Instruções sobre o Trabalho 2

481556 — Algoritmos e Complexidade

Cândida Nunes da Silva

1º Semestre de 2015

1 Torneio Tribruxo

Hogwarts é uma escola de magia e bruxaria famosa no mundo bruxo. A cada quatro anos a escola sedia um torneio para testar as habilidades dos alunos de várias escolas. Por se tratar de um torneio perigoso os alunos só podem participar se forem maiores de 17 anos. Harry Potter caiu em uma armadilha feita por Voldermort e foi obrigado a participar do torneio. Ele se saiu muito bem nas duas primeiras provas, mas precisa de ajuda na terceira prova.

Nessa tarefa os participantes entram num labirinto mágico de forma triangular. O primeiro nível do triângulo possui um compartimento, o segundo possui dois até se chegar no N-ésimo nível que terá N compartimentos.

Cada compartimento do labirinto possui apenas duas passagens, uma que leva para o compartimento da frente e outra que leva para o compartimento à esquerda do que está à frente, isto é, na diagonal. Em cada compartimento existe precisamente uma poção mágica, que pode aumentar ou diminuir a intensidade dos poderes mágicos de quem a toma. Ganha o competidor que conseguir encontrar o caminho do primeiro ao último nível do labirinto que maximiza a intensidade de seus poderes mágicos ao final do caminho.

A cada poção um número inteiro é associado, que pode ser positivo, negativo ou nulo, e indica a potência de aumento ou diminuição de intensidade de poderes que a poção propicia. A potência final de aumento ou diminuição de intensidade de poderes obtida após beber uma sequência de poções é dada pela soma das potências individuais de cada poção na sequência.

A figura abaixo ilustra um labirinto de 4 níveis. É possível notar que no primeiro compartimento a poção possui potência 2 e no próximo movimento o competidor pode ir para frente e obter uma poção de potência -2 ou ir para sua diagonal à esquerda e obter uma poção de potência -1.

2			
−2	−1		
4	9	4	
3	2	1	2

Figura 1: Labirinto

Harry Potter entende muito de magias, mas quase nada de algoritmos, e quer a sua ajuda para desenvolver um feitiço (mal sabe ele que é um simples algoritmo...) que o ajude a determinar

para qualquer configuração de labirinto qual é o caminho que tem a sequência de poções de maior potência final.

2 Entrada

A entrada deve ser lida da entrada padrão (teclado) e será composta por diversos casos de teste. A primeira linha de cada caso de teste contém um número N ($1 \leq N \leq 5000$) que indica o número de níveis do labirinto (o labirinto da figura possuí 4 níveis). As N linhas seguintes indicam a configuração do labirinto. A i-ésima linha na sequência possui i inteiros, sendo que todos esses inteiros são, em valor absoluto, menores ou iguas a 10^4.

O último caso de teste é seguido por uma linha contendo um zero.

3 Saída

A saída deve ser escrita na saída padrão (terminal). Para cada caso de teste a saída deve ter uma única linha contendo um único inteiro que é o valor da potência final de um caminho que contém a sequência de maior potência total.

4 Exemplo

Entrada	Saída
1 6982 2 3631 7441 3435 3 8341 7412 1907 7526 6773 9837 4 5534 4229 9558 6712 3851 2391 8731 5131 2207 9838 0	6982 11072 23279 27321

5 Desenvolvimento e Apresentação

Cada aluno deve implementar a sua solução individual. A implementação da solução do problema deve ser em C em arquivo único. O nome do arquivo deve estar na forma “t2-nomesn.c”, onde “nomesn” representa o primeiro nome do aluno seguido das iniciais de seu sobrenome. Note que todas as letras são minúsculas e o separador é “-” (hífen) e não “_” (underscore).

Serão disponibilizados arquivos com diversas entradas (t2.in) e as respectivas saídas esperadas (t2.sol). É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** entrada do arquivo de testes. É **desejável** que a implementação seja eficiente.

6 Ambiente de Execução e Testes

O programa deve ser compilável em ambiente Unix com `gcc`. Sugere-se que os testes também sejam feitos em ambiente Unix. Deve-se esperar que a entrada seja dada na entrada padrão (teclado) e não por leitura do arquivo de testes. Da mesma forma, a saída deve ser impressa na saída padrão (terminal), e não em arquivo. Será disponibilizado no moodle um trabalho modelo (trabalho 0) que faz a entrada e a saída da forma requerida.

A motivação dessa exigência é apenas simplificar a implementação de entrada e saída, permitindo o uso das funções `scanf` e `printf` da biblioteca padrão para leitura e escrita dos dados, sem precisar manipular arquivos.

Por outro lado, é evidente que efetivamente entrar dados no teclado é muito trabalhoso. Em ambiente Unix, é possível usar redirecionamento de entrada na linha de comando de execução para contornar esse problema. Supondo que o nome do arquivo executável seja análogo ao arquivo fonte, e "t2.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./t2-nomesn < t2.in
```

executa o programa para todos os casos de teste de uma só vez, retornando todas as saídas em sequência para o teminal. Novamente, pode-se usar o redirecionamento de saída na linha de comando para escrever a saída em um arquivo de saída de nome, por exemplo, "t2.my.sol". A respectiva linha de comando seria:

```
shell$ ./t2-nomesn < t2.in > t2.my.sol
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "t2.sol" contém as saídas esperadas, a linha de comando:

```
shell$ diff t2.sol t2.my.sol
```

serve para comparar automaticamente os dois arquivos, retornando nada caso sejam idênticos e as linhas onde há discrepâncias caso contrário.

7 Entrega e Prazos

A data para a entrega do projeto é dia 28 de maio. Cada aluno deve entregar via moodle seu único arquivo fonte com nome no padrão requerido até essa data. Lembre-se que é **imprescindível** que o código compile em ambiente Unix e que a entrada e a saída sejam feitas pela entrada e saída padrão.

8 Notas

As notas serão baseadas na correção da solução implementada, clareza do código fonte e eficiência da solução.

Trabalhos que não atendam aos requisitos mínimos expressos neste documento de forma a invibializar o teste do programa receberão nota ZERO. Em particular, receberá nota ZERO todo programa que:

- não compila em ambiente Unix;
- dá erro de execução;
- não usa entrada e saída padrão para leitura e escrita.