Entrega 1

Daniel Brito

1) Dado um grafo direcionado $G = (V, A)$, sendo representado por uma matriz de adjacência, o algoritmo abaixo decide se G possui uma celebridade. A ideia consiste em encontrar um candidato por meio da função *encontrarCandidato*, e depois verificar se ele é realmente uma celebridade ou não, por meio da função *verificarCelebridade*, checando se o candidato conhece ninguém e todos o conhecem. Se o candidato for uma celebridade a função retorna *True*, caso contrário, retorna *False*.

```
def encontrarCandidato(G, n):
    if(n==1):
      return n-1

    candidato = encontrarCandidato(G, n-1)

    if(candidato==-1):
      return n-1

    elif(G[candidato][n-1] and not(G[n-1][candidato])):
      return n-1

    elif(G[n-1][candidato] and not(G[candidato][n-1])):
      return candidato

    return -1

def verificarCelebridade(G, n):

  candidato = encontrarCandidato(G, n)

  if(candidato==-1):
    return False

  u = v = 0

  for i in range(n):
    if (i!=candidato):
      u += G[candidato][i]
      v += G[i][candidato]

  if(u==0 and v==n-1):
    return True
  else:
    return False
```

2) Para mostrar que $(n+a)^b = \Theta(n^b)$, devemos encontrar constantes $c_1, c_2, n_0 > 0$, tal que $0 \leq c_1 \cdot n^b \leq (n+a)^b \leq c_2 \cdot n^b$ para todo $n \geq n_0$. Assim, temos:

$$\begin{aligned} n + a &\leq n + |a| \\ &\leq 2n, |a| \leq n \end{aligned}$$

e

$$\begin{aligned} n + a &\geq n - |a| \\ &\geq \frac{1}{2}n, |a| \leq \frac{1}{2}n \end{aligned}$$

Portanto, quando $n \geq 2 \cdot |a|$, temos:

$$0 \leq \frac{1}{2}n \leq n + a \leq 2n.$$

Como $b > 0$, a inequação ainda é válida, ou seja:

$$\begin{aligned} 0 \leq (\frac{1}{2}n)^b \leq (n+a)^b \leq (2n)^b, \\ 0 \leq (\frac{1}{2})^b \cdot n^b \leq (n+a)^b \leq 2^b \cdot n^b. \end{aligned}$$

Por fim, temos que $c_1 = (\frac{1}{2})^b$, $c_2 = 2^b$ e $n_0 = 2 \cdot |a|$, satisfazendo a definição.

3) Sim. Para mostrar que $2^{n+1} \in O(2^n)$, devemos encontrar constantes $c, n_0 > 0$, tal que $0 \leq 2^{n+1} \leq c \cdot 2^n$, para todo $n \geq n_0$. Como $2^{n+1} = 2 \cdot 2^n$ para todo $n \geq 0$, temos que a definição é satisfeita com $c = 2$ e $n_0 = 1$.

Não. Note que $2^{2n} = 2^n \cdot 2^n$. Desta maneira, para $2^{2n} \in O(2^n)$, precisamos de uma constante c, tal que $0 \leq 2^n \cdot 2^n \leq c \cdot 2^n$. Fica evidente que precisamos de um $c \geq 2^n$. Entretanto, isso não é possível para um valor de n arbitrariamente alto. Ou seja, não importa o valor escolhido para c, para algum valor suficientemente alto de n, c não será suficiente. Portanto, concluímos que $2^{2n} \notin O(2^n)$.

4) Utilizando as definições de limite, $o(n)$ e $\omega(n)$, temos que:

$$o(g(n)) = \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

e

$$\omega(g(n)) = \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

Entretanto, ambos os casos não se mantêm verdadeiros quando n se aproxima de ∞. Desta maneira, a interseção é vazia.

