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

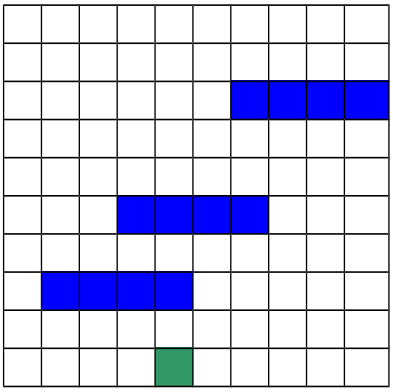
# Trabalho Prático 2

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

Tarefa A) Projetar um algoritmo com backtracking para o contexto abaixo, implementá-lo em C e documentá-lo, de acordo com as especificações abaixo.

1) Contexto do trabalho

Observe a figura abaixo:



Esta tabela representa um espaço geográfico, onde cada célula representa um quadrado de 1 metro quadrado. Temos neste espaço um cachorro, cuja posição inicial está representada por um quadrado verde (ou seja, o cachorro está inicialmente nesta posição). O cachorro deverá caminhar pelos vários quadrados (ou seja, várias posições) até chegar em uma das células da primeira linha da tabela (linha de cima).
Os quadrados azuis representam paredes, pelas quais o cachorro obviamente não pode passar. O cachorro só pode movimentar para cima, para baixo e para os lados. Ou seja, não pode se movimentar nas diagonais.

Você deverá escrever um programa na linguagem C que utilize um algoritmo projetado por você, que leia um arquivo com as informações do local onde o cachorro está bem como sua exata posição inicial. Seu programa deverá então movimentar o cachorro até que ele chegue em uma das células da primeira linha da tabela, mostrando na tela cada movimentação feita.

Seu programa deverá obrigatoriamente usar **backtracking**. Uma função recursiva chamada **movimenta_cachorro** deverá ser criada. Isso significa que primeiramente você

deverá encontrar a posição inicial do cachorro. Quando encontrar deverá chamar esta função uma única vez, e a partir daí ela chamará ela mesma, até que o cachorro chegue na primeira linha (linha zero).

Importante:
- você deverá definir as estruturas de dados necessárias ao algoritmo;
- na **documentação** você deverá explicar seu algoritmo com base nos conceitos de backtracking, e como ele foi implementado;

2) Formato de entrada de dados

O espaço geográfico será a entrada para seu programa, a partir de um arquivo texto. O arquivo terá um formato padronizado, sendo que na primeira linha do arquivo temos o número de linhas, um espaço e um número de colunas.
Nas linhas seguintes deverão ser informadas as cores de cada uma das células, de cada linha, sendo que as células de cada linha **não** são separadas por espaço. Cada cor será representada por um número de 1 a 3, sendo:

1-branco (célula vazia, por onde o cachorro pode passar)
2-verde (célula onde o cachorro está inicialmente)
3-azul (célula ocupada por parede)

Veja o exemplo abaixo, que seria o conteúdo do arquivo de entrada referente ao primeiro espaço geográfico mostrado anteriormente:

```
10 10
1111111111
1111111111
1111113333
1111111111
1111111111
1113333111
1111111111
1333311111
1111111111
1111211111
```

Este exemplo possui 10 linhas com 10 colunas cada. Mas estes valores podem ser valores quaisquer, e o espaço geográfico pode ser retangular (número de linhas diferente do número de colunas).

3) Formato de saída de dados

O programa deverá imprimir a resposta na tela. Cada posição ocupada pelo cachorro deverá ser impressa em uma linha da saída. Imaginando que o cachorro tenha começado

na linha 9 e coluna 4, e depois tenha se movimentado para a linha 8 e coluna 4, as duas primeiras linhas do resultado seriam, portanto:

```
Linha: 9 Coluna: 4
Linha: 8 Coluna: 4
```

Ao final da execução e da impressão de todas as células pelas quais o cachorro passou, deverá ser impressa a quantidade total de movimentos feitos e em qual coluna da primeira linha o cachorro chegou. Exemplo:

```
O cachorro se movimentou 25 vezes e chegou na coluna 7 da primeira linha
```

Observe que muitas vezes o cachorro pode seguir por um caminho sem saída, e aí deverá voltar e achar outro caminho. Todas essas movimentações deverão ser contabilizadas no total de movimentações.

**Importante**: além disso o labirinto pode não ter saída! Nesse caso o cachorro pára na célula em que estiver quando perceber que não há saída e será impresso pelo programa (por exemplo):

```
O cachorro se movimentou 25 vezes e percebeu que o labirinto nao tem saida.
```

4) Aparência geral do programa:

Portanto, o programa deverá mostrar as seguintes opções para o usuário:

```
PROGRAMA Labirinto: Opcoes do programa:
1) Carregar novo arquivo de dados.
2) Processar e exibir resposta.
3 ou qualquer outro caracter) Sair do programa.
Digite um numero:
```

**Obs.:Todos esses exemplos de interface são sugestões. Desde que as funcionalidades estejam presentes, a dupla pode personalizar a interface, justificando as alterações, mas de forma que melhore o trabalho. Isso pode ser levado em consideração na correção.**

Veja abaixo exemplos de execução de cada opção (cada opção limpa a tela inicialmente):

Escolhendo a opção 1:

```
Por favor digite o nome do arquivo:
```

Escolhendo a opção 2, o arquivo deverá ser processado e a saída exibida conforme explicado na seção anterior.

Caso a opção 2 seja escolhida antes de carregar qualquer arquivo, deverá ser impresso:

```
Por favor carregue antes um arquivo de dados!
Pressione qualquer tecla para continuar...
```

Escolhendo a opção 3, o programa é encerrado.

Lembrando que enquanto o programa não for encerrado, o usuário poderá carregar novos arquivos e processá-los.

**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 e o nível máximo de recursão alcançado durante toda a solução. 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).

A interface e como exibir o resultado também ficará a critério da dupla. Poderá ser oferecida mais de uma forma de se exibir os resultados, podendo o usuário escolher qual formato deseja.

Vocês deverão ainda criar uma opção (ou um outro programa) para a geração de labirintos de teste, considerando todos os dados envolvidos e o formato, como descrito acima. Labirintos não necessariamente possuem saída. Seu programa de geração de arquivos de teste deverá ter alguns parâmetros de configuração, como largura e altura do labirinto, e "dificuldade" do labirinto, entre outros, que vocês poderão colocar.

Tarefa B) Implementar em C a solução para o problema das N rainhas com backtracking, escolhendo e documentando os formatos de entrada e saída de dados do programa. Deverá haver a possibilidade de escolha de tamanho do tabuleiro. Deverá ser contabilizada uma estatística da execução e impressa ao final: quantidade total de colocações de peças (usados e descartados).

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. Poderá ser oferecida mais de uma forma de se exibir os resultados, podendo o usuário escolher qual formato deseja.

**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.**

**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;
- implementação do algoritmo projetado (estruturas de dados criadas, etc);
- resultados de execução, mostrando entrada e saída;
- 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 e o nível máximo de recursão obtido para cada teste através da execução no modo análise.

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!