Recursão

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

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

Torres de Hanói

```
/* prog5.7.c */
#include <stdio.h>
#include <stdlib.h>
void hanoi(int N, int d)
  { if (N == 0) return;
    hanoi(N-1, -d);
    printf("%d ", d*N); /* shift(N, d); */
    hanoi(N-1, -d);
  }
```

Torres de Hanói

```
/* prog5.7.c */
[...]
int main(int argc, char *argv[])
{ int N=atoi(argv[1]);
  hanoi(N,1);
  putchar('\n');
  return 0;
}
```

Torres de Hanói

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 1
1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 2
-1 2 -1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 3
1 -2 1 3 1 -2 1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 4
-1 2 -1 -3 -1 2 -1 4 -1 2 -1 -3 -1 2 -1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog5.7 5
1 -2 1 3 1 -2 1 -4 1 -2 1 3 1 -2 1 5 1 -2 1 3 1 -2 1 -4 1 -2 1 3 1 -2 1
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$
```

Exercício

Sugestão: estude e faça os exercícios em

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

Algoritmos de ordenação elementares

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

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

Exemplo completo, simplificado

```
/* prog6.1_simpl.c */
#include <stdio.h>
#include <stdlib.h>

#define exch(A, B) { int t = A; A = B; B = t; }
#define compexch(A, B) if ((B) < (A)) exch(A, B)
```

Exemplo completo, simplificado

```
/* prog6.1_simpl.c */
[...]
/* ordenacao por insercao */
void sort(int a[], int l, int r)
  { int i, j;
    for (i = l+1; i <= r; i++)
      for (j = i; j > l; j--)
        compexch(a[j-1], a[j]);
  }
```

Exemplo completo, simplificado

```
/* prog6.1_simpl.c */
[...]
int main(int argc, char *argv[])
  { int i, N = atoi(argv[1]), sw = atoi(argv[2]);
    int *a = malloc(N*sizeof(int));
    if (sw)
      for (i = 0; i < N; i++) a[i] = 1000*(1.0*rand()/RAND_MAX);
    else
      { N=0; while (scanf("%d", &a[N]) == 1) N++; }
    sort(a, 0, N-1);
    for (i = 0; i < N; i++) printf("%3d ", a[i]);
    printf("\n");
    return 0;
  }
```

Exemplo de execução

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ make prog6.1_simpl
make prog6.1_simpl.o
make[1]: Entering directory `/home/yoshi/IME/www/2006ii/mac122a/exx'
gcc -g -I. -Wall -pedantic -ansi -c prog6.1_simpl.c
make[1]: Leaving directory `/home/yoshi/IME/www/2006ii/mac122a/exx'
gcc -g -I. -Wall -pedantic -ansi -o prog6.1_simpl prog6.1_simpl.o -lm
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog6.1_simpl 10 1
197 277 335 394 553 768 783 798 840 911
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog6.1_simpl 15 1
197 277 335 364 394 477 513 553 628 768 783 798 840 911 952
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$ prog6.1_simpl 0 0
33 11 44 11 55 99 22 55 44
 11  11  22  33  44  44  55  55  99
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
```

Exemplo completo

```
/* prog6.1.c */
#include <stdio.h>
#include <stdlib.h>

typedef int Item;

#define key(A) (A)
#define less(A, B) (key(A) < key(B))
#define exch(A, B) { Item t = A; A = B; B = t; }
#define compexch(A, B) if (less(B, A)) exch(A, B)
```

Exemplo completo

```
/* prog6.1.c */
[...]
/* ordenacao por insercao */
void sort(Item a[], int l, int r)
  { int i, j;
    for (i = l+1; i <= r; i++)
      for (j = i; j > l; j--)
        compexch(a[j-1], a[j]);
  }
```

Exemplo completo

```
/* prog6.1.c */
[...]
int main(int argc, char *argv[])
  { int i, N = atoi(argv[1]), sw = atoi(argv[2]);
    int *a = malloc(N*sizeof(int));
    if (sw)
      for (i = 0; i < N; i++)
        a[i] = 1000*(1.0*rand()/RAND_MAX);
    else
      while (scanf("%d", &a[N]) == 1) N++;
    sort(a, 0, N-1);
    for (i = 0; i < N; i++) printf("%3d ", a[i]);
    printf("\n");
    return 0;
  }
```

