Algoritmos e Programação de Computadores

Ordenação: Quick Sort & Merge Sort

Profa. Sandra Avila
Instituto de Computação (IC/Unicamp)

MC102, 12 Junho, 2019

Introdução

Vamos usar a técnica de **recursão** para resolver o problema de **ordenação**.

- Problema:
  - Temos uma lista `v` de inteiros de tamanho `n`.
  - Devemos deixar `v` ordenada em ordem crescente de valores.

Dividir e Conquistar

- Temos que resolver um problema `P` de tamanho `n`.
- **Dividir**: Quebramos `P` em sub-problemas menores.
- Resolvemos os sub-problemas de forma recursiva.
- **Conquistar**: Unimos as soluções dos sub-problemas para obter solução do problema maior `P`.

Quick Sort

Quick Sort

- Vamos supor que devemos ordenar uma lista de uma posição `inicial` até `fim`.
- **Dividir**:
  - Escolha em elemento especial da lista chamado `pivô`.
  - Particione a lista em uma posição `pos` tal que todos elementos de `inicial` até `pos - 1` são menores ou iguais do que o `pivô`, e todos elementos de `pos` até `fim` são maiores ou iguais ao `pivô`.

Quick Sort

- Resolvemos o problema de ordenação de forma recursiva para estas duas sub-listas (uma de `inicial` até `pos-1` e a outra de `pos` até `fim`).
- **Conquistar**: Nada a fazer já que a lista estará ordenada devido à fase de divisão.

Quick Sort: Particionamento

Dado um valor `p` como `pivô`, como fazer o particionamento?

- Podemos “varrer” a lista do início para o fim até encontrarmos um elemento maior que o `pivô`.
- “Varremos” o vetor do fim para o início até encontrarmos um elemento menor ou igual ao `pivô`.
- Trocamos então estes elementos de posições e continuamos com o processo até termos verificado todas as posições do vetor.

Quick Sort: Particionamento

A função retorna a posição de partição. Ela considera o último elemento como o `pivô`.

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
    # o laço para quando inicio == fim => checamos o vetor inteiro
        while (inicio < fim) and (v[inicio] <= pivo):
        # acha posição de elemento maior que pivo
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
        # acha posição de elemento menor ou igual que pivo
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio] # troca elementos de posição
    return inicio
```

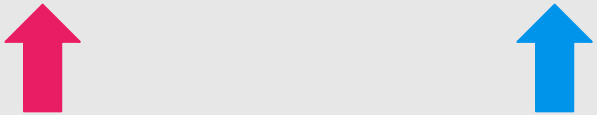
Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,9,3,7,6,2,3,8,5)`



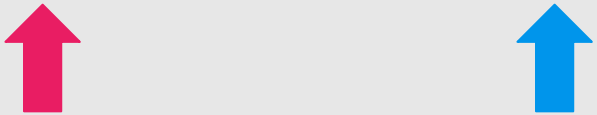
inicio = 0
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,9,3,7,6,2,3,8,5)`



inicio = 0
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,9,3,7,6,2,3,8,5)`

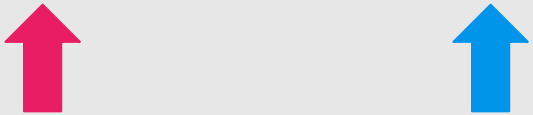
inicio = 1
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: (1,9,3,7,6,2,3,8,5) e pivô=5.

- (1,**9**,3,7,6,2,3,8,**5**)



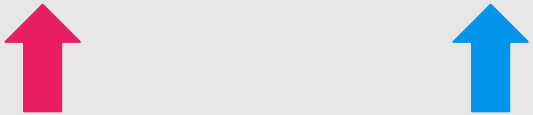
inicio = 1
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,9,3,7,6,2,3,8,5)` → `(1,5,3,7,6,2,3,8,9)`



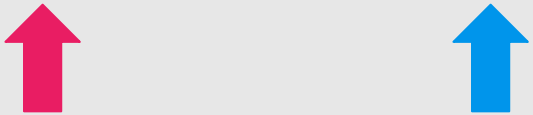
inicio = 1
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



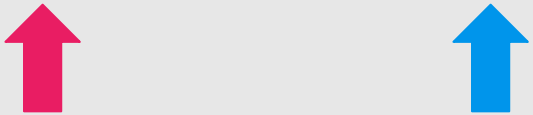
inicio = 1
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



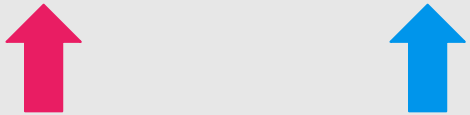
inicio = 1
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



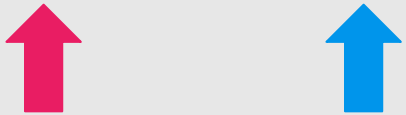
inicio = 2
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



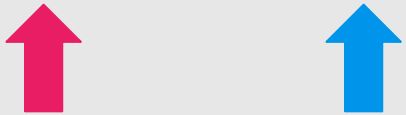
inicio = 3
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



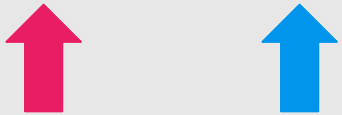
inicio = 3
fim = 8
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`



inicio = 3
fim = 7
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,7,6,2,3,8,9)`

inicio = 3
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: (1,9,3,7,6,2,3,8,5) e pivô=5.

- (1,5,3,7,6,2,3,8,9) → (1,5,3,3,6,2,7,8,9)

inicio = 3
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,6,2,7,8,9)`

inicio = 3
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,6,2,7,8,9)`

inicio = 3
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,6,2,7,8,9)`

inicio = 4
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,6,2,7,8,9)`

inicio = 4
fim = 6
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,6,2,7,8,9)`

inicio = 4
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: (1,9,3,7,6,2,3,8,5) e pivô=5.

- (1,5,3,3,**6**,**2**,7,8,9) → (1,5,3,3,**2**,**6**,7,8,9)

inicio = 4
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,2,6,7,8,9)`

inicio = 4
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,2,6,7,8,9)`

inicio = 4
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,2,6,7,8,9)`

inicio = 5
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,5,3,3,2,6,7,8,9)` → Retorna posição `5 (inicio=5)`.

inicio = 5
fim = 5
pivo = 5

