Projeto de Algoritmo com Implementação nº 2

MC458 — Projeto e Análise de Algoritmos I

Prof. Pedro J. de Rezende

2º Semestre de 2020

1 Introdução

Imagine um alpinista tentando escalar uma parede até o topo. Uma parede é construída a partir de blocos quadrados de mesmo tamanho, cada um provendo um apoio para a mão, sendo que alguns apoios são mais perigosos/complicados que outros. A partir de cada bloco (abaixo do topo), o alpinista pode alcançar três blocos da fileira acima: o bloco logo acima, o acima e à esquerda e o acima e à direita (a menos que o da esquerda ou direita não estejam disponíveis porque é o fim da parede). Mais formalmente, podemos considerar a parede como uma grade $n \times m$, na qual cada célula (i, j) possui um custo positivo $C(i, j)$ associado. A fileira da base (nível do chão) é a de número 1, e a do topo é a de número n. A partir da célula (i, j), com $i < n$, em um único movimento é possível alcançar as células $(i + 1, j - 1)$, se $j > 1$, $(i + 1, j)$ e $(i + 1, j + 1)$, se $(j < m)$. O objetivo é encontrar o caminho menos perigoso da base até o topo da parede, onde o grau de periculosidade (custo) de um caminho é dado pela soma dos custos dos blocos usados nesse caminho.

Considere o exemplo abaixo. O caminho de menor custo está indicado em vermelho. O custo deste caminho é 12.

2	8	9	**5**	8
4	4	6	**2**	3
5	7	5	6	**1**
3	2	5	**4**	8

Figura 1:

2 Problema

O objetivo aqui é encontrar o custo do caminho menos perigoso da base até o topo da parede, onde o grau de periculosidade (custo) de um caminho é dado pela soma dos custos dos blocos usados nesse caminho.

Obs.: Tentativas de apresentar resoluções advindas de implementações que produzem soluções corretas para as instâncias do problema mas **sem um projeto de algoritmo de acordo com as técnicas vistas na disciplina** para computação das respostas serão consideradas inválidas.

3 Entrada

A entrada de uma instância é composta de $n+1$ linhas. A primeira linha contém dois inteiros positivos, n e m, respectivamente. As n linhas seguintes correspondem à descrição da parede: cada linha contém, portanto, m inteiros positivos. Abaixo, segue a entrada correspondente ao exemplo ilustrado acima (observe a ordem em que são dadas as linhas):

4 5

3 2 5 4 8

5 7 5 6 1

4 4 6 2 3

2 8 9 5 8

O seu programa (`escalada`) deve receber sua entrada através da entrada-padrão, i.e., dado um arquivo (`instancia.in`) contendo a descrição de uma instância, você deve redirecionar tal arquivo como entrada para o seu programa utilizando o operador $<$:

```
./escalada < instancia.in
```

4 Saída

A saída deve conter apenas o custo total do caminho menos perigoso. Para a entrada acima, a saída correspondente seria, simplesmente:

```
12
```

5 Arquivos providos

Para simplificar a implementação, as especificações citadas já foram implementadas em arquivos que podem servir como modelo para seu código. Baixe o código em C ou o código em C++ de acordo com sua escolha de linguagem. O uso dos modelos não é obrigatório, porém, eles ajudam a evitar dificuldades com o formato de entrada e saída, já que discrepâncias em relação a esses formatos serão penalizadas na avaliação.

6 Experimentos

Após a finalização do seu código, a implementação deve ser avaliada através de uma série de experimentos. Antes de iniciá-los, lembre-se de compilar o código com a opção `-O3` para ativar as rotinas de otimização do compilador! Utilize um dos seguintes comandos para compilação:

```
gcc -std=c99 -O3 -pedantic -Wall escalada.c -o escalada
```

```
g++ -std=c++11 -O3 -pedantic -Wall escalada.cpp -o escalada
```

7 Testes

Existe um conjunto de testes abertos (arquivos de entrada + saída) que está disponível para uso durante a implementação. Além disso, existe um conjunto de testes fechados, aos quais você não terá acesso.

