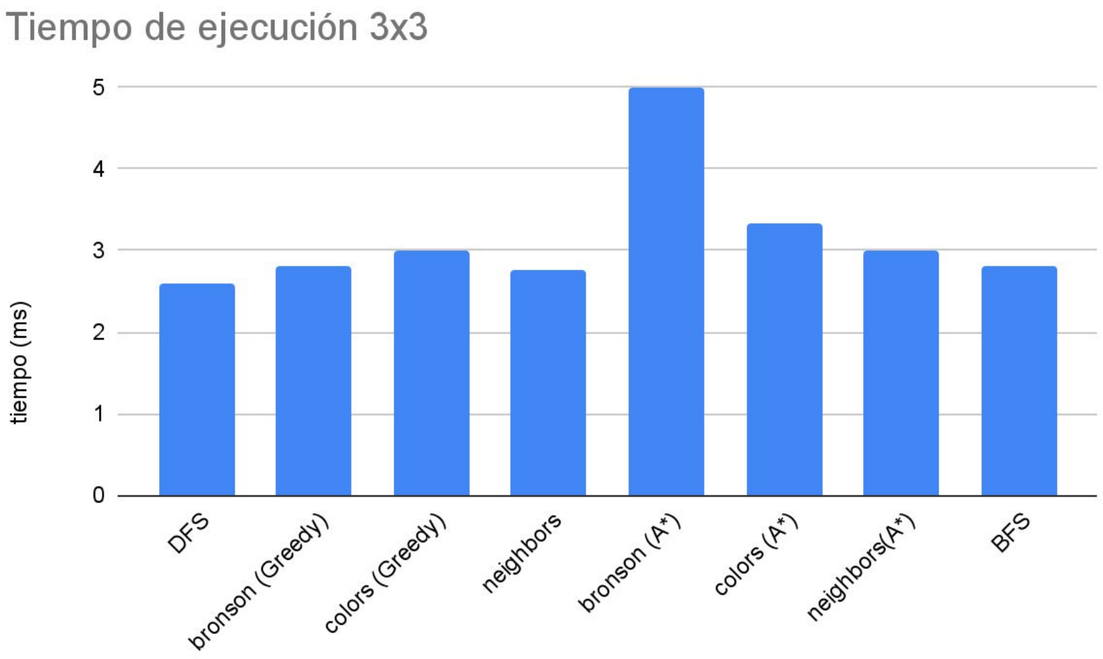


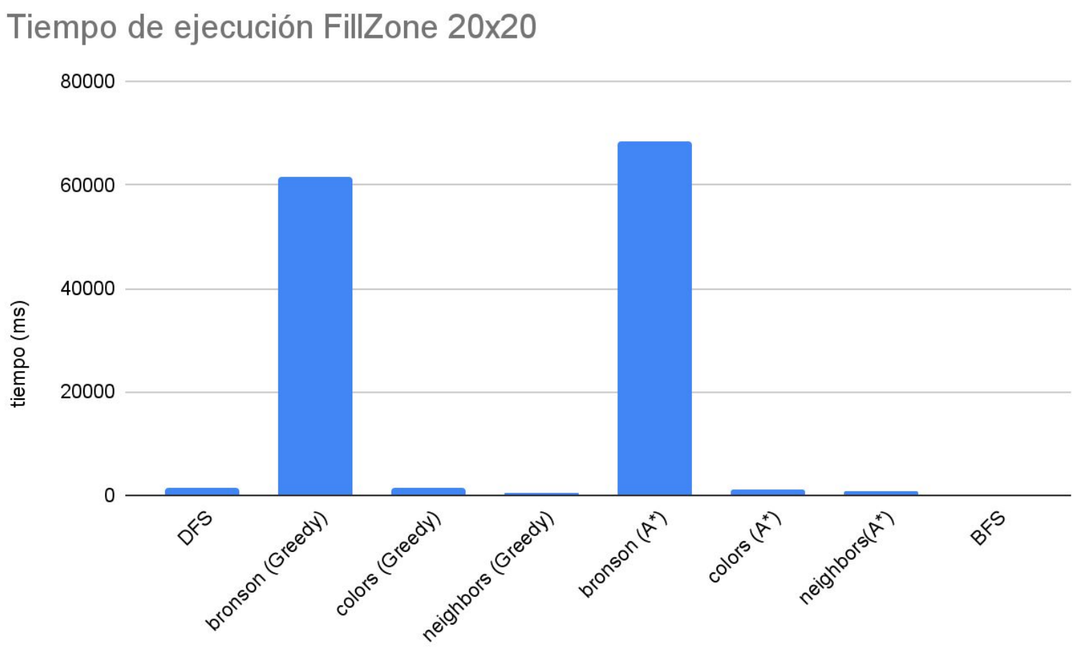
Heurística 3: Most Neighbors

- Consiste en elegir el camino que me permita reducir en mayor medida la cantidad de celdas no controladas.
- No es admisible ya que no me permite estimar la cantidad de turnos restantes requeridos para completar el juego.



