pythonpro

Python para profisionais

www.python.pro.br

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

# Roteiro - QuickSort

Definição

Análise

Visualização

Implementação

Merge versus Quick

Exercício *

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

# Definição

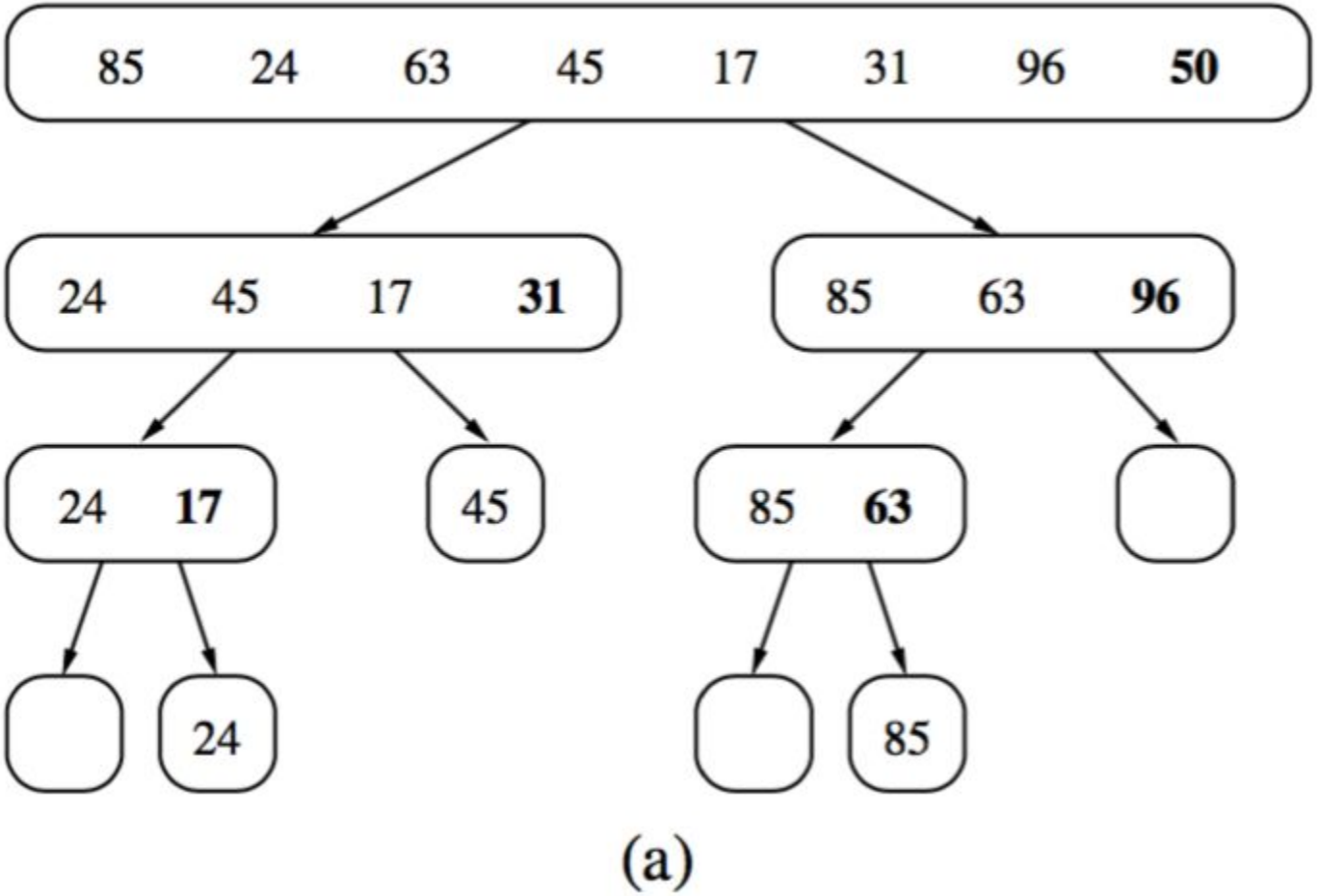
Solução óbvia quando len(sequencia)<=1

Divisão: Escolher pivot. Criar duas listas. Uma com elementos menores que pivot e outra com maiores

