Departamento de Matemática da Universidade de Coimbra		
2022/2023	**Métodos de Programação II**	**Projecto 2 (2022/12/13)**

Geração de Labirintos

Em Creta onde o Minotauro reina atravessei a vaga
De olhos abertos inteiramente acordada
Sem drogas e sem filtro
Só vinho bebido em frente da solenidade das coisas
Porque pertenço à raça daqueles que percorrem o labirinto,
Sem jamais perderem o fio de linho da palavra

Excerto de *Em Creta*, Sophia de Mello Breyner Andresen

Um labirinto pode ser gerado[4] começando com um arranjo predeterminado de células (mais comumente uma grade retangular, mas outros arranjos são possíveis) com locais de parede entre eles. Este arranjo predeterminado pode ser considerado como um grafo conexo com os arcos representando possíveis locais de parede e os nós que representam as células.

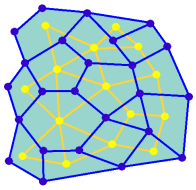

Figura 1: Geração de um Labirinto

A figura 1 ilustra as etapas de geração do labirinto para um grafo que, neste caso em particular, não é uma grade retangular.[5] Primeiro, o computador cria aleatoriamente um grafo planar G mostrado em azul, e seu complementar F mostrado em amarelo. Em segundo lugar, o computador percorre F usando um algoritmo escolhido, como uma pesquisa, colorindo o caminho de vermelho. Durante a travessia, sempre que um arco vermelho cruza um arco azul, o arco azul é removida. Finalmente, quando todos os nós de F foram visitados, F é apagado e dois arcos de G, um para a entrada e outro para a saída, são removidas.

Pesquisa em Profundidade, Aleatória O algoritmo, também conhecido como algoritmo de "retrocesso recursivo", é uma versão aleatória do algoritmo de busca em profundidade.

Frequentemente implementada com uma pilha, essa abordagem é uma das maneiras mais simples de gerar um labirinto usando um computador. Considere o espaço para um labirinto sendo uma grande grade de células (como um grande tabuleiro de xadrez), cada célula começando com quatro paredes. A partir de uma célula aleatória, o computador seleciona uma célula vizinha aleatória que ainda não foi visitada. O computador remove a parede entre as duas células e marca a nova célula como visitada e a adiciona à pilha para facilitar o retrocesso. O computador continua esse processo, com uma célula que não tenha vizinhos não visitados sendo considerada um beco sem saída. Quando em um beco sem saída, ele retrocede pelo caminho até chegar a uma célula com um vizinho não visitado, continuando a

[4] `https://en.wikipedia.org/wiki/Maze_generation_algorithm`
[5] `wikimedia.org/w/index.php?curid=35892827`

geração do caminho visitando essa nova célula não visitada (criando uma nova junção). Esse processo continua até que todas as células tenham sido visitadas, fazendo com que o computador volte atrás até a célula inicial.

Implementação recursiva. O algoritmo de busca em profundidade para geração de labirinto é frequentemente implementado usando «backtracking». Isso pode ser descrito com uma rotina recursiva a seguir:

```
Dada uma célula como parâmetro
Marcar a célula como visitada
Enquanto a célula corrente tiver células vizinhas não visitadas
   Escolha um dos vizinhos não visitados
   Remova a parede entre a célula e a célula escolhida
   Invocar a rotina recursivamente para uma célula escolhida
```

que é invocado uma vez para qualquer célula «exterior» da grelha.

Construa um programa que, baseado na estratégia descrita acima, seja capaz de gerar labirintos rectangulares de uma dada dimensão. Por exemplo, 6×6.

```
+   +---+---+---+---+---+
|   |                   |
+   +---+   +---+   +   +
|           |   |   |   |
+---+   +---+   +   +   +
|       |           |   |
+---+---+---+---+   +   +
|                   |   |
+---+---+---+---+   +---+
|       |       |       |
+---+   +   +   +   +   +
|           |       |   |
+---+---+---+---+---+   +
```

O projecto deve estar estruturado da seguinte forma:

- Programa principal que lida com as leituras e escritas, assim como o invocar da função de geração do labirinto.
- T.A.D. Grafo, que além das operações elementares sobre grafos, a saber: `criaGrafo`; `insereNo`; `retiraNo`; `insereArco`; `retiraArco`; `vazio`, deve conter as outras funções necessárias para a geração do labirinto.
- Outros T.A.D. necessários para a implementação das funções de geração do labirinto, por exemplo o T.A.D. Pilha.