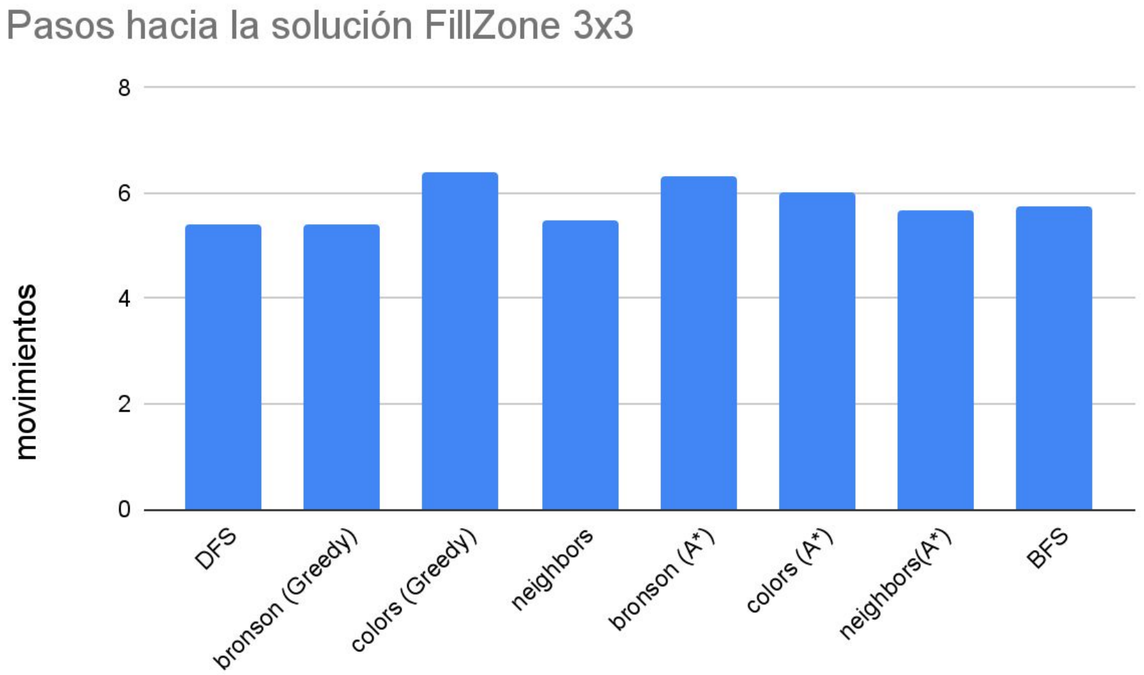
Resultados: tiempo de ejecución

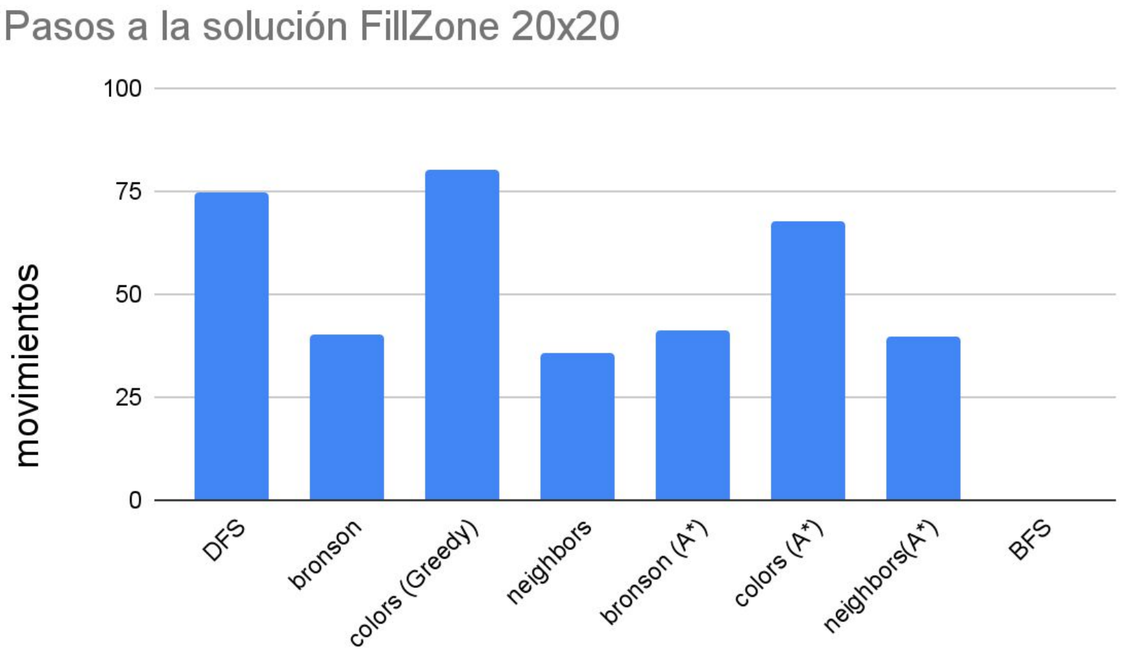




Podemos concluir que el algoritmo con el mayor tiempo de ejecución es el que utiliza la heurística de Bronson.