Algoritmos elementares de ordenação

- Ordenação por seleção
- Ordenação por inserção
- O método da bolha (“bubble sort”)

Ordenação por seleção

```
void selection(Item a[], int l, int r)
  { int i, j;
    for (i = l; i < r; i++)
      { int min = i;
        for (j = i+1; j <= r; j++)
            if (less(a[j], a[min])) min = j;
        exch(a[i], a[min]);
      }
  }
```

Ordenação por seleção

```
A S O R T I N G E X A M P L E
A S O R T I N G E X A M P L E
A A O R T I N G E X S M P L E
A A E R T I N G O X S M P L E
A A E E T I N G O X S M P L R
A A E E G I N T O X S M P L R
A A E E G I N T O X S M P L R
A A E E G I L T O X S M P N R
A A E E G I L M O X S T P N R
A A E E G I L M N X S T P O R
A A E E G I L M N O S T P X R
A A E E G I L M N O P T S X R
A A E E G I L M N O P R S X T
A A E E G I L M N O P R S X T
A A E E G I L M N O P R S T X
```

Ordenação por inseção

```
void insertion(Item a[], int l, int r)
  { int i;
    for (i = l+1; i <= r; i++) compexch(a[l], a[i]);
    for (i = l+2; i <= r; i++)
      { int j = i; Item v = a[i];
        while (less(v, a[j-1]))
          { a[j] = a[j-1]; j--; }
        a[j] = v;
      }
  }
```

Ordenação por inseção

```
A S O R T I N G E X A M P L E
A O S R T I N G E X A M P L E
A O R S T I N G E X A M P L E
A O R S T I N G E X A M P L E
A I O R S T N G E X A M P L E
A I N O R S T G E X A M P L E
A G I N O R S T E X A M P L E
A E G I N O R S T X A M P L E
A E G I N O R S T X A M P L E
A A E G I N O R S T X M P L E
A A E G I M N O R S T X P L E
A A E G I M N O P R S T X L E
A A E G I L M N O P R S T X E
A A E E G I L M N O P R S T X
```

Método da bolha

```
void bubble(Item a[], int l, int r)
  { int i, j;
    for (i = l; i < r; i++)
      for (j = r; j > i; j--)
        compexch(a[j-1], a[j]);
  }
```

Método da bolha

```
A S O R T I N G E X A M P L E
A A S O R T I N G E X E M P L
A A E S O R T I N G E X L M P
A A E E S O R T I N G L X M P
A A E E G S O R T I N L M X P
A A E E G I S O R T L N M P X
A A E E G I L S O R T M N P X
A A E E G I L M S O R T N P X
A A E E G I L M N S O R T P X
A A E E G I L M N O S P R T X
A A E E G I L M N O P S R T X
A A E E G I L M N O P R S T X
A A E E G I L M N O P R S T X
A A E E G I L M N O P R S T X
A A E E G I L M N O P R S T X
```

Características dos algoritmos elementares de ordenação

Propriedade 1. *Ordenação por seleção faz* $\binom{N}{2} \sim N^2/2$ *comparações e* $N - 1$ *trocas.*

Propriedade 2. *Ordenação por inserção faz basicamente* $N^2/4$ *comparações e* $N^2/4$ *movimentos de dados na média e o dobro destas quantidades no pior caso.*

Propriedade 3. *O método da bolha faz basicamente* $N^2/2$ *comparações e* $N^2/2$ *trocas de dados na média e no pior caso.*

Exercícios

Definição 4. *Uma inversão é um par de objetos que estão fora de ordem em uma seqüência.*

Por exemplo, quantas inversões há em `A B R A C A`? Em quantas inversões participa `R`?

Exercício 5. *Suponha que todos os elementos de uma seqüência participam em uma quantidade limitada de inversões (por exemplo, no máximo* 100*). Qual dos três métodos elementares de ordenação você usaria sobre tal seqüência?*

Exercícios

Exercício 6. *Suponha que todos os elementos de uma seqüência, exceto por uma quantidade limitada (por exemplo,* 100*), participam em uma quantidade limitada de inversões (por exemplo, no máximo* 200*). Qual dos três métodos elementares de ordenação você usaria sobre tal seqüência?*

Exercício 7. *Suponha que a movimentação de dados seja muito mais custosa que a comparação de chaves. Qual dos três métodos elementares de ordenação você usaria sobre tal seqüência?*