Nas operações com as pilhas o, T.A.D. Pilha deve ser usado, não se recorrendo a um acesso directo à estrutura pilha.

Datas

- Data de disponibilização do enunciado: 28/11/2022.
- Prazo para entrega: 10/1/2023 (às 24h00).

Material a Entregar

1. Documente o seu programa. Tanto em termos de documentação interna, como de documentação externa, esta última na forma de um pequeno manual de utilização (relatório, max. 10pg.).

2. É obrigatório a organização do código em termos de um conjunto de ficheiros ≪`.h`≫ e ≪`.c`≫. É obrigatório a apresentação de uma `Makefile` que automatize o processo de compilação.

3. Deve entregar (por correio electrónico) um arquivo `zip`, contendo os ficheiros referentes ao programa (`Makefile, .c, .h`), assim como o ficheiro referente ao relatório (formato `PDF`), até às 24h00 do último dia do prazo. O nome do ficheiro a entregar deve ser `proj2MPIIGrpN.zip`, com N o número do grupo de trabalho.

4. Os código entregue pode ser desenvolvido em qualquer sistema computacional que contenha um compilador de *C*, mas deve compilar e os correspondentes programas devem funcionar nos computadores do laboratório de cálculo (sistemas Linux).

Referências

[1] Harary, Frank *Graph theory* Reading, Mass: Addison-Wesley, 1972. (biblioteca DMUC - 05-01/HAR/3ª imp)

[2] Pereira, José Manuel dos Santos Simões. *Grafos e redes: teoria e algoritmos básicos.* Rio de Janeiro : Interciência, 2013. (biblioteca DMUC - 94C/PER)

[3] Quaresma, Pedro. *Métodos de Programação II.* Apontementos da disciplina, Departamento de Matemática, Universidade de Coimbra, 2020.

A Algoritmos

Construção Um possível algoritmo para construir um labirinto "perfeito" (possível de ser resolvido) baseia-se em um algoritmo de "busca em profundidade". Partindo de um "grade" de células com todos os "muros" entre células levantados.

```
levantar muros entre todas as células
criar pilha posicoesCelulas
totalCelulas = número de células da grade
escolher aleatoriamente uma célula e denominá–la celulaCorrente
celulasVisitadas = 1
enquanto celulasVisitadas < totalCelulas faz
  encontrar todas as células vizinhas de celulaCorrente com muros intactos
  se uma ou mais foram encontradas entao
    escolher uma aleatoriamente e denominá–la proximaCelula
    derrubar o muro entre proximaCelula e celulaCorrente
    colocar a posição da celulaCorrente na pilha posicoesCelulas
    considerar proximaCelula como celulaCorrente
    celulasVisitadas = celulasVisitadas + 1
  senao
    retirar elemento da pilha posicoesCelulas, passa a ser celulaCorrente
  fimse
fimenquanto
```

Resolução Também é possível determinar a saída do labirinto a partir do algoritmo de "busca em profundidade".

```
marcar todas as células como não visitadas
criar pilha posicoesCelulas
marcar a célula de entrada do labirinto como visitada
considerar a célula de entrada como sendo a celulaCorrente
enquanto celulaCorrente diferente da célula de saída do labirinto faz
  encontrar todas as células vizinhas acessíveis de celulaCorrente
                                                  ainda não visitadas
  se uma ou mais foram encontradas entao
    escolher uma aleatoriamente e denominá–la proximaCelula
    marcar proximaCelula como visitada
    traçar segmento de reta entre celulaCorrente e proximaCelula
    empilhar a posição de celulaCorrente em pilhaDePosicoesDeCelulas
    considerar proximaCelula como celulaCorrente
  senao
    retirar célula da pilha posicoesDeCelulas, passa a ser celulaAnterior
    apagar segmento de reta entre celulaCorrente e celulaAnterior
    considerar celulaAnterior como celulaCorrente
  fimse
fimenquanto
```