5) Sejam os itens:
a) $n! \in \omega(2^n)$:

$$\begin{aligned} n! &= n \cdot (n-1) \cdot ... \cdot 2 \cdot 1 \\ &> 2 \cdot 2 \cdot ... \cdot 2, n \geq 4 \\ &= 2^n \end{aligned}$$

Desta forma, existe uma constante positiva $c = 1$ e $n_0 = 4$, na qual $n! > c \cdot 2^n$, para todo $n \geq n_0$.

b) $n! \in o(n^n)$:
Com base na definição, temos $0 \leq f(n) < c \cdot g(n)$, para todo $c > 0$ e $n \geq n_0$. Seja $c = 1$ e $n_0 = 1$.

$$\begin{aligned} \text{Caso Base}: \quad & n = 1 \\ & n! \leq n^n \\ & 1! \leq 1^1 \\ & 1 \leq 1 \\ \text{Hipótese de indução}: \quad & \text{Assuma que } k! \leq k^k, k > 1 \\ \text{Passo indutivo}: \quad & \text{Queremos provar que } (k+1)! \leq (k+1)^{(k+1)}. \\ & k! \leq k^k \\ & (k+1)(k!) \leq (k+1) \cdot (k^k) < \\ & (k+1)! \leq \leq c(k+1) + 1 \\ & = c(k+1) + O(1) \\ & = c(k+1) \\ & = O(n) \end{aligned}$$

Desta forma, existe uma constante positiva $c = 1$ e $n_0 = 2$, na qual $n! < c \cdot n^n$, para todo $n \geq n_0$.

6) Sejam os itens:

a) Se $k \geq d$, então $p(n) \in O(n^k)$:
Como $k \geq d$, n^k cresce mais rápido ou no mesmo ritmo que n^d para um n suficientemente grande. Desta maneira, $p(n) = O(n^k)$.
Para c, temos $0 \leq p(n) \leq 1.5 \cdot a_d \cdot n^d \leq 1.5 \cdot a_d \cdot n^k$.
Assim, se tomarmos $c_1 = 1.5 \cdot a_d$, temos $0 \leq p(n) \leq c_1 \cdot n^k$, ou seja, $p(n) = O(n^k)$.

b) Se $k \leq d$, então $p(n) \in \Omega(n^k)$:

Como $k \leq d$, n^k cresce mais devagar ou no mesmo ritmo que n^d para um n suficientemente grande. Desta maneira, $p(n) = \Omega(n^k)$.

Para c, temos $0 \leq p(n) \leq 0.5 \cdot a_d \cdot n^d \leq 0.5 \cdot a_d \cdot n^k$.

Assim, se tomarmos $c_1 = 0.5 \cdot a_d$, temos $0 \leq p(n) \leq c_1 \cdot n^k$, ou seja, $p(n) = \Omega(n^k)$.

c) TO-DO
d) TO-DO
e) TO-DO

7) Ordenação crescente, sendo que as funções na mesma linha possuem classe de equivalência iguais:

$$1, n^{\frac{1}{log(n)}}$$
$$2^{log(n)}$$
$$ln(ln(n))$$
$$\sqrt{log(n)}$$
$$ln(n)$$
$$log^2(n)$$
$$2^{\sqrt{2 \cdot log(n)}}$$
$$(\sqrt{2})^{log(n)}$$
$$n$$
$$n \cdot log(n)$$
$$4^{log(n)}, n^2$$
$$n^3$$
$$n^{log(log(n))}, log(n)^{log(n)}$$
$$log(n!)$$
$$(\frac{3}{2})^n$$
$$2^n$$
$$e^n$$
$$n \cdot 2^n$$
$$n!$$
$$(n+1)!$$
$$2^{2^n}$$
$$2^{2^{n+1}}$$

8) Sejam os itens:

a) Falso. Contra-exemplo: Seja $f(n) = n$ e $g(n) = n^2$. Assim, temos que $n = O(n^2)$, mas $n^2 \neq O(n)$.

b) Falso. Contra-exemplo: Seja $f(n) = n$ e $g(n) = n^2$, mas $n^2 + n \neq \Theta(n)$.

