Universidade Federal de Viçosa
Departamento de Informática
INF 112 – Programação 2
Trabalho prático 1
Data de entrega: 27/10/2017

---

# Trabalho Prático 1

Um passatempo bastante famoso, chamado de “fill a pix” (disponível em : http://www.conceptispuzzles.com/index.aspx?uri=puzzle/fill-a-pix ), consiste em, dada uma matriz contendo números (e algumas células vazias), descobrir quais células devem ser pintadas e quais células devem ser deixadas vazias. Para solucionar o jogo, deve-se observar a seguinte regra: deve-se pintar os quadrados que estão em volta de células numeradas de forma que o número de cada célula indique exatamente quantos pontos pintados há no quadrado 3x3 centrado nela. Por exemplo, se uma célula *c* possui o número 9, então todas as células em torno de *c* (incluindo *c*) devem ser pintadas. Por outro lado, se uma célula *d* possui número 0, então nenhuma célula vizinha a *d* (incluindo *d*) deve ser pintada.

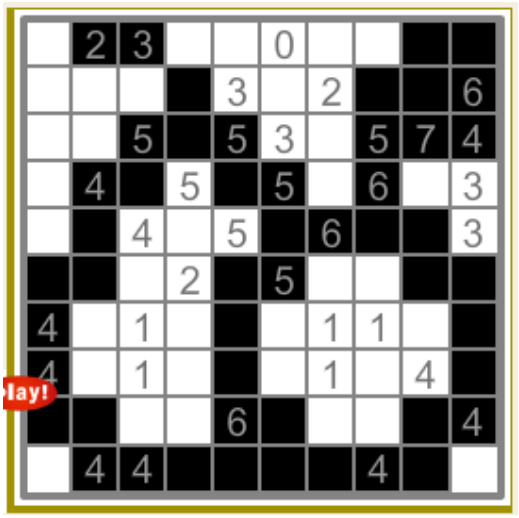


Figura 1: Exemplo de jogo (fonte: http://www.conceptispuzzles.com/index.aspx?uri=puzzle/fill-a-pix )

Como o objetivo do jogo é basicamente gerar uma “imagem” que satisfaça os números das células, uma possível forma de resolvê-lo seria gerar todas as possíveis imagens em preto e branco (com a mesma resolução da matriz do jogo) e, então, testar cada uma delas para verificar se a solução é válida. Porém, o número de possíveis soluções a serem avaliadas seria $2^{(n \times n)}$ (por exemplo, se a matriz tiver dimensões 15x15, há 5391989333430127958933403017403926134727428884508114496220722049843 2 possíveis soluções!) e, portanto, um método de “força bruta” seria EXTREMAMENTE ineficiente (considerando uma matriz com dimensões 15x15, um computador capaz de executar 1 bilhão de avaliações por segundo gastaria 17097886014174682771858837574213362933559832840 27 milênios para avaliar todas as soluções).

**Tarefa 1:** Neste trabalho, você deverá desenvolver um algoritmo utilizando a **estratégia de backtracking** para resolver esse problema de forma eficiente. Mais especificamente, o programa deverá ler a partir da entrada padrão (utilizando o *cin*) o número *C* de colunas da matriz e, a seguir, o número *L* de linhas. Após esses dois números, haverá *L* linhas cada uma contendo *C* números. Tais números representarão o tabuleiro do jogo, sendo que o valor -1 será utilizado para indicar células vazias.

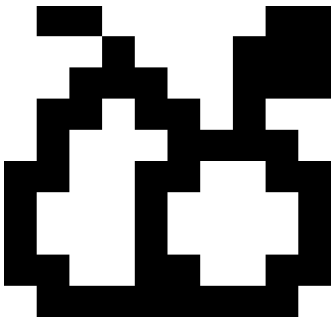
Por exemplo, a entrada abaixo ilustra o problema exibido na Figura 1:

```
10 10
-1 2 3 -1 -1 0 -1 -1 -1 -1
-1 -1 -1 -1 3 -1 2 -1 -1 6
-1 -1 5 -1 5 3 -1 5 7 4
-1 4 -1 5 -1 5 -1 6 -1 3
-1 -1 4 -1 5 -1 6 -1 -1 3
-1 -1 -1 2 -1 5 -1 -1 -1 -1
4 -1 1 -1 -1 -1 1 1 -1 -1
4 -1 1 -1 -1 -1 1 1 -1 4 -1
-1 -1 -1 -1 6 -1 -1 -1 -1 4
-1 4 4 -1 -1 -1 -1 4 -1 -1
```

Como saída, seu programa deverá gravar na saída padrão (utilizando o cout) uma imagem no formato “pbm”. Tal imagem deverá representar a solução para o jogo. Por exemplo, a saída para a entrada anterior deveria ser:

```
P1
10 10
0 1 1 0 0 0 0 0 1 1
0 0 0 1 0 0 0 1 1 1
0 0 1 1 1 0 0 1 1 1
0 1 1 0 1 1 0 1 0 0
0 1 0 0 0 1 1 1 1 0
1 1 0 0 1 1 0 0 1 1
1 0 0 0 1 0 0 0 0 1
1 0 0 0 1 0 0 0 0 1
1 1 0 0 1 1 0 0 1 1
0 1 1 1 1 1 1 1 1 0
```

Ao salvar essa saída em um arquivo “pbm”, a visualização dele em um visualizador de imagens seria:



A seguir, temos um exemplo de execução do programa que você deverá desenvolver:

```
$time ./exercicio1.exe < entrada10_10.txt  > saida.pbm

real    0m0.026s
user    0m0.024s
sys     0m0.000s
```

**Tarefa 2:** Desenvolver um algoritmo utilizando outra estratégia para resolver o problema “fill a pix” de forma eficiente. Na documentação, você deverá apresentar o algoritmo implementado e a fonte do mesmo.

**Tarefa 3:** Comparar o desempenho das duas estratégias, utilizando pelo menos 10 exemplares do jogo para cada uma das dimensões: 10x10, 15x15, 20x20. Apresente gráficos e/ou tabelas para os 3 tempos mostrados no exemplo de execução (real, user, sys). Compare o desempenho médio das 2 estratégias para cada tamanho do jogo (10x10, 15x15, 20x20). Faça uma discussão dos resultados.

**Entrega:** devem ser entregues, através do PVAnet, o código fonte com seu respectivo Makefile e uma documentação em formato pdf com todo o memorial descritivo da solução, contendo todas as decisões de implementação e detalhes das

estratégias de solução.

Observem que o desafio do trabalho está não só na implementação das estratégias de resolução dos problemas, mas também na avaliação, comparação e discussão de tais estratégias. Dessa maneira, na distribuição de pontos a documentação receberá, no mínimo, 50% dos pontos totais. Uma documentação de boa qualidade deverá apresentar não só as respostas para o jogo, mas principalmente discussões e conclusões críticas, considerando qual o algoritmo é superior em desempenho.

**Observe que seu trabalho deverá:**
- Ser compatível com o sistema operacional Linux (ou seja, se você o desenvolver em outro Sistema Operacional você deverá testá-lo também no Linux e enviar um Makefile compatível com o Linux).
- Ser executado em tempo razoável (serão 10 min por entrevista).
- Ler a entrada e gerar uma saída em formatos EXATAMENTE iguais aos formatos especificados neste documento.
- Estar bem indentado, organizado e apresentando comentários explicativos no código fonte.