Conquista: ordenar sublistas com quick_sort

Combinação: concatenar listas ordenadas *

# Análise

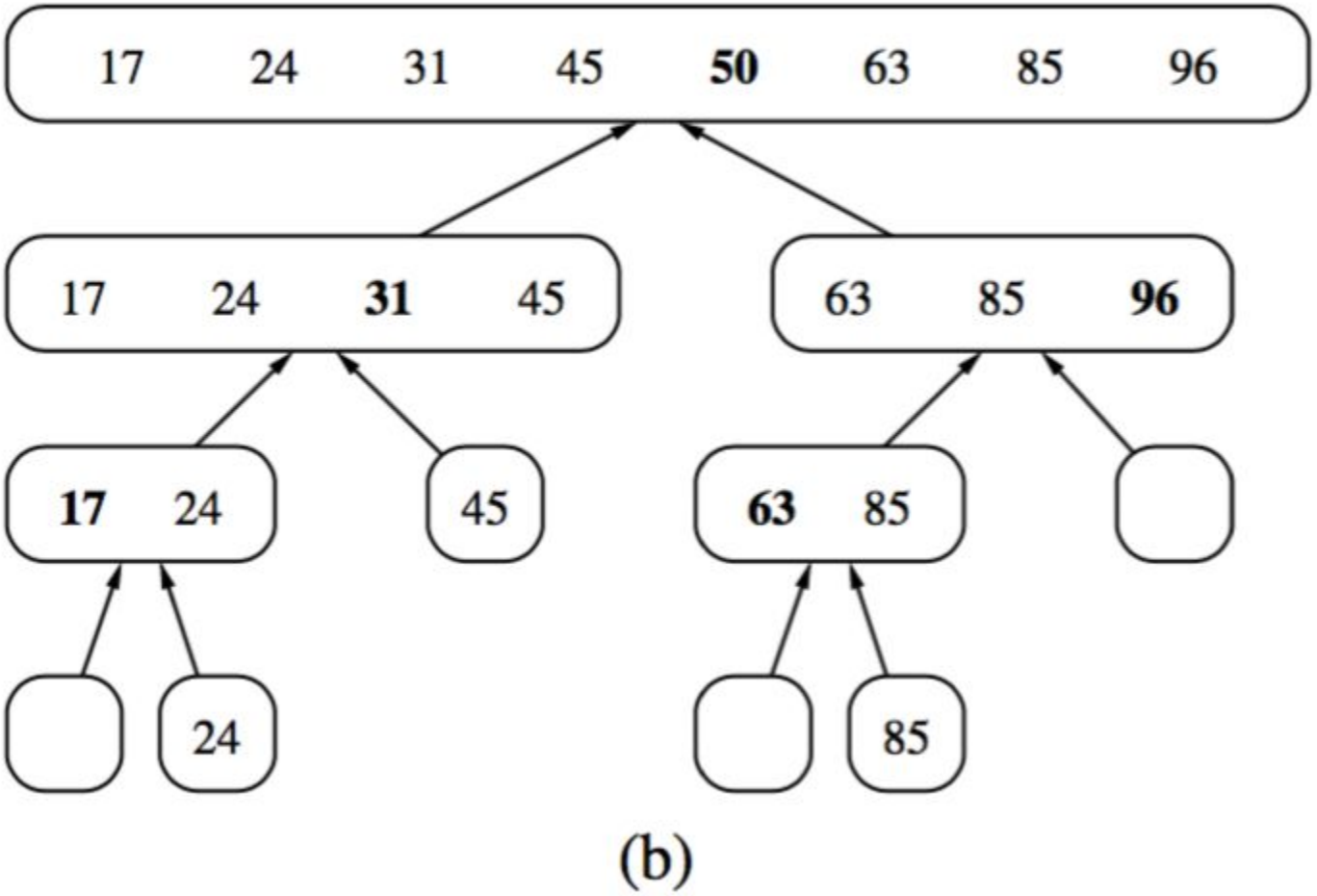



(a)

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

pythonpro

# Análise




(b)

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

pythonpro

# Análise - Caso Médio

Pivot separa itens na metade

Cada Nível Possui n elementos

Concatenação de sublistas será então O(n) para cada nível

O número de níveis é igual ao teto de $\log_2(n)$

Portanto complexidade é O(n*log(n))

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

# Análise - Pior Caso

Lista ordenada e escolha de pivot como último elemento

Separação sempre será em um lista vazia e outra com n-1 elementos

Nesse caso o algoritmo é $O(n^2)$

Escolha de Pivot aleatório ou com mediana evita problema *

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

# Visualiação


**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

pythonpro

# Implementação

Usar o código base de testes de ordenação

Implementar o QuickSort rercusivo: http://bit.ly/1pQWXeP

# Merge versus Quick

Escolha de pivot randomico

Merge exige criação de muita lista adicional com grande footprint de memória

Por conta disso Quick costuma ser mais eficiente *

**Email**: renzo@python.pro.br **Twitter**: @renzoprobr

# Exercício QuckSort in-place

Usar teste base

Implementar QuickSort interno (sem usar criação de sublistas) *

# Obrigado

renzo@python.pro.br
@renzoprobr