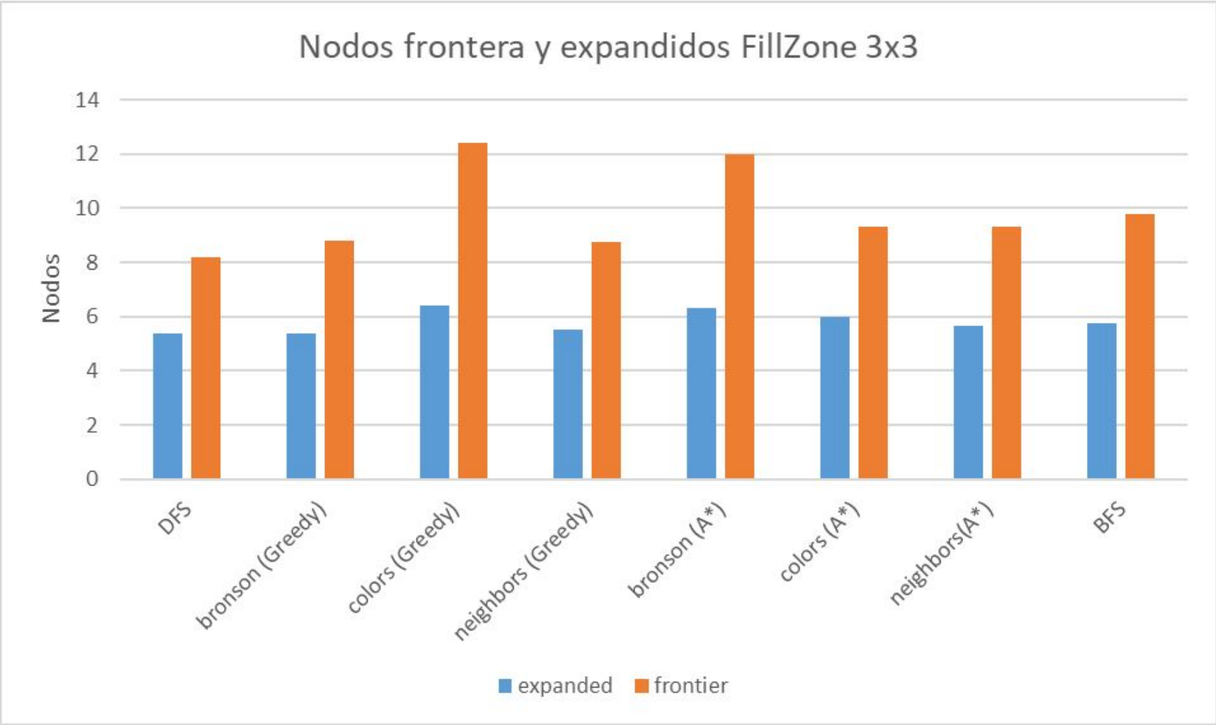
Resultados: pasos

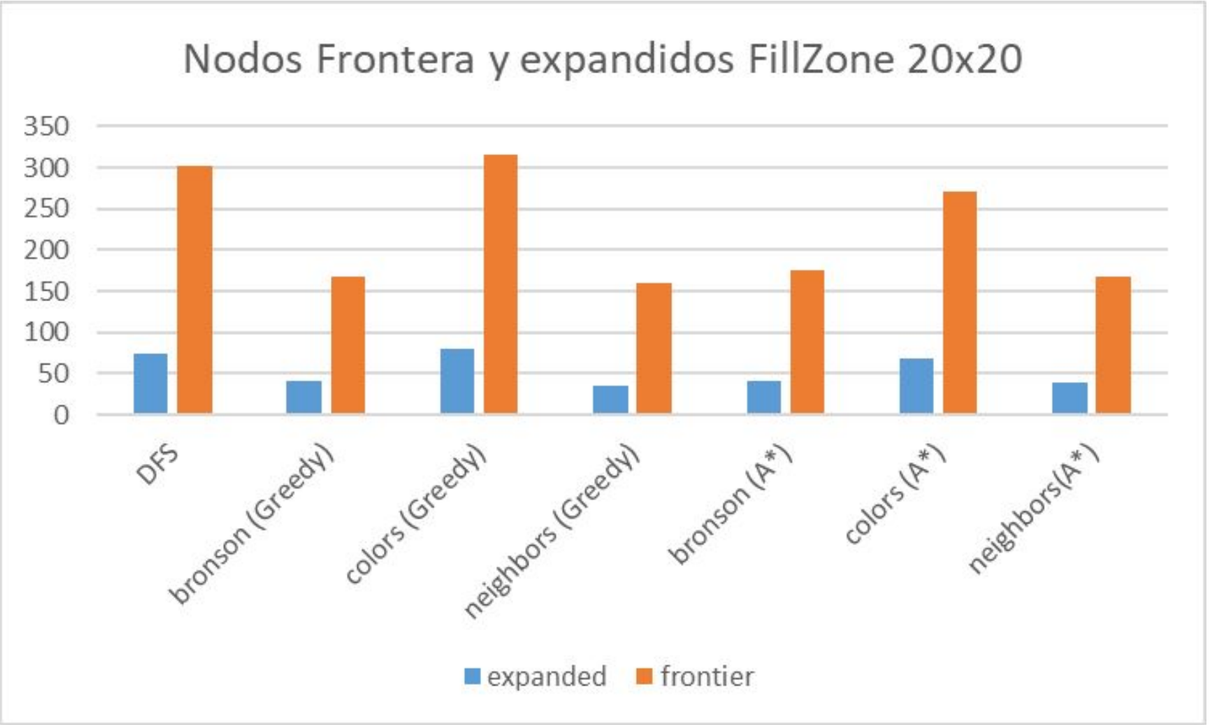




Podemos ver que si bien todos comienzan con una cantidad similar de pasos en tableros pequeños, al aumentar la dimensión los algoritmos que requieren mayor cantidad de pasos son DFS y los que utilizan la heurística Remaining Colors.

Resultados: nodos frontera y expandidos





Se puede observar que al aumentar las dimensiones del tablero los algoritmos DFS y los que utilizan la heurística de remaining colors utilizan mayor cantidad de nodos.

Resultados: Tabla de comparación

	3x3				14X14				20X20			
	t(ms)	steps	expanded	frontier	t (ms)	steps	expanded	frontier	t (ms)	steps	expanded	frontier
DFS	2,6	5,4	5,4	8,2	342,2	42,2	42,2	154,4	1285,4	74,6	74,6	301,4
bronson (Greedy)	2,8	5,4	5,4	8,8	9613,4	27,8	27,8	109	61706,4	40	40	167,8
colors (Greedy)	3	6,4	6,4	12,4	219,8	27,2	27,2	112,2	1384	80,2	80,2	316,4
neighbors (Greedy)	2,75	5,5	5,5	8,75	221,75	28,75	28,75	116,75	579,5	35,75	35,75	159,75
bronson (A*)	5	6,333333333	6,333333333	12	9730,666667	30,33333333	30,33333333	116,6666667	68399	41,33333333	41,33333333	175,6666667
colors (A*)	3,333333333	6	6	9,333333333	306,6666667	38	38	149,3333333	1153,666667	67,66666667	67,66666667	270,6666667
neighbors(A*)	3	5,666666667	5,666666667	9,333333333	201	26	26	105	647,3333333	39,66666667	39,66666667	167,6666667
BFS	2,8	5,733333333	5,733333333	9,8	---	---	---	---	---	---	---	---

nota: durante la ejecución del algoritmo BFS para tableros de 14x14 y superior se debió suspender la ejecución debido a que este no terminaba

Lado B: Algoritmos Genéticos.

