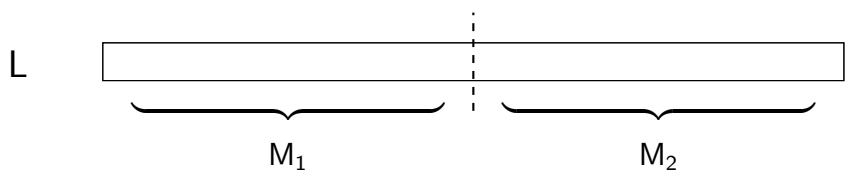
O algoritmo de divisão e conquista

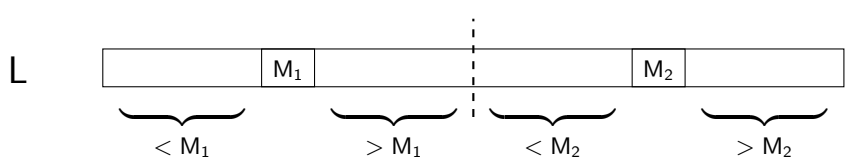
A nossa primeira tentativa é aplicar o método da divisão e conquista.

E para isso, a gente resolve o problema nas duas metades da lista



Mas, ninguém garante que M_1 ou M_2 é a mediana da lista inteira.

Então, a gente começa particionando as duas metades utilizando M_1 e M_2 como pivôs



E daí, a gente raciocina assim

- Suponha que $\mathsf{M}_1 < \mathsf{M}_2$.

  Então,

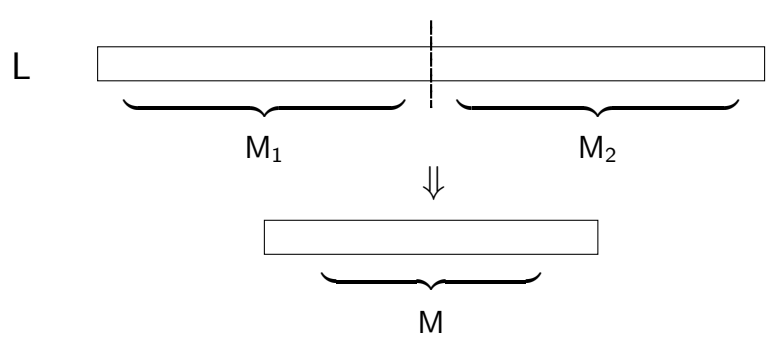
  - todo elemento à esquerda de M_1 é menor do que $n/2$ elementos (ou mais)
  - todo elemento à direita de M_2 é maior do que $n/2 + 2$ elementos (ou mais)

  Logo, nenhum desses elementos pode ser a mediana.

— (*e o caso em que* $\mathsf{M}_2 < \mathsf{M}_1$ *é análogo ...*)

Esse raciocínio nos permite identificar e excluir $n/2$ elementos — (*em tempo* $O(n)$)

E agora, basta encontrar a mediana dos elementos que restaram



Quer dizer, agora nós temos um algoritmo de divisão e conquista

```
                Algoritmo  Mediana-DC ( L, inicio, fim )
                 {
O(1)                  Se ( fim = inicio + 1 )   Retorna ( L[fim] )

                    meio  <--  ( inicio + fim ) / 2

T(n/2) ←              M1  <--  Mediana-DC  (L, inicio, meio)

T(n/2) ←              M2  <--  Mediana-DC  (L, meio+1, fim)

O(n)   ←           j,k  <--  Reorganiza (L, M1, M2, inicio, fim)

T(n/2) ←             M  <--  Mediana-DC  (L, j, k)

O(1)                  Retorna ( M )
                 }
```

E agora, nós podemos analisar o tempo de execução do algoritmo pelo método da equação de recorrência.

