UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ALGORITMOS E ESTRUTURAS DE DADOS III

1º semestre de 2023
Professor: Leonardo Chaves Dutra da Rocha

Trabalho Prático 1
Data de Entrega: 20 de Junho 2023. Trabalho Dupla

Esse trabalho prático tem por objetivo ampliar os conhecimentos adquiridos em sala sobre processamento de cadeias de caractéres. O aluno terá a oportunidade de implementar e comparar vários métodos de casamento de padrões.

Descrição

Há muito tempo atrás, existia um lugar na Terra no qual a mágica era verdadeira. Para os habitantes do reino Xulambis, era algo bem comum conviver com diferentes poderes, e todos recebiam sua pedra mágica quando completavam a maior idade.

Cada pedra era composta por um conjunto de símbolos coloridos. De acordo com a sequência dos símbolos um poder diferente podia surgir. Por exemplo, uma sequência de símbolos azul, azul, branco, branco, verde, vermelho dava ao usuário o poder da velocidade. As inscrições eram geradas de maneira aleatória, sendo assim existiam pedras que não conferiam nenhum poder.

Com o tempo o segredo da magía, continuou oculto e perdido. Porém recentemente, um grupo de pesquisadores fascinados com o mundo antigo, em uma de suas explorações, encontraram o templo do reino de Xulambis. E após vasculharem os livros, descobriram um dicionario que relacionava cada conjunto de símbolos a um poder. Agora estão decididos a separarem as pedras que realmente tem poderes das que não tem, para assim provarem ao mundo que a magía é real.

O problema é que as pedras podem ter milhares de símbolos, e verificar manualmente se a pedra possui alguma habiliadade é um processo custoso e muito sujeito a erros humanos. Sua tarefa é construir soluções que dados a sequência do poder e a descrição da pedra, determinar se aquela sequência esta presente na pedra.

O que deve ser feito

Você deve propor, implementar e avaliar PELO MENOS TRÊS algoritmos que resolvam o problema dos pesquisadores.

Na avaliação das estratégias, você deve considerar sua análise de complexidade feita sobre as estratégias implementadas comparando-as com os tempos reais de execução (utilize a rotina gettimeofday()). A ideia de tal comparação é tentar mostrar que sua análise de complexidade está correta bem como determinar qual dos métodos acima obtém o melhor desempenho na resolução do problema proposto.

Entrada e Saída

O arquivo executável deve ser chamado de tp3 e deve receber dois parâmetros: nome do arquivo de entrada e um inteiro que representa uma das estratégias implementadas. Exemplo:

./tp3 entrada.txt 1

A entrada do programa deve ser lida de um arquivo de texto. A entrada é composta por vários casos de teste. A primeira linha contém um inteiro T, que representa o número de casos de teste. Cada uma das T linhas seguintes representa um caso de teste e possui duas cadeias de caractéres separadas por um espaço. A primeira representa a sequência da habilidade e a segunda a descrição da pedra. Ambas as cadeias são compostas de letras minúsculas a-z, cada letra representa uma cor distinta. A pedra é esférica: o simbolo representado pelo ultimo caractere é adjacente àquele representado pela primeira. A sequencia do poder possui entre 1 e 10^2 caracteres, inclusive. A pedra possui entre 1 e 10^4 caracteres inclusive.

Para cada caso de teste, imprima uma linha contendo S se a sequência está presente, junto com a posiç onde começa, naquela pedra e N se ela não está presente.

A saída deve ser em um arquivo texto e o nome do arquivo de saída é o nome do arquivo de entrada porém com a extensão "out".

Exemplo de entrada:

```
4
ava av
patapon npatapatapatapo
isitfriday ohnoitisnt
haskell lleksah
```

Exemplo de saída

```
S 1
S 10
N
S 7
```

Na tela, o programa deve imprimir apenas os tempos de usuário e os tempos de sistema para comparação. Para avaliação do tempo, utilize as funções *getrusage* e *gettimeofday*.

Documentação

Deve ser clara e objetiva, descrevendo as soluções adotadas e justificando bem as escolhas realizadas. Devem possuir também uma análise de complexidade detalhada das soluções. Em termos de análise de resultados, avalie o desempenho e funcionamento de seus algoritmos para diversas configurações e avalie também o tempo de execução dos mesmos (compare-os). Lembre-se, o importante é você apresentar maturidade técnica em suas discussões.

Observações:

- O código fonte do trabalho deve ser submetido para compilação e execução em ambiente Linux, tendo como padrão os computadores dos laboratórios do DCOMP.
- Deve ser escrito na linguagem C (trabalhos implementados em outras linguagens como C++/Java/Python e outras não serão aceitos);
- As estruturas de dados devem ser alocadas dinamicamente e o código deve ser modularizado utilizando os arquivos .c .h.
- O utilitário Make deve ser utilizado para compilar o programa;

- A saída deve ser impressa no arquivo pedido seguindo estritamente o formato da especicação caso contrário o resultado será considerado errado;
- O arquivo executável deve ser chamado de **tp1** e deve receber como parâmetro apenas o nome do arquivo de entrada de dados. Não serão aceitos outros nomes de executáveis além dos mencionados.
- Faça seu código de forma legível

Avaliação

Deverão ser entregues:

- listagem das rotinas;
- documentação contendo:;
  - descrição das soluções e estruturas de dados utilizadas;
  - análise da complexidade das rotinas;
  - análise dos resultados obtidos.
  - a documentação não pode exceder 12 páginas.

Distribuição dos pontos:

- execução (E)
  execução correta: 80%
- estilo de programação
  código bem estruturado: 10%
  código legível: 10%
- documentação (D)
  comentários explicativos: 30%
  análise de complexidade: 30%
  análise de resultados: 40%

A nota final é calculada como a média harmônica entre execução (E) e documentação (D):

$$\frac{D * E}{\frac{D+E}{2}}$$