TP N°1

01

Ejercicio 1:

ASCII art

Recreando imágenes

Problema ASCII art

Objetivo

A partir de una imagen cuadrada, representar de la mejor manera posible dicha imagen en un mapa de NxN caracteres ASCII.

Gen

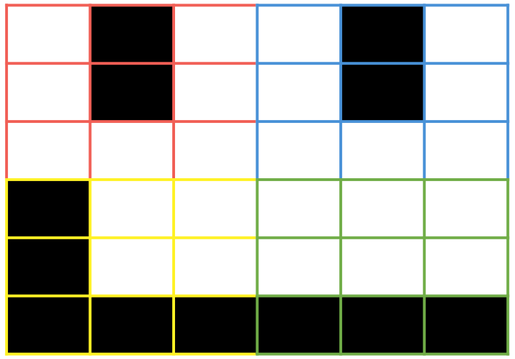
Se define un gen para cada sección cuadrada de MxM pixeles de la imagen.

Alelos

Valor decimal de un caracter ASCII. Hay 255 elementos en la tabla ASCII extendida, pero para representar una imagen los caracteres de control no sirven. Consideremos entonces 223 caracteres ASCII, con el espacio “ “ incluído.

Representación

Imagen cuadrada
de 6x6 pixeles



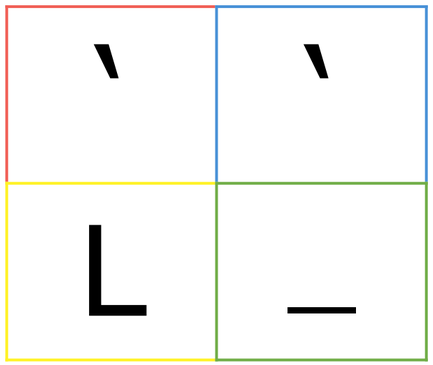
Genotipo

Gen 1	Gen 2	Gen 3	Gen 4
[32-255]	[32-255]	[32-255]	[32-255]

Ejemplo de cromosoma

Gen 1	Gen 2	Gen 3	Gen 4
39	39	76	95

\ \ L _



Cálculo de fitness

M_{imagen} de la imagen input

0	0	0	0	0	0
1	0	1	1	0	1
1	0	1	1	0	1
0	0	0	0	0	0
0	1	1	1	1	0
0	0	0	0	0	0

Ejemplo de cromosoma

Gen 1	Gen 2	Gen 3	Gen 4
84	67	84	79

$$M = \sqrt{\frac{N \cdot N}{\#genes}}$$

Elijo #genes tal que la división devuelva un entero

M_{ascii} Gen 1

0	0	0
1	0	1
1	0	1

M_{ascii} Gen 2

0	0	0
0	1	1
0	0	0

M_{ascii} Gen 3

0	0	0
1	0	1
1	0	1

M_{ascii} Gen 4

0	0	0
0	1	0
0	0	0

<u>Al iniciar el algoritmo</u>

Paso 1: Convierto la imagen de NxN a blanco y negro.

Paso 2: Genero una matriz M_{imagen} de NxN, asignando 0 a los píxeles negros y 1 a los blancos.

<u>Cálculo de fitness</u>

Paso 1: Obtengo la matriz M_{ascii} de MxM, mapa de píxeles del ASCII seleccionado, para cada gen.

Paso 2: Genero la matriz extendida con las matrices M_{ascii} de cada gen, de dimensión NxN

Paso 3: Defino fitness en base a la similaridad de las dos matrices mediante la Norma de Frobenius.

Implementación teórica

Selección por Torneos Determinísticos

Rápido de ejecutar, no usa variables intermedias, ni usa toda la población.

Crossover de dos puntos

Combino secciones de los padres que probablemente ya son buenas aproximaciones a la imagen deseada.

Mutación

Evito el problema de los máximos locales, intercambiando de lugar a los valores de los genes.