c) Verdadeiro. Prova: $f(n) = O(g(n))$, ou seja, $0 \leq f(n) \leq c \cdot g(n)$, para todo $n \geq n_0$, com $c, n_0 > 0$. Desta forma, $0 \leq log(f(n)) \leq log(c) + log(g(n)) \leq k \cdot log(g(n))$. Portanto, $log(f(n)) = O(log(g(n))$.

d) Falso. Contra-exemplo: Seja $f(n) = 2n$ e $g(n) = n$. Desta forma, $f(n) = O(g(n))$, mas $2^{2n} = 4^n \neq O(2^n)$.

9) Sejam os itens:

a) Tomando como estimativa $T(n) \leq cn = O(n)$.

$$
\begin{aligned}
\text{Caso Base :} \quad & n = 1 \\
& T(1) = 1 \\
\text{Hipótese de indução:} \quad & T(k) \leq ck, \text{para } c > 0, k > 1 \\
\text{Passo indutivo:} \quad & \text{Queremos provar que } T(k+1) \leq c(k+1) = O(n). \\
T(k+1) &= T(k+1-1) + 1 \\
&= T(k) + 1 \\
&\leq c(k+1) + 1 \\
&= c(k+1) + O(1) \\
&= c(k+1) \\
&= O(n)
\end{aligned}
$$

b) Tomando como estimativa $T(n) \leq cn^2 = O(n^2)$.

$$
\begin{aligned}
\text{Caso Base :} \quad & n = 1 \\
& T(1) = 1 \\
\text{Hipótese de indução:} \quad & T(k) \leq ck^2, \text{para } c > 0, k > 1 \\
\text{Passo indutivo:} \quad & \text{Queremos provar que } T(k+1) \leq c(k+1)^2 = O(n^2). \\
T(k+1) &= T(k+1-1) + k \\
&= T(k) + k \\
&\leq c(k+1)^2 + k \\
&= c(k+1)^2 + O(n) \\
&= c(k+1)^2 \\
&= O(n^2)
\end{aligned}
$$

c) Para esta recorrência, temos $a = 1$, $b = 2$, $f(n) = 1$, e $n^{\log_b^a} = n^{\log_2^1} = n^0$. Então, podemos aplicar o caso 2 do Teorema Mestre. Assim:

$$T(n) = O(n^{\log_b^a} \log n) = O(n^0 \cdot \log n) = O(\log n).$$

d) Para esta recorrência, temos $a = 3$, $b = 2$, $f(n) = n$, e $n^{\log_b^a} = n^{\log_2^3} \simeq n^{1.585}$ (TO-DO)

e) Para esta recorrência, temos $a = 4$, $b = 3$, $f(n) = n$, e $n^{\log_b^a} = n^{\log_3^4} \simeq n^{0.792}$ (TO-DO)

f) Para esta recorrência, temos $a = 4$, $b = 2$, $f(n) = n^2$, e $n^{\log_b^a} = n^{\log_2^4} = n^2$. Então, podemos aplicar o caso 2 do Teorema Mestre. Assim:

$$T(n) = O(n^{\log_b^a} \log n) = O(n^2 \cdot \log n).$$

g) TO-DO

10) Sejam os itens:

a) Para esta recorrência, temos $a = 2$, $b = 4$, $f(n) = 1$, e $n^{\log_b^a} = n^{\log_4^2}$. Como $f(n) = 1 = O(n^{\log_4^{2-\epsilon}})$, onde $\epsilon = 0.2$, podemos aplicar o caso 1 do Teorema Mestre, e concluir que:

$$T(n) = \Theta(n^{\log_4^2} \log n) = \Theta(n^{\frac{1}{2}}) = \Theta(\sqrt{n}).$$

b) Para esta recorrência, temos $a = 2$, $b = 4$, $f(n) = \sqrt{n}$, e $n^{\log_b^a} = n^{\frac{1}{2}} = \sqrt{n}$. Como $f(n) = \Theta(\sqrt{n})$, podemos aplicar o caso 2 do Teorema Mestre, e concluir que:

