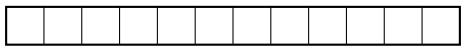
Legal.

Agora nós estamos prontos para acelerar o nosso algoritmo ao máximo.

Quer dizer, a gente já viu que aumentar o número de blocos é uma boa ideia.

Daí que, aumentando esse número ao máximo, a gente divide a lista em n blocos de tamanho 1



Mas isso não é uma boa ideia ...

Quer dizer, basta olhar o que acontece com o tempo de execução.

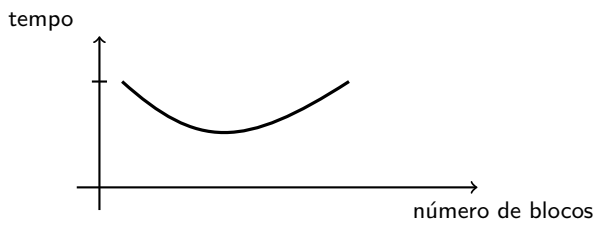
Substituindo o k por n na expressão acima, nós obtemos

$$\left(\frac{n^2}{n}\right) \quad + \quad (n-1) \cdot O(n) \quad = \quad O(n^2)$$

`Hmm ... :/`

Quer dizer, o que está acontecendo é o seguinte

→ *Quando a gente aumenta o número de blocos,*
o tempo de execução primeiro diminui e depois aumenta



Então, nós precisamos descobrir o ponto onde o tempo de execução é o menor possível.

E para isso, nós precisamos entender melhor o que está acontecendo.

Olhando para os dois casos extremos, nós vemos o seguinte

- 2 blocos: $O(n^2/2) \quad + \quad O(n)$
- n blocos: $O(n) \quad + \quad (n-1) \cdot O(n)$

Quer dizer, no primeiro caso o tempo da ordenação é grande, e o tempo da intercalação é pequeno.

E no segundo caso, o tempo da ordenação é pequeno e o tempo da intercalação é grande.

Daí que, o tempo de execução mínimo acontece quando esses dois tempos são iguais.

Usando os termos da expressão acima, nós escrevemos

$$O\left(\frac{n^2}{k}\right) \quad = \quad (k-1) \cdot O(n)$$

o que a gente reescreve de maneira simplificada assim

$$\frac{n^2}{k} \quad = \quad k \cdot n$$

o que pode ser reescrito assim

$$n \quad = \quad k^2$$

o que nos dá

$$k \quad = \quad \sqrt{n}$$

Quer dizer, a situação ideal acontece quando nós dividimos a lista em $\sqrt{n}$ blocos

— (*com tamanho* $\sqrt{n}$ *cada um*)

Nessa situação, o tempo de execução do algoritmo é

$$O\left(\frac{n^2}{\sqrt{n}}\right) \quad + \quad (\sqrt{n}-1) \cdot O(n) \quad = \quad O\left(n \cdot \sqrt{n}\right)$$

E agora, nós temos um algoritmo que é bem mais rápido do que os algoritmos que nós vimos na aula passada.

Não é legal?