exemplo p/ quadro comparativo

$f_1 : 2^{\pi}$

$f_2 : 2^{n}$

$f_3 : n \log(n)$

$f_4 : \log(n)$

$f_5 : 100n^2 + 15000n$

$f_6 : n + \log(n)$

$f_7 : n^2$

$f_8 : n$

O:

$0 \leq \log(n) \leq c * n$

• usando derivadas

$k \cdot \ln(n) \leq c * n \iff$

$\ln(n) \leq c_1 \cdot n$

note que

$$\frac{d}{dn} \ln(n) = \frac{1}{n}$$

$$\frac{d}{dn} c_1 * n = c_1$$

tomando $c_1 = 2$, $\ln(n)$ cresce mais devagar que $2*n$ - uma vez $2*n \geq \ln(n)$, não há como $\ln(n)$ passar $2*n$. (Note que isso ocorre p/ $n = 1$, logo,

para $c_1 = kc = 2$ e $n_0 = 1$, $\ln(n) \in O(n)$

usando exponencial:

$\log(n) \leq c * n \iff$ b: base do log

$n \leq b^{c*n} = (b^c)^n = d^n \iff$ d: b^c

$n \leq d^n$

enquanto n cresce em 1 unidade, d^n é multiplicado por d. Se $d > 1$, e uma vez que d^n seja maior que n, d^n não é mais superado por n. ~~Sabe-se que a conversão de bases é feita por um produto~~. Note que apenas bases maiores que 1 nos interessam, logo qualquer $c \geq 1$ nos satisfaz. Tome $c = 2$ e $n_0 = 1$.

usando limites e L'Hopital

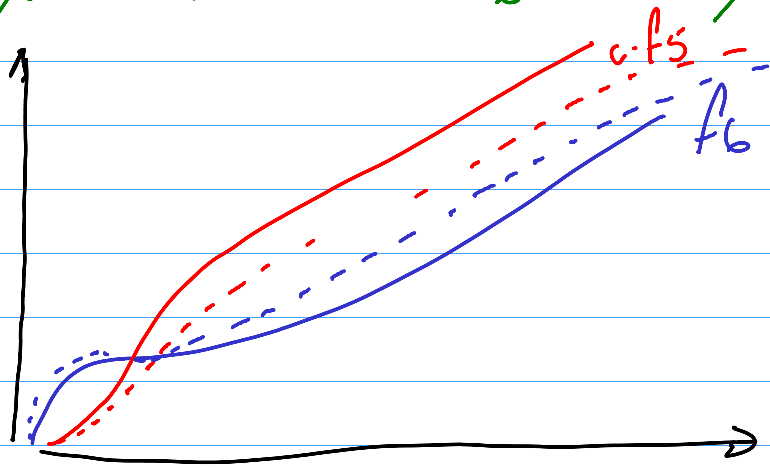
$$\lim_{n \to \infty} \left(\frac{\log(n)}{n} \right) = \lim_{n \to \infty} \left(\frac{\frac{d}{dn} \log(n)}{\frac{d}{dn} n} \right) = \lim_{n \to \infty} \left(\frac{k}{n} \right) = 0$$

pelos três métodos, $\log(n) \in O(n)$, pelo último, $\log(n) \in o(n)$ $\blacksquare$

$f_5 : 100n^2 + 15000n$
$f_6 : n + \log(n)$

neste caso temos que prestar atenção nas "direções" das aproximações.

• queremos demonstrar que $f_6 \in O(f_5)$



se mostrarmos que algo menor que f5 é maior que algo maior que f6, então está ok.

$g(n) = 100n^2$ é menor que f_5

$h(n) = 2n$ é maior que f_6 (por causa da demonstração anterior)

Como sabemos que $h(n) \in O(g(n))$ consequentemente $f_6 \in O(f_5)$ □

Busca Binária

```
int binaria(int valor, int[] vetor, int esq, int dir) {
        int meio = (esq + dir) / 2;
        if (esq <= dir) {
                if (valor > vetor[meio]) {
                        esq = meio + 1;
                        return binaria(valor, vetor, esq, dir);
                } else if (valor < vetor[meio]) {
                        dir = meio - 1;
                        return binaria(valor, vetor, esq, dir);
                } else {
                        return meio;
                }
        } else {
                return -1;
        }
}
```

metade do final

metade do começo

não achou

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c$$

o processamento nesta chamada é cte

Vai apenas por uma das metades

$T(n) = T(\frac{n}{2}) + k$ $\quad$ $T(1) = 0$ 1

merece observação

$$\underbrace{T(n) = O(\log(n))}_{0 \leq T(n) \leq c \cdot \log(n) \quad p/ n \geq n_0}$$

---

por simplicidade vamos adotar $n = 2^m$ e log na base 2.

---

parte 1: $T(n)$ é proporcional a $\log(n)$

- testando usando a definição

$T(1) = 0$

$T(2) = k$

$T(4) = T(2) + k = 2k$

$T(8) = T(4) + k = 3k$

conjectura:

$T(n) = k \cdot \log_2(n)$

$$T(n) = k \cdot \log_2 n \quad (hip)$$

é isto q queremos demonstrar

até agora extraímos a recorrência, testamos valores e conjecturamos que a fórmula é a apresentada acima. Agora precisamos demonstrar por indução. Para a base, testamos a fórmula p/ diferentes valores de n e vemos que funciona.

Agora o passo:

tome o próximo termo: $2 \cdot n$

PELA DEFINIÇÃO

$$T(2 \cdot n) = T(n) + k$$

usando a hipótese

$$T(2 \cdot n) = k \cdot \log_2 n + k = k\left(\log_2 n + 1\right) =$$

$$= k\left(\log_2 n + \log_2 2\right) = k \log 2n$$

$\log_2 2 = 1$

provamos que $T(n) = k \log(n)$

isto é $O(\log(n))$?

sim, basta tomar $c = k$ e $n_0 = 1$

note que $T(n) \in \Theta(\log n)$!!!

$T(1) = 0$ ⊥

merece observação

Lembre que $T(n)$ representa tempo de execução ou número de operações logo, admitir que vale zero é uma simplificação. Note que admitir que $T(1) = k$, ou que $T(1) = k_2$, uma constante diferente de k, não muda a classe de complexidade do algoritmo.