$$T(n) = \Theta(\sqrt{n} \cdot \log n).$$

c) Para esta recorrência, temos $a = 2$, $b = 4$, $f(n) = n$, e $n^{\log_b^a} = n^{\log_4^2}$. Como $f(n) = \Omega(n^{\log_4^{2+\epsilon}})$, onde $\epsilon = 0.2$, podemos aplicar o caso 3 do Teorema Mestre. Assim, temos que provar que $af(\frac{n}{b}) \leq cf(n)$, para alguma constante $c < 1$, e todos os valores suficientemente altos de n. Logo, como $af(\frac{n}{b}) = 2(\frac{n}{4}) = \frac{n}{2} \leq cf(n)$, para $c = 0.7$ e $n \geq 2$, podemos concluir que:

$$T(n) = \Theta(f(n)) = \Theta(n).$$

d) Para esta recorrência, temos $a = 2$, $b = 4$, $f(n) = n^2$, e $n^{\log_b^a} = n^{\log_4^2}$. Como $f(n) = \Omega(n^{\log_4^{2+\epsilon}})$, onde $\epsilon = 1$, podemos aplicar o caso 3 do Teorema Mestre. Assim, temos que provar que

$af(\frac{n}{b}) \leq cf(n)$, para alguma constante $c < 1$, e todos os valores suficientemente altos de n. Logo, como $af(\frac{n}{b}) = 2(\frac{n}{4})^2 = (\frac{1}{8})n^2 \leq cf(n)$, para $c = 0.5$ e $n \geq 4$, podemos concluir que:

$$T(n) = \Theta(n^2).$$

e) Para esta recorrência, temos $a = 1$, $b = 2$, $f(n) = 1$, e $n^{\log_b^a} = n^{\log_2^1} = n^0$. Então, podemos aplicar o caso 2 do Teorema Mestre. Assim:

$$T(n) = \Theta(n^{\log_b^a} \log n) = \Theta(n^0 \cdot \log n) = \Theta(\log n).$$

11) Para esta recorrência, temos $a = 4$, $b = 2$, $f(n) = n^2 \cdot \log n$, e $n^{\log_b^a} = n^{\log_2^4} = n^2$. Assintoticamente falando, $f(n) = n^2 \cdot \log n > n^2$, mas não polinomialmente. Portanto, não podemos aplicar o Teorema Mestre neste caso. (TO-DO)

12) Para escolher qual dos algoritmos utilizar, precisamos fazer a seguinte análise:

1) $5T(\frac{n}{2}) + n$
Para esta recorrência, temos $a = 5$, $b = 2$, $f(n) = n$, e $n^{\log_b^a} = n^{\log_2^5} = O(n^{2.322})$.

2) $2T(n-1) + 1$
Tomando como estimativa $T(n) \leq 2^n = O(2^n)$.

$$\begin{aligned}
\text{Caso Base :} \quad & n = 1 \\
& T(1) \leq 2 \\
\text{Hipótese de indução:} \quad & T(k) \leq 2^k, \text{para } k > 1 \\
\text{Passo indutivo:} \quad & \text{Queremos provar que } T(k+1) \leq 2^{(k+1)} = O(2^{(k+1)}). \\
T(k+1) &= 2T(k+1-1) + 1 \\
&= 2T(k) + 1 \\
&\leq 2 \cdot 2^k + 1 \\
&= 2^{(k+1)} + O(1) \\
&= 2^{(k+1)} \\
&= O(2^{(k+1)})
\end{aligned}$$

3) $9T(\frac{n}{3}) + n^2$
Para esta recorrência, temos $a = 9$, $b = 3$, $f(n) = n^2$, e $n^{\log_b^a} = n^{\log_3^9} = O(n^2 \log n)$.

Portanto, com base nesta análise, o algoritmo (1) deveria ser escolhido para resolver o problema.