Quicksort

Além do Sedgewick (**sempre** leiam o Sedgewick), veja

- `http://www.ime.usp.br/~pf/algoritmos/aulas/quick.html`

Quicksort

- Baseado na noção de *divisão-e-conquista* (divide-and-conquer)
- *Divisão-e-conquista:* (*i*) decomponha o problema em 'subproblemas' menores; (*ii*) resolva estes subproblemas (tipicamente de forma recursiva); (*iii*) use as soluções obtidas para compor uma solução do problema original

Divisão-e-conquista e ordenação

- Para ordenar uma coleção de objetos usando divisão-e-conquista: (*i*) particione a coleção em duas partes menores; (*ii*) ordene as partes recursivamente; (*iii*) componha a solução a partir das partes ordenadas.
- Duas implementações desta idéia:
  - Quicksort
  - Mergesort
  - Diferença fundamental: como executamos a partição

Quicksort

- O algoritmo de partição do quicksort rearranja o vetor dado `a[l..r]` e determina um índice `i` de forma que
  - o objeto em `a[i]` está em sua posição final
  - todos os elementos em `a[l..i-1]` são menores ou iguais a `a[i]`
  - todos os elementos em `a[i+1..r]` são maiores ou iguais a `a[i]`
- Suponha que `partition(a, l, r)` executa tal rearranjo e devolve o índice `i`

Quicksort

```
void quicksort(Item a[], int l, int r)
  { int i;
    if (r <= l) return;
    i = partition(a, l, r);
    quicksort(a, l, i-1);
    quicksort(a, i+1, r);
  }
```

Quicksort

- Etapa de (*iii*) da divisão-e-conquista é trivial
- Crucial: etapa (*i*), correspondente a `partition()`

Quicksort: algoritmo de partição

```
int partition(Item a[], int l, int r)
  { int i = l-1, j = r; Item v = a[r];
    for (;;)
      { while (less(a[++i], v)) ;
        while (less(v, a[--j])) if (j == l) break;
        if (i >= j) break;
        exch(a[i], a[j]);
      }
    exch(a[i], a[r]);
    return i;
  }
```

Quicksort

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog7.1b 16 1
[l r i =  0 15 13] 84 39 78 79 51 19 33 76 27 55 47 62 36 91 95 91
[l r i =  0 12  3] 27 33 19 36 51 78 39 76 84 55 47 62 79 91 95 91
[l r i =  0  2  0] 19 33 27 36 51 78 39 76 84 55 47 62 79 91 95 91
[l r i =  1  2  1] 19 27 33 36 51 78 39 76 84 55 47 62 79 91 95 91
[l r i =  4 12 11] 19 27 33 36 51 78 39 76 62 55 47 79 84 91 95 91
[l r i =  4 10  5] 19 27 33 36 39 47 51 76 62 55 78 79 84 91 95 91
[l r i =  6 10 10] 19 27 33 36 39 47 51 76 62 55 78 79 84 91 95 91
[l r i =  6  9  7] 19 27 33 36 39 47 51 55 62 76 78 79 84 91 95 91
[l r i =  8  9  9] 19 27 33 36 39 47 51 55 62 76 78 79 84 91 95 91
[l r i = 14 15 14] 19 27 33 36 39 47 51 55 62 76 78 79 84 91 91 95
19 27 33 36 39 47 51 55 62 76 78 79 84 91 91 95
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
```

Quicksort

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog7.1 10 1
197 277 335 394 553 768 783 798 840 911
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog6.1 10 1
197 277 335 394 553 768 783 798 840 911
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog7.1 10 1
197 277 335 394 553 768 783 798 840 911

real    0m0.006s
user    0m0.001s
sys     0m0.002s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog7.1 1000000 1 > /dev/null

real    0m0.826s
user    0m0.788s
sys     0m0.005s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog7.1 10000000 1 > /dev/null

real    0m9.292s
user    0m8.592s
sys     0m0.123s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
```

Insersion sort

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog6.1 4000 1 > /dev/null

real    0m0.163s
user    0m0.161s
sys     0m0.003s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog6.1 8000 1 > /dev/null

real    0m0.661s
user    0m0.641s
sys     0m0.002s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ time prog6.1 16000 1 > /dev/null

real    0m2.688s
user    0m2.560s
sys     0m0.003s
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
```

Desempenho do quicksort

Propriedade 1. *Quicksort faz* $\sim N^2/2$ *comparações (entre itens) no pior caso.*

- ▷ Seqüência ruim: já ordenada!
- ▷ Melhor caso? Caso médio?

Propriedade 2. *Quicksort faz* $\sim 2N \log N$ *comparações no caso em que a entrada é uma permutação escohida uniformemente ao acaso dentre as* $N!$ *possíveis permutações de* N *objetos distintos.*