Universidade Federal de Viçosa – Campus UFV-Florestal
Ciência da Computação – Projeto e Análise de Algoritmos
Professor: Daniel Mendes Barbosa

Trabalho Prático 2

Este trabalho é **obrigatoriamente em dupla** e deverá ser entregue no PVANet de acordo com as instruções presentes no final da especificação. Cada tarefa (A e B) é um programa separado a ser feito.

Tarefa A) Nesta tarefa você deverá implementar um "caça-palavras" utilizando **backtracking**, mas com um funcionamento diferente do caça-palavras padrão. No jogo padrão as palavras são dispostas na horizontal, vertical ou diagonal. Nesta versão que você irá implementar, as palavras deverão ser encontradas da seguinte forma: começar a pesquisa pela primeira linha e à esquerda; a cada letra da palavra de entrada que for encontrada, a próxima letra deverá ser buscada sempre abaixo, à esquerda ou à direita da anterior, mas sempre com uma restrição: dado um movimento que tenha sido feito, este movimento não pode repetir imediatamente, ou seja, você pode encontrar uma letra, encontrar a segunda abaixo, mas a terceira não poderia ser abaixo. Obviamente não se pode ter um movimento à esquerda seguido de um à direita imediatamente e vice-versa.

O arquivo de entrada, no formato texto, deverá ter dois números inteiros na primeira linha, separados por um espaço, sendo o número de linhas e colunas respectivamente do quadro a ser verificado. A partir da segunda linha, o arquivo conterá as letras que formam o quadro, sem espaços. As letras possíveis serão as 26 letras sem acentuação, sendo maiúsculas e minúsculas diferenciadas (*case sensitive*). O usuário digitará o nome do arquivo de entrada, e o mesmo será carregado para as suas estruturas de dados.

A palavra de entrada deverá ser digitada e então exibido o resultado de ocorrências encontradas no quadro que foi previamente carregado, mostrando "*" no lugar das letras que não casaram.

Exemplo de arquivo de entrada para o quadro a ser verificado:

```
5 5
kcjek
casmy
asahe
nrhwd
bvhfj
```

Imaginando que a palavra a ser encontrada seja "casa", teríamos o seguinte resultado:

```
Foram encontradas 3 ocorrências para a palavra casa

*c***
cas**
asa**
*****
```

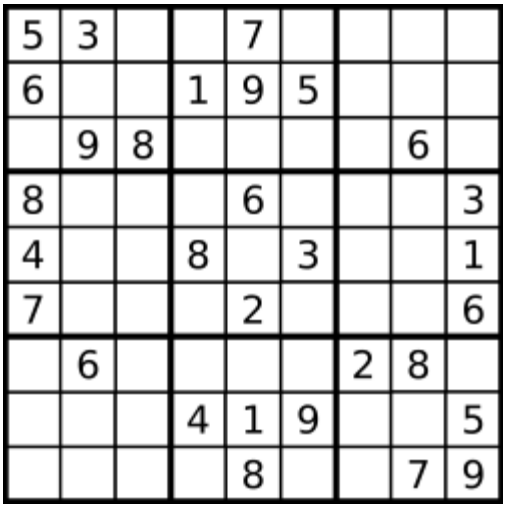
Maiores detalhes de interface e outras opções do programa ficarão a critério da dupla.

Você não poderá escrever nos arquivos de entrada e nem em outros arquivos, ou seja, só poderá abri-los no modo "r" (somente leitura). Para armazenar os dados do arquivo no programa, para poderem ser processados, você deverá **alocar memória dinamicamente**, visto que não se sabe a princípio o tamanho do arquivo (sabe-se apenas quando for lida sua primeira linha).

Considere que os arquivos de entrada seguirão fielmente o formato que foi definido.

Além disso, seu programa deverá ter um #define para configurar se o programa estará no **modo análise** ou não. Se não estiver no modo análise, a execução será como descrito anteriormente. Se estiver no modo análise, deverá fazer tudo mas também contabilizar o número total de chamadas recursivas que foram feitas. O programa irá contabilizar isso e imprimir na tela somente se o modo análise estiver ligado. Deverá ser estudada melhor forma de se fazer isso com #define e explicar no relatório como utilizar (compilar o programa em modo análise ou não). O modo análise poderá ser disponibilizado também através de opções no programa, devidamente documentadas.

Tarefa B) Implementar em C um algoritmo com **backtracking** para resolver puzzles sudoku, puzzle já discutido na disciplina. Abaixo um exemplo:



Não podem haver números iguais em uma mesma linha, em uma mesma coluna ou em um mesmo grid (quadro 3 por 3). Considere sempre a versão do sudoku com quadros 9 por 9, e podendo ser preenchido com números de 1 a 9.

Assim como na Tarefa A, deverá haver um **modo análise**, que neste caso deverá exibir quantas tentativas foram feitas (quantos preenchimentos foram considerados até se achar a solução).

A interface e como exibir o resultado também ficará a critério da dupla.

Faça exatamente o que está sendo pedido neste trabalho, ou seja, mesmo que você tenha uma idéia mais interessante para o programa, você deverá implementar exatamente o que está definido aqui no que diz respeito ao problema em si e ao paradigma backtracking. No entanto você pode implementar algo além disso, desde que não atrapalhe a obtenção dos resultados necessários a esta especificação, o que pode eventualmente render pontos extras.

Formato e data de entrega:

Você deverá entregar todo o **código-fonte produzido (de preferência os dois projetos inteiros do Codeblocks)**, que será testado no sistema operacional **Linux**, bem como um **relatório** de documentação, que deverá conter para cada tarefa:

- explicação do algoritmo projetado, de acordo com o esqueleto e conceitos de backtracking;
- implementação do algoritmo projetado (estruturas de dados criadas, etc);
- resultados de execução, mostrando entrada e saída, no modo normal e modo análise;
- arquivos de entrada usados nos testes.
- explicação de como compilar o programa em modo análise.
- nos resultados deverá constar a quantidade total de chamadas recursivas no modo análise.

Obs.: as diretrizes básicas para a documentação continuam válidas para este trabalho.

Importante: o arquivo a ser entregue no PVANet (até a data e horário limite lá estabelecidos) deverá ser um arquivo .zip contendo todo esse material produzido. O nome do arquivo deverá ter o nome e sobrenome dos membros da dupla. Exemplo: se os nomes dos alunos forem fulano jobs e beltrano jobs, o nome do arquivo deverá ser **fulanojobs-beltranojobs.zip**.

Bom trabalho!