# Lista de Computação para Arquitetura

LAMO / FAU / UFRJ

Pedro Maciel Xavier
pedromxavier@poli.ufrj.br

April 8, 2020

## 1 Intersecção de Retângulos

Uma maneira simples de representar retângulos em um computador é através de um par de pontos, onde cada ponto é um par ordenado $(x, y)$. Por convenção, o primeiro ponto indica o canto superior esquerdo do retângulo; e o segundo, o canto inferior direito.

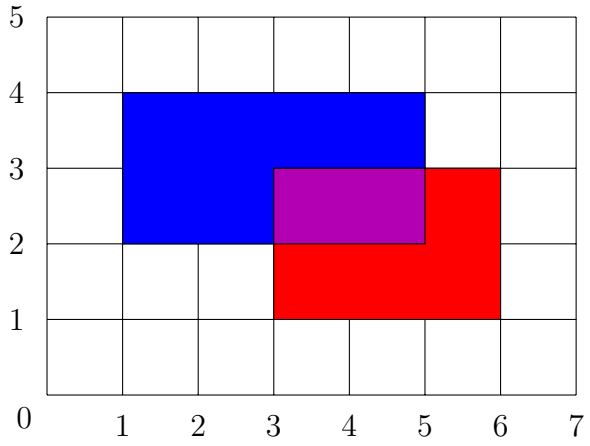


Figure 1: Retângulos

Assim, o retângulo **azul** pode ser descrito pelos pares $(1, 4)$ e $(5, 2)$, enquanto

o retângulo **vermelho** é dado pelos pontos $(3, 3)$ e $(6, 1)$. A intersecção entre os dois é justamente a área **lilás** que se encontra entre os pontos $(3, 3)$ e $(5, 2)$.

**Desafio:** Fazer uma função que, dados dois retângulos, retorna a intersecção entre eles, ou seja, um outro retângulo ou `None`, caso não haja sobreposição.

**Exemplo:**

```
>>> A = (1, 4, 5, 2)
>>> B = (3, 3, 6, 1)
>>> C = intersec(A, B)
>>> print(C)
(3, 3, 5, 2)
```

## 2 Anos bissextos

A humanidade sempre teve problemas com a contagem dos anos, uma vez que estes duram 365,24 dias. O calendário Juliano, que vigorou de 45 a.C. até 1582 d.C., introduziu o uso dos anos bissextos acrescentou um dia a cada 3 anos. O erro só foi percebido décadas depois e o acréscimo foi abandonado. Isso fez com que se acumulasse um erro de cerca de 10 dias até o momento da transição para o calendário Gregoriano. Por conta disso, os dias entre 4 e 15 de outubro de 1582 simplesmente não existiram.

Com este novo calendário foi definida a nova regra para o cálculo dos anos bissextos:

- De 4 em 4 anos é ano bissexto.
- De 100 em 100 anos não é ano bissexto.
- De 400 em 400 anos é ano bissexto.
- Prevalecem as últimas regras sobre as primeiras.

**Desafio:** Sabendo que o ano 2000 foi bissexto, crie uma função que, dado um ano, informe se ele é bissexto ou não, retornando `True` ou `False`, respectivamente.

**Exemplo:**

```
>>> bissexto(2000)
True
>>> bissexto(2100)
False
>>> bissexto(2104)
True
```

## 3 Coordenadas polares

Estamos acostumados a pensar em coordenadas cartesianas na hora de descrever a geometria de um determinado objeto. No entanto, o sistema de coordenadas deve ser escolhido conforme o cenário em que se está trabalhando.

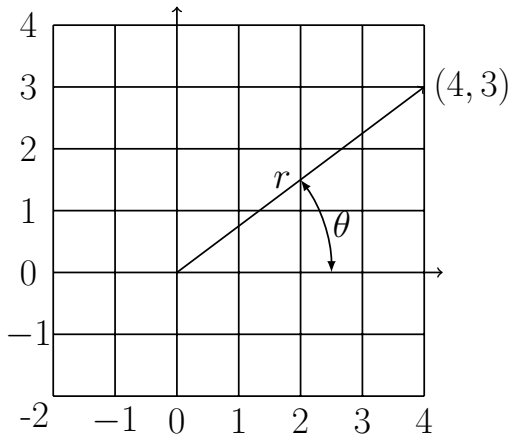


Figure 2: Coordenadas cartesianas e polares.

O ponto $(4, 3)$, quando escrito em coordenadas polares, nos dá:

$$r = \sqrt{4^2 + 3^2} = \sqrt{16 + 9} = \sqrt{25} = 5$$
$$\theta = \arctan \frac{3}{4} = 0.01123 \text{ rad} \approx 36.87^\circ$$

**Desafio:** Construa duas funções: `polar(x, y)` levará um ponto em coordenadas cartesianas $(x, y)$ para a forma polar $(r, \theta)$ e `cart(r, theta)`, que fará o caminho contrário.

**Exemplo:**

```
1 >>> polar(-1, 0)
2 (1, 3.141592653589793)
3 >>> cart(2, math.pi)
4 (-2, 0)
```

## 4 Sequência de *Collatz*

A sequência de *Collatz* é obtida aplicando sucessivamente a função

$$f(n) = \begin{cases} 3n+1, & \text{se } n \text{ for ímpar} \\ n \div 2, & \text{se } n \text{ for par} \end{cases}$$

Por exemplo, começamos com $n = 26$. Após sucessivas aplicações temos:

$$26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Isso nos dá uma sequência com 11 números. 40 é maior que 26, mas sua sequência só teria 9 números.

Ainda não se sabe se todos os números induzem uma sequência que termina em 1. No entanto, até agora não foi encontrado um número sequer em que isso não tenha acontecido!

**Desafio:** Faça uma função que calcule o comprimento da sequência gerada a partir de um número natural $n$ qualquer.

**Exemplo:**

```
>>> collatz(26)
11
>>> collatz(40)
9
>>> collatz(1)
1
```