UNIVERSIDADE FEDERAL DO ABC
CENTRO DE MATEMÁTICA, COMPUTAÇÃO E COGNIÇÃO
Prof. Monael Pinheiro Ribeiro
Prof. Paulo Henrique Pisani

INSERTION SORT
ANÁLISE DE EFICIÊNCIA

`InsertionEficiencia.[ c | cpp | java | cs ]`

O algoritmo de ordenação Insertion Sort funciona bem para listas pequenas ou com boa parte dela já ordenada. Entretanto, esse algoritmo pode consumir um longo tempo para ordenar uma lista de maior comprimento. Para verificar isso, este exercício trata de analisar o seu tempo de execução e sua eficiência nos mais diversos casos.

O tempo de execução de um algoritmo para uma entrada específica depende do número de operações executadas. Quanto maior o número de operações, maior será o tempo de execução de um algoritmo e menor será sua eficiência. Normalmente deseja-se saber quantas operações um algoritmo executará com relação ao tamanho das entradas. Esse tamanho será chamado de T(**N**).

Quanto a análise do algoritmo de ordenação Insertion Sort, para cada elemento na lista de **N** números, o Insertion Sort o move para a direita até que esse elemento possa ser inserido na sublista ordenada. No melhor caso, ou seja, quando a lista já está ordenada, nenhum elemento será movido e, consequentemente, o algoritmo irá executar uma vez sobre a lista e retorná-la ordenada. O tempo de execução seria diretamente proporcional ao tamanho da entrada. Portanto, podemos dizer que levará tempo T(**N**) proporcional à **N** para executar.

Contudo, normalmente se foca no pior caso do tempo de execução. O pior caso para a ordenação Insertion Sort ocorre quando a lista está ordenada de forma inversa à desejada. Para inserir cada elemento na sublista ordenada, o algoritmo precisará mover cada elemento para início da lista.

Neste caso, ordenar uma lista de **N** número consumira, portanto:

T(N) = **1 + 2 + ... + (N-1)**

O que é:

T(N) = N(N-1)/2

Pelas regras da notação assintótica este termo é generalizado para N^2 (termo dominante) e dizemos então, que o algoritmo de ordenação Insertion Sort é um algoritmo de tempo O(N^2), ou ainda, um algoritmo quadrático.

Isso significa que a medida que o tamanho **N** da entrada aumenta, o algoritmo de ordenação Insertion Sort aumentará pelo quadrado de **N** o seu esforço para ordenar a lista. Dessa forma, o Insertion Sort pode funcionar bem para entradas pequenas, mas torna-se muito lento para entradas suficientemente grandes.

Neste exercício você terá que determinar a quantidade de movimentos dos elementos da lista que o algoritmo de ordenação Insertion Sort consumirá para ordenar uma lista de tamanho **N**.

Entrada

A entrada é composta de apenas um caso de teste com 2 linhas.

Na primeira linha há um inteiro **N**, que consiste no comprimento da lista **L**.

A segunda linha terá uma sucessão de **N** valores inteiros separados por um espaço em branco cada, representando os **N** elementos da lista **L**.

Restrições

- $1 \leq \mathbf{N} \leq 1000$
- $-10000 \leq \mathbf{L_i} \leq 1000$

Saída

Seu programa gera como saída um único número inteiro, representando a quantidade de movimentos que o algoritmo de ordenação Insertion Sort realizou com os elementos da lista **L** para torna-la ordenada. Após imprimir este valor salte uma linha.

Exemplos

Entrada	Saída
5 2 1 3 1 2	4

Entrada	Saída
4 4 4 3 4	2