O programa será submetido aos testes abertos dentro do SuSy, enquanto que o conjunto fechado será executado apenas durante a correção, fora do Susy e após o prazo final de entrega.

Os testes abertos podem ser usados de forma automatizada via SuSy, ou podem ser usados de maneira manual, baixando o arquivo `testes.zip` dos arquivos auxiliares, descompactando-o, e executando o script dentro da pasta `testes`, onde também deve ser colocado o seu código. O comando de execução deve ser:

```
./run_tests.sh
```

8 Relatório

Crie um arquivo `pdf` para o seu relatório, chamado '`PA#NNNNNN.pdf`', onde `#` é o número do presente PA e `NNNNNN` é o seu R.A. O seu relatório deve necessariamente conter os seguintes itens:

1. Prova de que o problema a ser resolvido tem subestrutura ótima, deduzindo uma fórmula de recorrência que poderá ser usada para projetar seu algoritmo. Explique o que é cada termo da recorrência. Indique os casos base e seus valores.
2. Depois (e só depois de fazer o item (1.)), descreva em alto nível (ou em pseudo-código) o algoritmo mais eficiente que você conseguiu projetar para solução do problema e justifique sua escolha, mencionando outras possibilidades que você considerou! Lembre-se de utilizar e explicitar no projeto de seu algoritmo quais técnicas vistas na disciplina são utilizadas para computação das respostas esperadas.
3. Em seguida, faça uma análise de complexidade de tempo e de espaço (em função de n) de seu algoritmo.

9 Critérios de avaliação

A avaliação apresentará os seguintes critérios:

- Programas que não compilam receberão nota zero.
- Programas que implementam um algoritmo que **não corresponde** a um projeto fundamentado nos itens apresentados no Relatório (para efetivamente computar uma resposta ótima) receberão nota zero.
- Programas que implementam algoritmos corretos com complexidade $O(n^2)$: poderão receber até $1,0$ ponto.
- Programas que implementam algoritmos corretos com complexidade $\omega(n^2)$: poderão receber até $0,5$ ponto.
- Programas que resolvem corretamente apenas algumas instâncias serão analisados levando-se em conta tanto o algoritmo implementado quanto o número de instâncias efetivamente resolvidas.

Além disso, será levado em conta a qualidade do conteúdo e completude do Relatório submetido.

10 Entrega

Observe os seguintes quesitos para realizar a entrega de seu trabalho:

- A submissão do arquivo do Relatório deve ser feita via Google Classroom na Tarefa correspondente a esta Atividade. Não se esqueça de efetivar o envio do seu Relatório após fazer o seu *upload*, ou seu Relatório será desconsiderado.
- A sua submissão do arquivo de código será feita **exclusivamente** através do sistema SuSy. Para acessá-lo, basta utilizar seu R.A. e sua senha da DAC (Ex: `usuário: 123456` e `senha: senha-da-DAC`).
- O arquivo com o seu código deve se chamar '`escalada.c`' ou '`escalada.cpp`', de acordo com a linguagem usada. O SuSy não irá alertá-lo sobre erros nos nomes, portanto, tenha atenção na hora de fazer suas submissões.
- No SuSy, serão aceitas **até 5 submissões** por aluno, sendo preservada apenas a última submissão.
- O prazo para submissão das resoluções se encerrará às 23hs do dia indicado no Google Classroom. Envios realizados (do programa no SuSy ou do Relatório do Google Classroom) após esse horário serão considerados **atrasados**. Se o atraso for de **até duas horas** após o encerramento do prazo regular de submissão, as resoluções submetidas ainda serão corrigidas e receberão nota integral. Resoluções enviadas com **mais de 2hs** de atraso **não serão corrigidas** e receberão nota **zero**.

Link para submissão no SuSy: https://susy.ic.unicamp.br:9999/mc458a/PA2.

AVISO: as ferramentas de detecção de plágio do SuSy serão utilizadas durante a avaliação das submissões!