02

Ejercicio 2:
Color perfecto

Recreando colores en la paleta del artista

Generación de colores

Objetivo

A partir de una paleta de colores, buscar la mejor forma de mezclar proporciones de los diferentes colores para llegar a un color objetivo.

Gen

Se define un gen para cada color de la paleta de colores de entrada.

Alelos

Proporción del color en la mezcla, entre 0 y 1.

Restricción: La suma de los valores de cada gen debe ser = 1.

Representación

Color objetivo

RGB
[0 255 255]

Genotipo

Gen 1	Gen 2	Gen 3	Gen 4
[0-1]	[0-1]	[0-1]	[0-1]

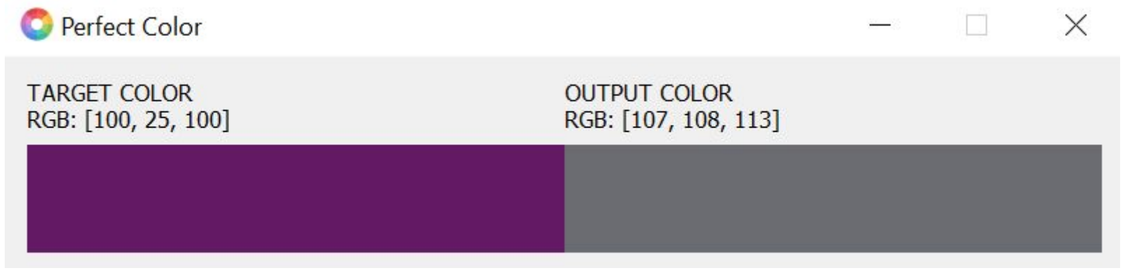
Restricción: La suma de los valores de cada gen debe ser = 1

Ejemplo de cromosoma

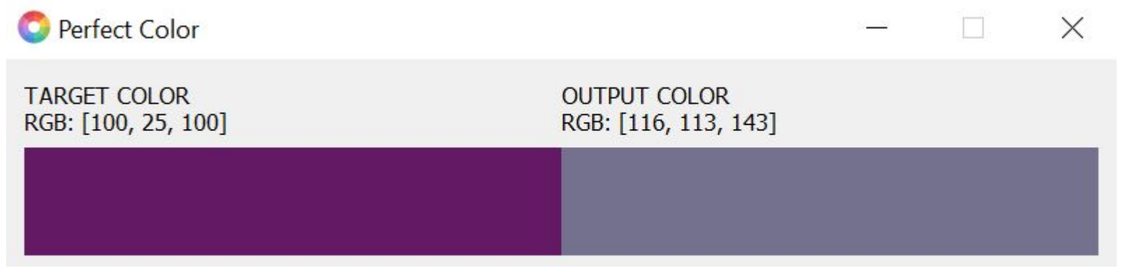
Gen 1	Gen 2	Gen 3	Gen 4
0.XX	0.XX	0.XX	0.XX

RGB
[0 x x]

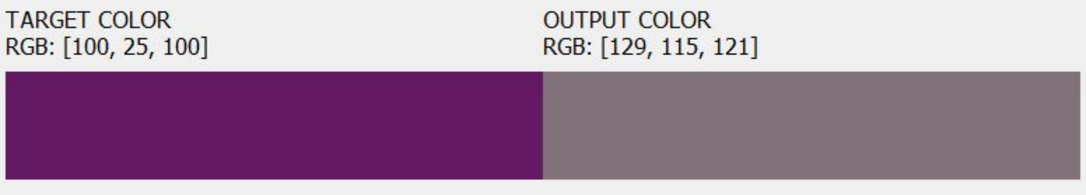
Usando det-tournament como método de selección



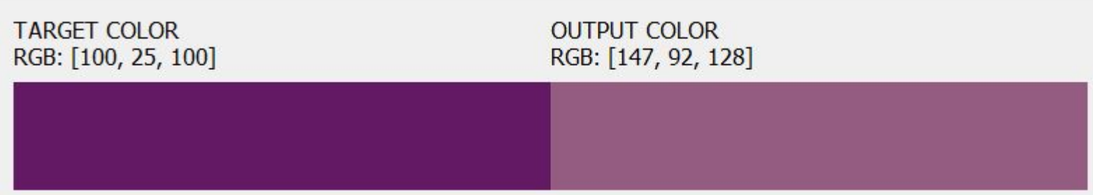
Usando roulette como método de selección