Com base nas anotações do código, nós escrevemos a recorrência

$$T(n) \quad = \quad 3 \cdot T\left(\frac{n}{2}\right) \quad + \quad n$$

Para obter a solução, nós desenrolamos a recorrência

$$\begin{aligned}
T(n) \quad &= \quad 3 \cdot T\left(\tfrac{n}{2}\right) \quad + \quad n \\
&= \quad 3 \cdot \left[\, 3 \cdot T\left(\tfrac{n}{2^2}\right) \quad + \quad \tfrac{n}{2} \,\right] \quad + \quad n \\
&= \quad 3^2 \cdot T\left(\tfrac{n}{2^2}\right) \quad + \quad \tfrac{3n}{2} \quad + \quad n \\
&= \quad 3^3 \cdot T\left(\tfrac{n}{2^3}\right) \quad + \quad \tfrac{3^2 n}{2^2} \quad + \quad \tfrac{3n}{2} \quad + \quad n \\
&= \quad \ldots \\
&= \quad 3^{\log_2 n} \cdot T(1) \quad + \quad n \left(\tfrac{3}{2}\right)^{\log_2 n - 1} \quad + \quad \ldots \quad + \quad \tfrac{3n}{2} \quad + \quad n
\end{aligned}$$

Daí, a gente pode fazer a seguinte esperteza

$$3^{\log_2 n} \quad = \quad \left(2^{\log_2 3}\right)^{\log_2 n} \quad = \quad \left(2^{\log_2 n}\right)^{\log_2 3} \quad = \quad n^{\log_2 3}$$

Por outro lado, o restante dos termos podem ser reescritos assim

$$n \cdot \left[\, 1 \quad + \quad \tfrac{3}{2} \quad + \quad \left(\tfrac{3}{2}\right)^2 \quad + \quad \ldots \quad + \quad \left(\tfrac{3}{2}\right)^{\log_2 n - 1} \,\right]$$

Quer dizer, nós temos uma soma de PG dentro dos parênteses.

E a gente lembra que

$$\begin{aligned}
\mathsf{S} \quad &= \quad 1 \quad + \quad \tfrac{3}{2} \quad + \quad \left(\tfrac{3}{2}\right)^2 \quad + \quad \ldots \quad + \quad \left(\tfrac{3}{2}\right)^k \\
&= \quad 1 \quad + \quad \tfrac{3}{2} \cdot \Big[\, 1 \underbrace{\quad + \quad \left(\tfrac{3}{2}\right) \quad + \quad \ldots \quad + \quad \left(\tfrac{3}{2}\right)^{k-1}}_{\mathsf{S} \quad - \quad \left(\frac{3}{2}\right)^k} \,\Big]
\end{aligned}$$

donde

$$\mathsf{S} \quad = \quad 1 \quad + \quad \tfrac{3}{2} \cdot \mathsf{S} \quad - \quad \left(\tfrac{3}{2}\right)^{k+1}$$

donde

$$\mathsf{S} \quad = \quad 2 \cdot \left(\left(\tfrac{3}{2}\right)^{k+1} \quad - \quad 1 \right)$$

Substituindo esse resultado na expressão acima, nós obtemos

$$\begin{aligned}
n \cdot \left(2 \cdot \left(\tfrac{3}{2}\right)^{\log_2 n} \quad - \quad 2 \right) \quad &= \quad 2n \cdot \frac{n^{\log_2 3}}{n} \quad - \quad 2n \\
&= \quad 2 \cdot n^{\log_2 3} \quad - \quad 2n
\end{aligned}$$

Finalmente, juntando uma coisa com a outra, a gente obtém

$$\begin{aligned}
T(n) \quad &= \quad n^{\log_2 3} \quad + \quad 2 \cdot n^{\log_2 3} \quad - \quad 2n \\
&= \quad O(n^{\log_2 3}) \quad = \quad O(n^{1{,}58})
\end{aligned}$$

Quer dizer, isso é melhor do que o algoritmo porcaria.

Mas, não é tão bom quanto a solução via ordenação.

Então, ainda deve ser possível fazer melhor do que isso.