Estrutura de Dados

Aula 11 – Quick sort

Agenda

- Introdução
- Quicksort
  - Partição
  - Funcionamento
  - Comportamento
  - Implementação recursiva
  - Implementação iterativa
- Comparação
- Exercícios

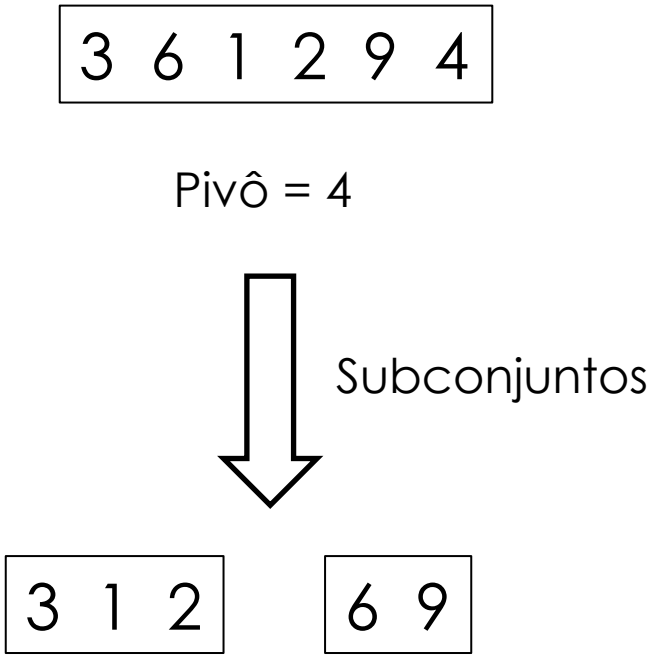
Introdução

- É o algoritmo de ordenação interna mais rápido que se conhece para uma ampla variedade de situações
- Provavelmente o mais utilizado
- Assim como o Merge Sort, aproveita do conceito de divisão e conquista
  - Difere da maneira em que os subconjuntos são gerados

Partição

- A parte principal do método é a partição do conjunto de dados
- Um subconjunto é gerado levando em consideração um **pivô**
- Um conjunto é separdo em dois
  - Um subconjunto é montado com os dados menores que o pivô
  - O outro subconjunto é montado com os dados maiores que o pivô
- Ao combinar os dois subconjuntos, tem a garantia que o pivô estará no meio dos dois subconjuntos

Partição



Partição

- Pior caso
  - Acontece quando a chave escolhida como pivô sempre é a extremidade de um conjunto
- Melhor caso
  - Acontece quando a chave escolhida como pivô resulta na divisão de subconjuntos de mesmo tamanho
- Opções para escolha do pivô
  - Posição intermediária do conjunto
  - Sorteio
  - Mediana entre 3 dados quaisquer do conjunto
  - ...

Partição

- Algoritmo do particionamento
  - Escolha de um pivô (x)
  - Percorra o vetor a partir da esquerda até que v[i] >= x
  - Percorra o vetor a partir da direita até que v[j] <= x
  - Troque v[i] com v[j]
  - Continue este processo até os apontadores i e j se cruzem

Partição

- Depois da partição, será garantido que
  - Todos os itens do 1º subconjunto são menores que o pivô
  - Todos os itens do 2º subconjunto são maiores que o pivô
  - O pivô encontra-se na posição correta de ordenação

Funcionamento

6 5 3 1 8 7 2 4

https://upload.wikimedia.org/wikipedia/commons/9/9c/Quicksort-example.gif

Comportamento

- https://www.toptal.com/developers/sorting-algorithms/quick-sort
- Custo
  - O(n log n), mas pode alcançar $O(n^2)$
- Não é estável

Implementação recursiva

```
void quickSort (TItem *v, int n) {
   quickSortOrdena (v, 0, n-1);
}

void quickSortOrdena (TItem *v, int esq, int dir) {
   int i, j;

   quickSortParticao (v, esq, dir, &i, &j);

   if (esq < j)
      quickSortOrdena (v, esq, j);
   if (i < dir)
      quickSortOrdena (v, i, dir);
}
```

Implementação recursiva

```
void quickSortParticao (TItem *v, int esq, int dir, int *i, int *j)
{
	TItem pivo, aux;
	*i = esq;
	*j = dir;
	pivo = v [(*i + *j) /2];

	do {
		while (pivo.chave > v[*i].chave)
			(*i)++;

		while (pivo.chave < v[*j].chave)
			(*j)--;

		if (*i <= *j) {
			aux = v[*i];
			v[*i] = v[*j];
			v[*j] = aux ;
			(*i)++;
			(*j)--;
		}
	} while (*i <= *j);
}
```

Implementação iterativa

```
void quickSortIterativo (TItem *v, int n) {
	TPilha pilhaDir, pilhaEsq;
	int esq, dir, i, j;

	iniciarPilha (&pilhaDir);
	iniciarPilha (&pilhaEsq);
	esq = 0;
	dir = n - 1;

	push (&pilhaDir, dir);
	push (&pilhaEsq, esq);

	do {
		if (dir > esq) {
			quickSortParticao (v, esq, dir, &i, &j);
			push (&pilhaDir, j);
			push (&pilhaEsq, esq);
			esq = i;
		} else {
			pop (&pilhaDir, &dir);
			pop (&pilhaEsq, &esq);
		}
	} while (!isVazia (&pilhaDir));
}
```

Comparação

Algoritmo	Comparações			Movimentações		
	Melhor	**Médio**	**Pior**	**Melhor**	**Médio**	**Pior**
Bubble	$O(n^2)$			$O(n^2)$		
Selection	$O(n^2)$			$O(n)$		
Insertion	$O(n)$	$O(n^2)$		$O(n)$	$O(n^2)$	
Merge	O (n log n)			-		
Quick	O (n log n)		$O(n^2)$	-		

Exercícios

- Implemente o método de ordenação Quicksort
- Compare o tempo de ordenação entre o Quicksort e o Mergesort
  - Use um conjunto de dados grande para ficar mais fácil a comparação
  - Teste com diferentes conjuntos de dados
    - dados aleatórios
    - dados já quase ordenados
    - dados ordenados de maneira inversa

Estrutura de Dados

Material elaborado por:
Thiago Meirelles Ventura

Baseado em:

- Ascencio, A. F. G; Araújo, G. S. Estruturas de Dados. Pearson, 2011.
- Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. Algoritmos: teoria e prática. Elsevier, 2002.
- Aulas do Prof. Reinaldo Silva Fortes (http://www.decom.ufop.br/reinaldo/)