Cursos de Verão 2010 - IME/USP
Tópicos de Programação
Ordenação por inserção
Prof. Alexandre da Silva Freire

1 Ordenação por inserção

Eis uma outra solução para o problema da ordenação:

```
    void insercao(int v[], int n) {
       int i, j, x;
(2)    for(j = 1; j < n; j++) {
(3)       x = v[j];
(4)       for(i = j-1; i >= 0 && v[i] > x; i--) {
             v[i+1] = v[i];
          }
(6)       v[i+1] = x;
       }
    }
```

Vejamos os invariantes deste algoritmo:

- invariantes que valem imediatamente antes da execução da linha (4):
  - (i0) os intervalos $v[0\,..\,i]$ e $v[i+2\,..\,j]$ estão ordenados e, além disso, contêm os elementos de $v[0\,..\,j-1]$;
  - (i1) $x < v[i+2\,..\,j]$;
- invariantes que valem imediatamente antes da execução da linha (2):
  - (i2) *o intervalo* $v[0\,..\,j-1]$ *é uma permutação do intervalo* $v[0\,..\,j-1]$ *original*;
  - (i3) *o intervalo* $v[0\,..\,j-1]$ *está em ordem crescente.*

Vejamos como funciona o laço que começa na linha (4). Este laço é responsável por deslocar os elementos de $v[0\,..\,j-1]$ que são maiores que $v[j]$, a fim de "abrir um espaço" no vetor para inserir $v[j]$ de forma que $v[0\,..\,j]$ fique ordenado. Ou seja, o número de passos executados neste laço depende não só do tamanho da entrada, mas também da ordem dos elementos contidos nela. Quando $v[j] \geq v[0\,..\,j-1]$, o laço da linha (4) executa $\Theta(1)$ operações. Quando $v[j] < v[0\,..\,j-1]$, o laço da linha (4) executa aproximadamente j operações. Portanto, o consumo de tempo no pior caso (entrada em ordem decrescente) é $\sum_{j=1}^{n-1} j = 1+2+\cdots+(n-1) = \frac{n(n-1)}{2} = \frac{n^2-n}{2} = \Theta(n^2)$. Já no melhor caso, quando a entrada já está ordenada (em ordem crescente), o consumo de tempo é $\sum_{j=1}^{n-1} \Theta(1) = \Theta(n)$.