```
def particiona (v, inicio, fim):
    pivo = v[fim]
    while (inicio < fim):
        while (inicio < fim) and (v[inicio] <= pivo):
            inicio = inicio + 1
        while (inicio < fim) and (v[fim] > pivo) :
            fim = fim - 1
        v[inicio], v[fim] = v[fim], v[inicio]
    return inicio
```

Quick Sort: Particionamento

Exemplo: `(1,9,3,7,6,2,3,8,5)` e `pivô=5`.

- `(1,9,3,7,6,2,3,8,5)` → `(1,5,3,7,6,2,3,8,9)`
- `(1,5,3,7,6,2,3,8,9)` → `(1,5,3,3,6,2,7,8,9)`
- `(1,5,3,3,6,2,7,8,9)` → `(1,5,3,3,2,6,7,8,9)`
- `(1,5,3,3,2,6,7,8,9)` → Retorna posição `5`.

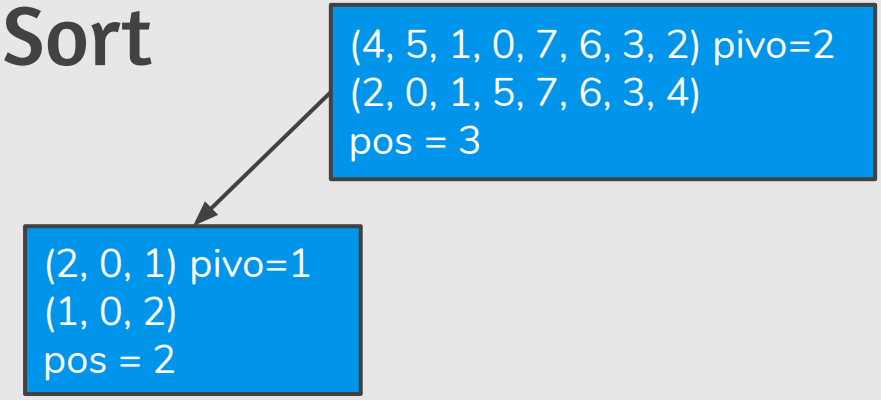
Quick Sort