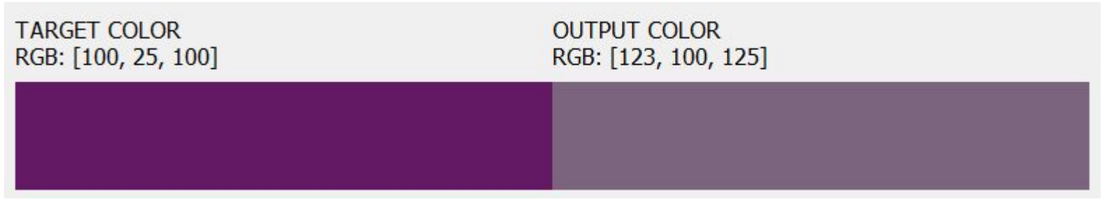
Usando det-tournament como método de selección
individuals_k=20
max_num_generations=20

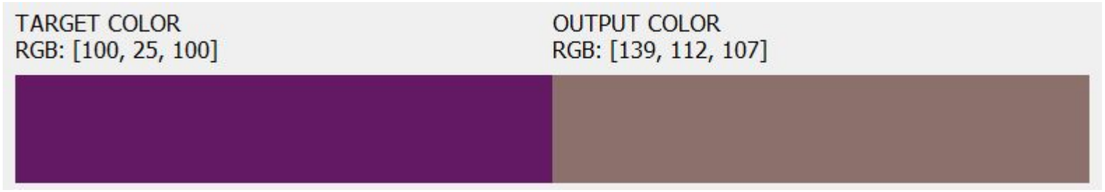


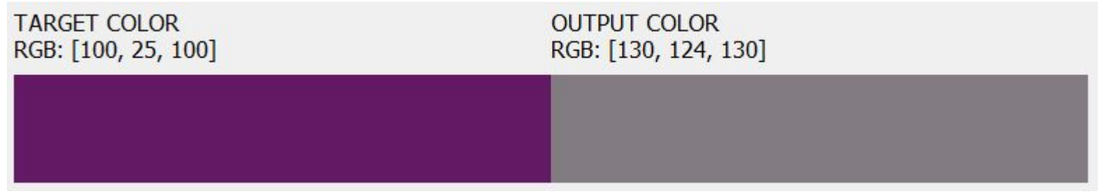
Usando det-tournament como método de selección
individuals_k=80
max_num_generations=20

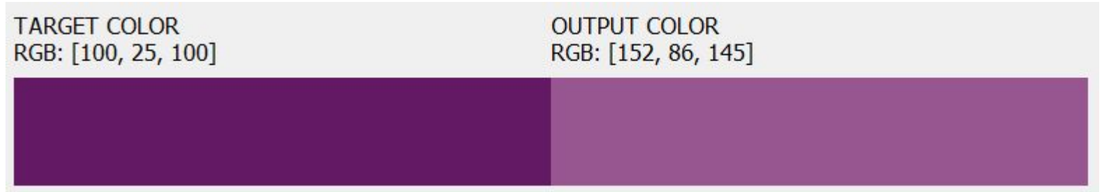


```
"color_palette_file": "color_palette.txt",

"hyperparameters" : {
    "implementation": "fill-all",

    "___selection_options": "elite, roulette, det-tournament",
    "selection": "det-tournament",
    "population_n": 100,
    "individuals_k": 80,

    "_torneos": "",
    "individuals_m": 100,
    "treshold": 0.5,

    "crossover": "one-p",
    "cross_prob": 0.9,

    "mutation": "",
    "mut_prob": 0.9,

    "cut_criteria": {
        "___cut_options": "generations, treshold, exhaustion",
        "method": "exhaustion",
        "max_num_generations": 20,
        "delta_treshold": 1
    }
}
```









```
"hyperparameters" : {
    "implementation": "fill-all",

    "___selection_options": "elite, roulette, det-tournament",
    "selection": "det-tournament",
    "population_n": 100,
    "individuals_k": 80,

    "_torneos": "",
    "individuals_m": 100,
    "treshold": 0.5,

    "crossover": "one-p",
    "cross_prob": 0.9,

    "mutation": "",
    "mut_prob": 0.9,

    "cut_criteria": {
        "___cut_options": "generations, treshold, exhaustion",
        "method": "exhaustion",
        "max_num_generations": 100,
        "delta_treshold": 1
    }
}
```