Paradigmas de Programación
Licenciatura en Ciencia de la Computación, USACH
Profesor: Pablo Pérez Lantero

---

# Trabajo de Laboratorio 1
# Apareando puntos coloreados con rectángulos

(Fecha de entrega: 19 de mayo de 2019)

## 1- Introducción

El objetivo de este trabajo es desarrollar en `Python` un algoritmo heurístico para aparear la mayor cantidad posible de puntos con rectángulos. Se utilizarán los archivos de código `match.py` y `visual.py` adjuntos a este documento. El algoritmo recibirá como entrada el total $n$ de puntos, generará un conjunto de $n$ puntos aleatorios utilizando las funciones del archivo `match.py`, ejecutará la parte heurística para aparear la mayor cantidad posible de puntos, y finalmente retornará el porciento de los puntos que pudieron ser apareados. Se puede usar el archivo `visual.py` para visualizar el conjunto de puntos y el apareamiento correspondiente. Se espera poder aparear más del 90% de los puntos. Detalles de cada una de estas cosas aparecen en las secciones siguientes.

## 2- ¿Qué es aparear con rectángulos?

Sea $P$ un conjunto de $n$ puntos en el plano, donde no existen en $P$ dos puntos con la misma coordenada $x$ o la misma coordenada $y$. Es decir, no hay dos puntos en la misma vertical o en la misma horizontal. Cada punto tiene asignado un color: rojo o azul. Para cualesquiera dos puntos distintos $p, q \in P$, se define $R(p, q)$ como el rectángulo de menor área, con los lados paralelos a los ejes de coordenadas, que cubre o contiene tanto a $p$ como a $q$. Notar que:

$$R(p, q) \;=\; [left, right] \times [bottom, top],$$

sabiendo que $left = \min(x(p), x(q))$, $right = \max(x(p), x(q))$, $bottom = \min(y(p), y(q))$, y $top = \max(y(p), y(q))$. Aquí $x(p)$ y $y(p)$ denotan, respectivamente, las coordenadas $x$ e $y$ de $p$ (lo mismo para $q$ y cualquier otro punto).

Los puntos $p, q \in P$, $p \neq q$, pueden ser apareados (mediante el rectángulo $R(p, q)$) si es que $p$ y $q$ tienen el mismo color y además el rectángulo $R(p, q)$ no contiene ningún otro punto de $P$. Además, para cualesquiera cuatro puntos distintos $p, q, p', q' \in P$, tales que los colores de $p$ y $q$ coinciden, y los colores de $p'$ y $q'$ también coinciden, podemos aparear simultáneamente la dupla $p, q$ y la dupla $p', q'$ si y solo si los rectángulos $R(p, q)$ y $R(p', q')$ son disjuntos.

Aparear con rectángulos los puntos de $P$ se refiere a encontrar duplas de puntos de $P$ (la mayor cantidad de duplas si es posible) de manera tal que cada punto participe en a lo más una dupla y que todas las duplas encontradas puedan ser apareadas simultaneamente.

## 3- Objetivo del trabajo

El objetivo del trabajo es implementar en `Python` una función que recibe como parámetro a $n$, genera un conjunto de $n$ puntos aleatorios, busca (en un tiempo de cómputo prudencial) la mayor cantidad que pueda de duplas de puntos que se puedan aparear de manera simultánea, y retorna qué porciento de los puntos aparecen en las duplas. Notar que este porciento es igual a dos veces el total de duplas dividido por el total $n$.

Para generar los $n$ puntos aleatorios se utlizará la función `create_random_points(n)` que viene dentro de `match.py`. Esta función genera una lista de $n$ puntos, donde las coordenadas $x$ de los puntos están en el conjunto $\{0, 1, \ldots, n-1\}$ al igual que las coordenadas $y$, sin que exitan dos puntos con la misma $x$ o la misma $y$. Cada punto generado es un objeto de la clase

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
```

La clase `Point` viene en `match.py`. Si hacemos `P = match.create_random_points(n)` y luego `p = P[0]`, entonces `p.color` es el color de `p` que siempre será un entero (1 o $-1$). En las lista `P` los elementos no están necesariamente ordenados por alguna coordenada.

Para crear un rectángulo, correspondiente al apareamiento de una dupla `p` y `q`, se utilizará la siguiente clase que viene en `match.py`:

```
class Rectangle:
    def __init__(self, left, right, bottom, top):
        self.left = left
        self.right = right
        self.bottom = bottom
        self.top = top
```

Por ejemplo, para crear el rectángulo $[0, 4] \times [1, 6]$ se puede hacer `r = match.Rectangle(0,4,1,6)`. O mejor, para crear el rectángulo $R(p, q)$ podemos escribir