```
def quickSort(v, inicio, fim):
    if (inicio < fim):
    # tem pelo menos 2 elementos a serem ordenados
        pos = particiona(v, inicio, fim)
        quickSort(v, inicio, pos-1)
        quickSort(v, pos, fim)
```

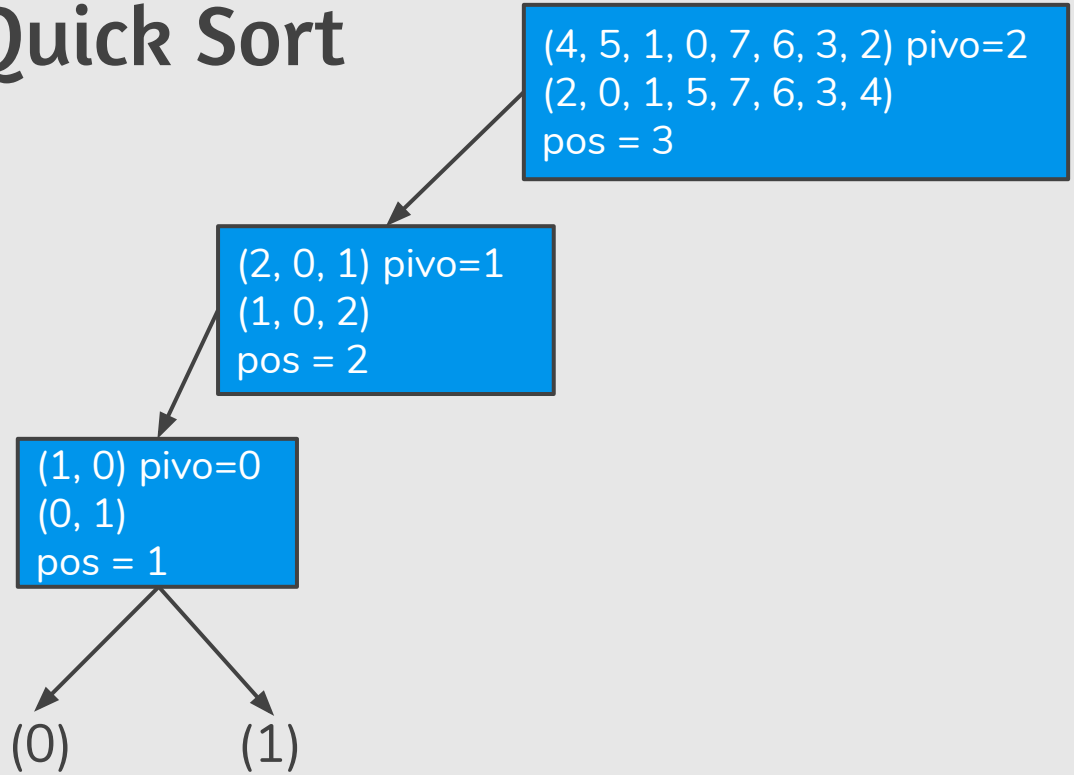
Quick Sort

(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3

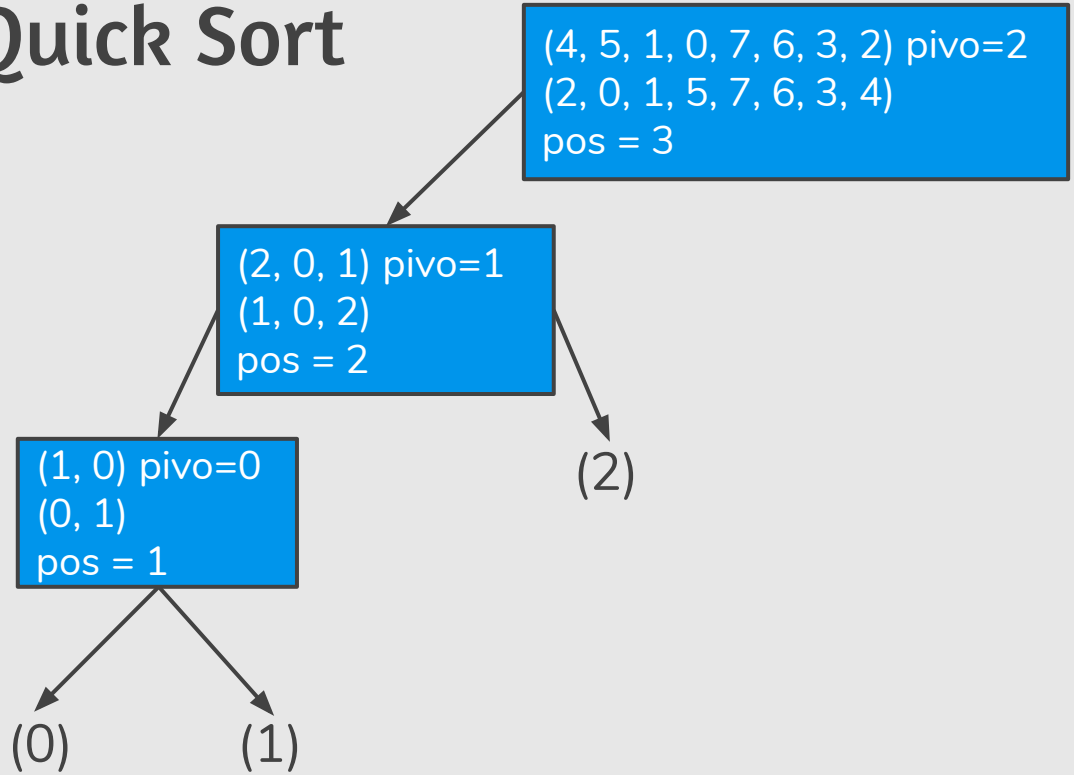
Quick Sort



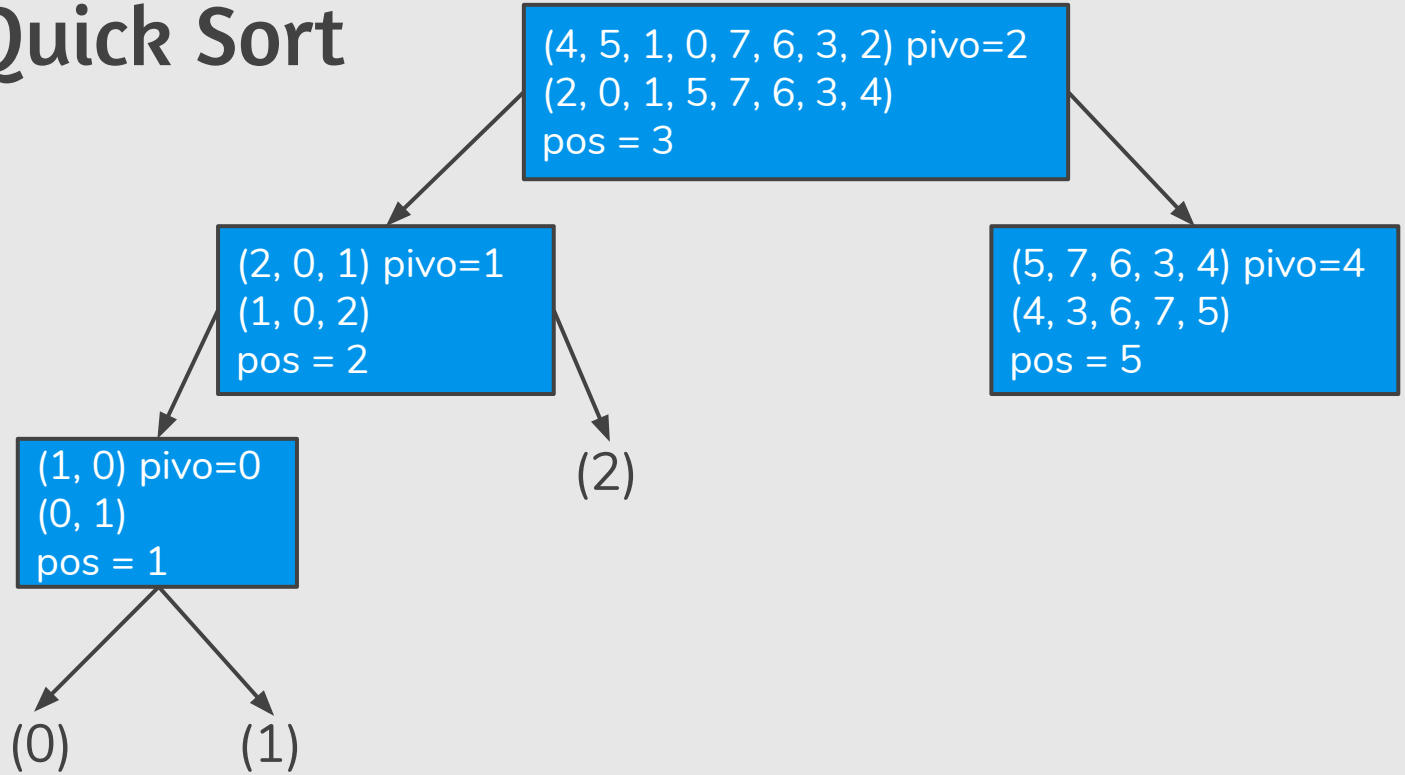
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(1, 0) pivo=0
(0, 1)
pos = 1
(0)
(1)

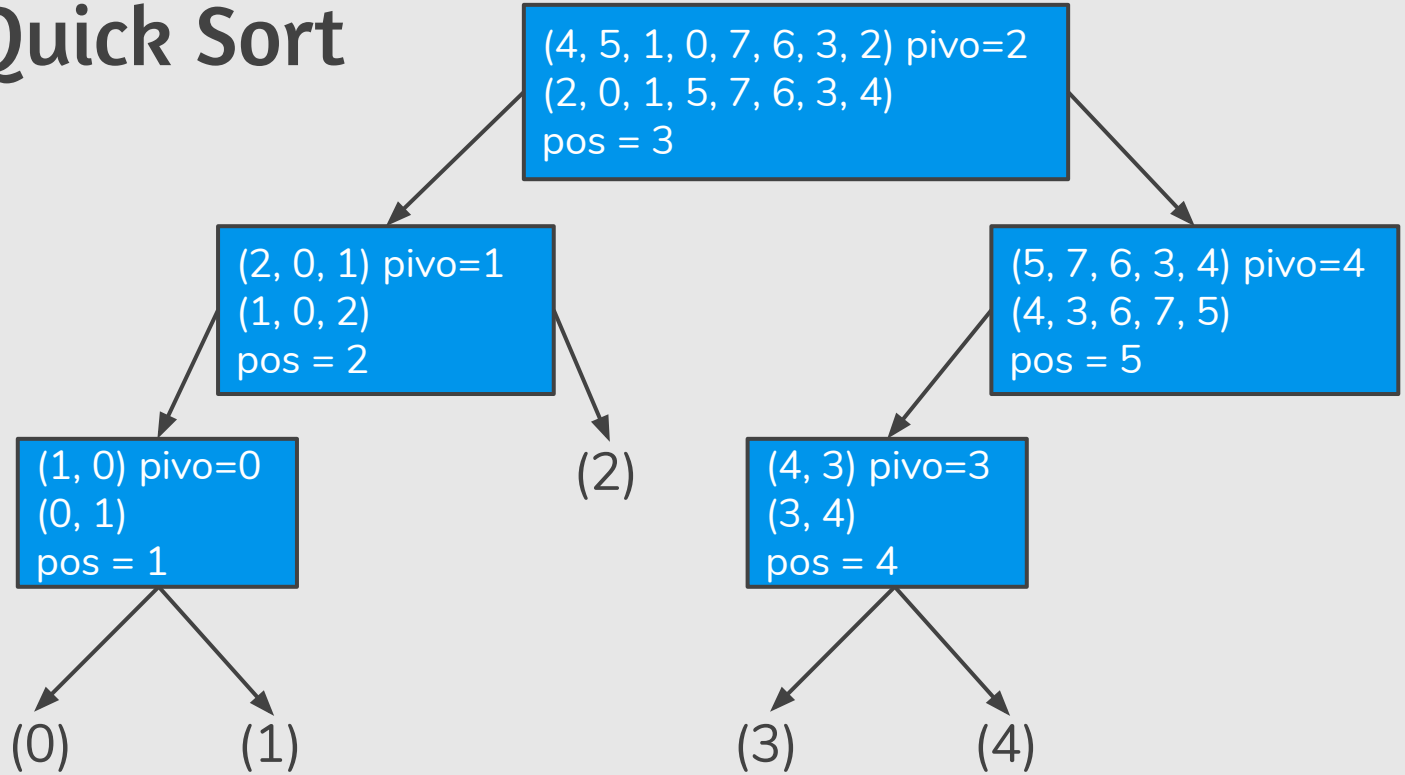
