# Teoria da coputacao q2.2018

João Carlos Pandolfi Santana

June 2018

## 1 Cálculo de complexidade

A complexidade do algoritmo é baseada nos elementos que os compõe. O algoritmo de *euclides extendido* para *MDC* tem complexidade logarítmica.

$$T(n) = O(logn)$$

A complexidade do restante do algoritmo após o o cálculo do *MDC*, $m$ e $n$ é constante, portanto se mantém a complexidade calculada anteriormente.

### Python

Código implementando algoritmos

```
1
# ------ FUNCOES -------
3
# Funcao para calcular o MDC
5 def euclides(a, b):
    r = a;
7   r1 = b;
    u = 1;
9   v = 0;
    u1 = 0;
11  v1 = 1;

13  while (r1 != 0):
      q = int(r / r1); # pega apenas a parte inteira
15    rs = r;
      us = u;
17    vs = v;
      r = r1;
19    u = u1;
      v = v1;
21    r1 = rs - q *r1;
      u1 = us - q*u;
23    v1 = vs - q*v1;

25
    return [r, u, v]; # tais que a*u + b*v = r et r = pgcd (a, b)
27

29 # ------ Codigo -------

31 # -- Lendo dados
  print("Resolvedor de equacoes diophantinas lineares")
```

```
print("Formato: Ax + By = C")
a = int(input("A = "))
b = int(input("B = "))
c = int(input("C = "))


# ---- Calculando GDC ----
gdcAB , n, m = euclides(a,b)

# --- Encontrando uma solucao ---
x0 = (n*c)/gdcAB
y0 = (m*c)/gdcAB

print("\nUma solucao pelo metodo comum")
print("x="+str(x0))
print("y="+str(y0))

# --- Econtrando todas as possiveis solucoes para LDE ---

xn = ((n*c)/gdcAB)
xnk = (b/gdcAB)

yn = ((m*c)/gdcAB)
ynk = (a/gdcAB)

print("\nSolucao usando o metodo do LDE")
print("x = "+str(xn)+" + "+str(xnk)+ "*k")
print("y = "+str(yn)+" - "+str(ynk)+ "*k")
```