```
  r = match.Rectangle(min(p.x,q.x), max(p.x,q.x), min(p.y,q.y), max(p.y,q.y))
```

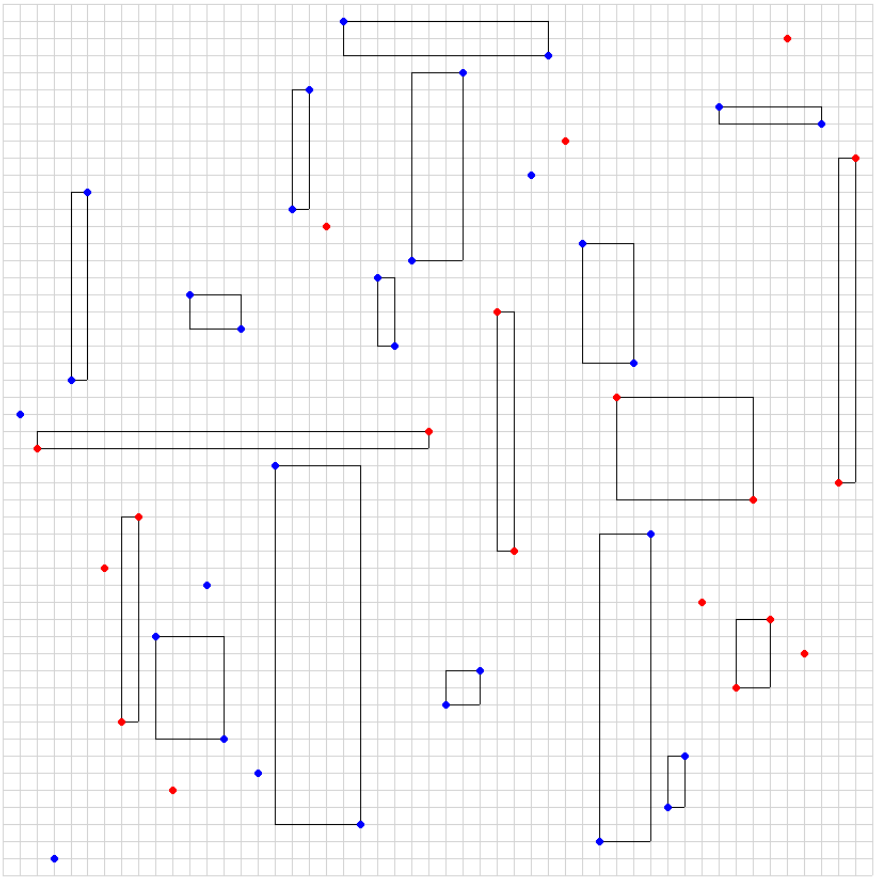
A modo de resumen, un prototipo de implementación es el siguiente:

```
  import match
  import visual

  def aparear(n):
      P = match.create_random_points(n)
      R = heuristica(P)
      return 2 * len(R) / n

  def heuristica(P):
```

```
10        # TODO
```

La función `heuristica(P)` es la encargada de buscar en un tiempo de cómputo razonable, la mayor cantidad de duplas que pueda de manera tal que puedan ser simultáneamente apareadas. El resultado es la lista `R` de objetos `match.Rectangle`, correspondientes a cada una de las duplas.

A modo de probar la implementación, se puede visualizar el resultado de la función `heuristica(P)` utilizando el archivo `visual.py`. Para ello hacer `visual.Window(points=P,rectangles=R)` y se abrirá una ventana con un dibujo como el siguiente:



Una vez en la ventana, usar el ratón para desplazar y escalar el dibujo (Sorry, la ventana no es user-friendly!). Observar del ejemplo que de un total de $n = 50$ puntos, se aparearon 38, por lo que la función `aparear(n)` devuelve $38/50 = 0.76$, correspondiente al 76%.

## 4- Requerimientos

La función `aparear(n)` deberá recibir valores bien grandes de `n`, por ejemplo $n = 10000$. En un tiempo prudencial se deben encontrar las duplas. El algoritmo a pensar y desarrollar, correspondiente al código de la función `heuristica(P)`, deberá ser capaz de aparear más del 90% de los puntos, y excelente que se logren porcientos cercanos al 100.

## 5- Reglas del Juego

(1) El trabajo se realizará en equipos de a dos. La nota será única, es decir, los miembros del equipo reciben todos la misma nota. Para aprobar el trabajo es necesario haber implementado satisfactoriamente el programa, que cada integrante del equipo haya participado en la solución y domine la solución, y que además la solución sea original.

(2) La fecha de entrega es el día **domingo 19 de mayo de 2019, hasta las 23:59 horas**. La fecha es **impostergable**. La entrega se hará por correo electrónico a la dirección del profesor. En la siguiente semana cada equipo de alumnos expondrá el trabajo realizado al profesor/ayudante en la oficina. Entregas fuera de fecha serán rechazadas.

(3) En el programa tanto las estructuras de datos como los algoritmos tienen que ser eficientes, y esto suma puntos a la nota del trabajo.

(4) En caso de ser necesario, sujeto a una explicación detallada del porqué, se pueden agregar nuevas funcionalidades a las clases `match.Point`, `match.Rectangle` y `visual.Window`.