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(2)
(1, 0) pivo=0
(0, 1)
pos = 1
(0)
(1)

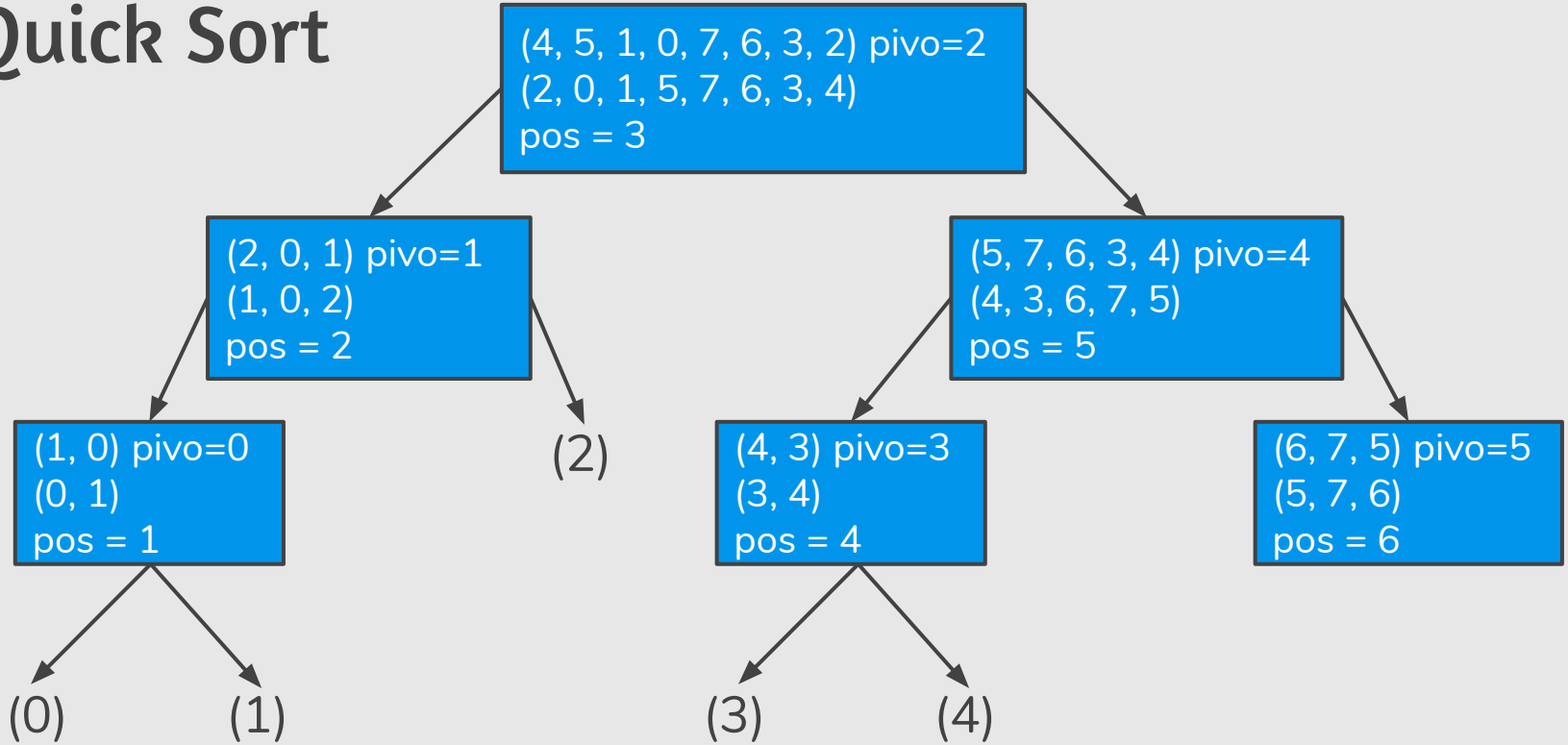
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(0)
(1)

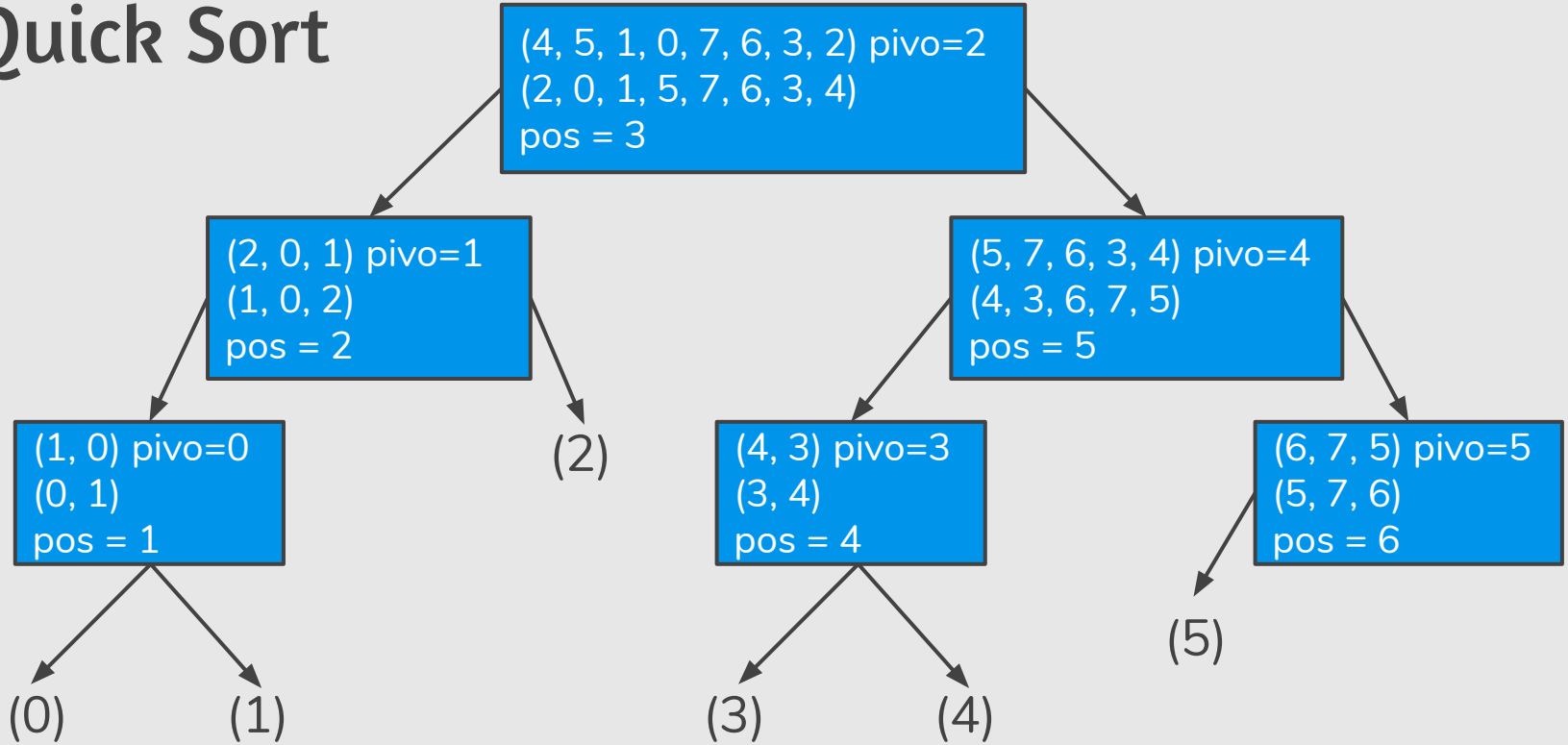
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(4, 3) pivo=3
(3, 4)
pos = 4
(0)
(1)
(3)
(4)

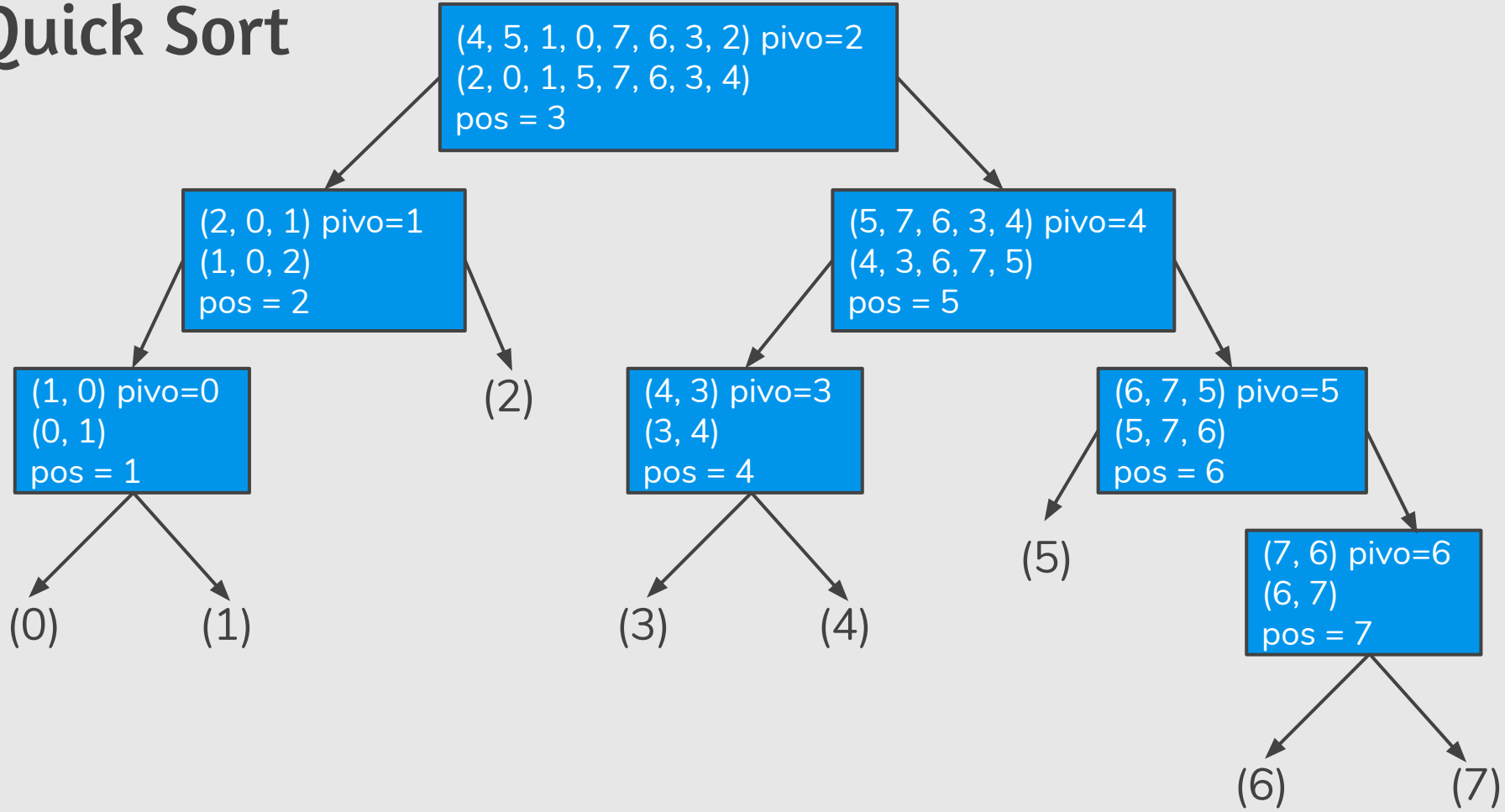
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(4, 3) pivo=3
(3, 4)
pos = 4
(6, 7, 5) pivo=5
(5, 7, 6)
pos = 6
(0)
(1)
(3)
(4)

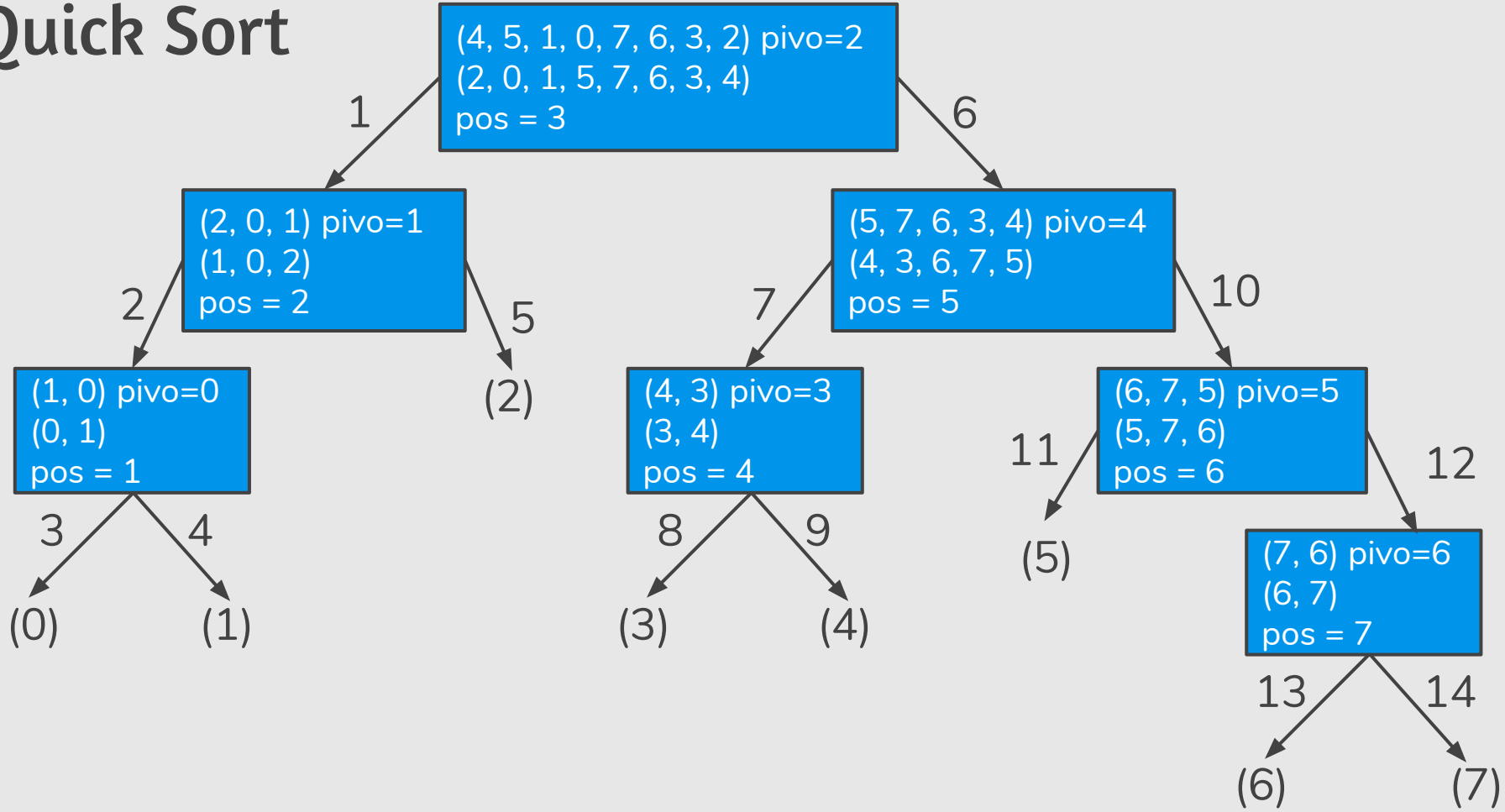
Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(4, 3) pivo=3
(3, 4)
pos = 4
(6, 7, 5) pivo=5
(5, 7, 6)
pos = 6
(0)
(1)
(3)
(4)
(5)

Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(4, 3) pivo=3
(3, 4)
pos = 4
(6, 7, 5) pivo=5
(5, 7, 6)
pos = 6
(0)
(1)
(3)
(4)
(5)
(7, 6) pivo=6
(6, 7)
pos = 7
(6)
(7)

Quick Sort


(4, 5, 1, 0, 7, 6, 3, 2) pivo=2
(2, 0, 1, 5, 7, 6, 3, 4)
pos = 3
1
6
(2, 0, 1) pivo=1
(1, 0, 2)
pos = 2
(5, 7, 6, 3, 4) pivo=4
(4, 3, 6, 7, 5)
pos = 5
2
5
7
10
(1, 0) pivo=0
(0, 1)
pos = 1
(2)
(4, 3) pivo=3
(3, 4)
pos = 4
(6, 7, 5) pivo=5
(5, 7, 6)
pos = 6
11
12
3
4
8
9
(0)
(1)
(3)
(4)
(5)
(7, 6) pivo=6
(6, 7)
pos = 7
13
14
(6)
(7)

Quick Sort

- Se o QuickSort particionar o vetor de tal forma que cada partição tenha mais ou menos o mesmo tamanho ele é muito eficiente.
- Porém se a partição for muito desigual (`n - 1` de um lado e `1` de outro) ele é ineficiente.
- Quando um vetor já está ordenado ou quase ordenado, ocorre este caso ruim. Por que?

Quick Sort: Tratando o pior caso

- Podemos implementar o QuickSort de tal forma a diminuirmos a chance de ocorrência do pior caso.
- Ao invés de escolhermos o pivô como um elemento de uma posição fixa, podemos escolher como pivô o elemento de uma posição aleatória.
- Podemos usar a função `random.randint(a,b)` da biblioteca `random` que retorna um número de forma aleatória entre `a` e `b`.

Random Quick Sort

- A única diferença é que escolhemos um elemento aleatório.
- Tal elemento é trocado com o que está no fim (será o pivô).

```
import random
def randomQuickSort(v, inicio, fim):
    if (inicio < fim):
        j = random.randint(inicio, fim)
        v[j], v[fim] = v[fim], v[j]
        pos = particiona(v, inicio, fim)
        randomQuickSort(v, inicio, pos-1)
        randomQuickSort(v, pos, fim)
```

Exercícios

1. Aplique o algoritmo de particionamento sobre o vetor `(13,19,9,5,12,21,7,4,11,2,6,6)` com pivô igual a `6`.
2. Qual o valor retornado pelo algoritmo de particionamento se todos os elementos do vetor tiverem valores iguais?
3. Faça uma execução passo-a-passo do `quickSort` com o vetor `(4,3,6,7,9,10,5,8)`.
4. Modifique o algoritmo `quickSort` para ordenar vetores em ordem decrescente.