USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação

LABORATÓRIO DE INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO II

SCC0220 - 2021/2

PRIMEIRA IMPLEMENTAÇÃO PRÁTICA

ORDENAÇÃO ACONCHEGANTE

1 Introdução

Paulo é estudante de Oceanografia e é fascinado por um dos mais perfeitos objetos da natureza: as conchas. Durante sua graduação e em suas aulas práticas que o permitiam entrar em contato com elas, Paulo construiu uma coleção de inúmeras conchas, em uma seção de prateleiras em uma das paredes de seu quarto.

O estudante, ávido na organização, costuma ter uma padronização específica na sua prateleira de conchas: Paulo sempre organiza suas conchas de acordo com diâmetro vertical, da menor para a maior (ordem crescente). Para ter controle sobre isso, o estudante anota essa informação em seu caderno a cada concha nova que pega para sua coleção, juntamente com outras informações: ele também anota a cor e a textura de cada uma.

Por acidente e por desgaste, a prateleira de Paulo soltou-se de um lado e ficou inclinada, de maneira que suas conchas fossem todas para o chão. O garoto, ao ver isso, tomou um choque, uma vez que suas conchas agora estão todas embaralhadas e, como seu amigo, você decide ajudá-lo a remontar a prateleira de sua coleção com base nas informações anotadas no caderno.

O processo de reordenação

Paulo percebeu que o processo seria muito demorado caso todas as conchas tivessem de ser comparadas entre si, e assim elaborou um método de ordenação que julgou mais eficiente, cuja principal heurística se baseava em dividir a lista total de conchas em listas menores. Ao lhe explicar, foi explicado que:

1. Dada uma lista A com n conchas, divida-a em $\lfloor n/2 \rfloor$ sub-listas virtuais, de maneira que um elemento na posição k pertença à sub-lista de número "$k \bmod \lfloor n/2 \rfloor$". O operador "mod" indica o resto da divisão de k por $\lfloor n/2 \rfloor$. Essa é uma maneira matemática de dizer que estamos agrupando em sub-listas os elementos que possuem distância $D = \lfloor n/2 \rfloor$ entre si, sendo n o tamanho da lista inicial.

- Por exemplo, para uma lista de valores $[1, 7, 8, 5, 3, 4, 6]$, as sub-listas formadas seriam três: $[1, 5, 6]$, $[7, 3]$, $[8, 4]$. Note que as sub-listas são apenas virtuais: não estamos criando novas listas de fato, apenas interpretando logicamente a lista original dessa forma.

2. Para cada uma das sub-listas, realize um **Insertion Sort** para que elas sejam ordenadas. A ideia é que a ordenação, ao contrário da usual, seja feita apenas entre esses elementos que possuem distância $\lfloor n/2 \rfloor$ entre si.

3. Volte ao passo 1 e repita o processo, dessa vez considerando que $n \coloneqq \lfloor n/2 \rfloor$, até que a distância D seja igual a 1 (incluso). Quando isso ocorrer, é como se estivéssemos realizando um Insertion Sort na lista completa, sem divisões.

   - Com esse processo, a distância D que estamos considerando tende a diminuir a cada iteração: começa com $n/2$, depois $n/4$, depois $n/8$ e assim sucessivamente (sempre arredondando para baixo) até o momento em que ela, eventualmente, será igual a 1.

4. Chega-se ao caso trivial: a lista original está ordenada.

Para automatizar esse processo, você resolveu construir um código em linguagem C para solucionar os problemas das conchas de Paulo, **utilizando o método de ordenação descrito por ele como base**.

2 Informações adicionais

- A ordenação deve possuir complexidade de tempo de, no máximo, n^2 — isto é, o algoritmo não pode performar pior do que algoritmos como Insertion Sort ou Bubble Sort. Casos de teste foram elaborados para facilitar sua análise de eficiência do algoritmo. No caso de *timeout*, é um sinal de que seu algoritmo não está de acordo.

- A ordenação deve possuir complexidade de espaço constante — isto é, o algoritmo não deve utilizar memória auxiliar para realizar a ordenação. Algoritmos como o Merge Sort, por exemplo, utilizam vetores auxiliares para realizá-la (e nesse caso isso não é permitido).

3 Entrada

De entrada, será dado um inteiro n que representa o número de conchas a serem ordenadas. Logo após, serão dadas triplas de dados, na mesma linha e separados por espaços, contendo respectivamente: o diâmetro vertical da concha, sua cor e sua textura.

- O valor de n é um **inteiro positivo**. Isso é assegurado pelas entradas que serão proporcionadas.

- O diâmetro vertical é dado por um número **real** (`double`) positivo, com precisão de 3 casas decimais.

- A cor e a textura são **cadeias de caracteres** de apenas uma palavra, com caracteres da tabela ASCII.
- Não há **repetição** nos valores dos diâmetros verticais.

4 Saída

A saída deverá listar todas as n conchas de entrada, ordenadas de acordo com seu diâmetro vertical, em ordem crescente. Juntamente ao diâmetro vertical, devem ser mostradas a cor e a textura da concha, respectivamente, na mesma linha e todos separados por um espaço.

5 Exemplos de entrada e saída

Entrada

```
10
44.528 Magenta Rough
9.214 Crimson Smooth
12.503 Lime Rough
47.930 Red Wavy
14.125 Purple Rough
9.730 Purple Rough
3.565 Crimson Rough
27.312 Magenta Wavy
20.751 Orange Smooth
2.612 Purple Rough
```

Saída esperada

```
2.612 Purple Rough
3.565 Crimson Rough
9.214 Crimson Smooth
9.730 Purple Rough
12.503 Lime Rough
14.125 Purple Rough
20.751 Orange Smooth
27.312 Magenta Wavy
44.528 Magenta Rough
47.930 Red Wavy
```

Cuidado: alguns dos casos de entrada podem ser grandes!

Bom trabalho!