Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2012

Trabalho Prático 2 - Paradigmas

Mussum sofre de um distúrbio raro chamado Aibofobia, que é uma fobia de palíndromos. Um palídromo é uma palavra que é lida da mesma forma da esquerda para direita ou da direita para esquerda. Ou seja, o reverso da palavra é igual à palavra original.

O psicólogo de Mussum propôs um tratamento que consiste em apresentar a Mussum palíndromos derivados de palavras que remetem boas recordações a ele. Dessa forma, para cada palavra que Mussum fornece aos médicos eles retornam um palíndromo gerado inserindo caracteres nessa palavra. Pouco educados nas ciências gramaticais e menos ainda nas ciências computacionais, os médicos de Mussum adotaram um esquema simples para gerar os palíndromos. Para cada palavra que eles recebiam eles retornavam a palavra concatenada com o reverso da palavra. Ou seja, se a palavra CASA fosse dada por Mussum, os médicos retornavam CASAASAC.

O tratamento foi um sucesso! Hoje fazem 17 meses que Mussum não tem nenhuma crise. Motivados pelo sucesso do tratamento os médicos querem ajuda para gerar os palíndromos de forma mais eficiente. Quanto menor o palíndromo mais ele irá remeter o paciente à palavra original. Eles desejam, portanto, saber qual é o menor palíndromo possível de ser gerado inserindo caracteres em uma dada palavra. No caso da palavra CASA, por exemplo, seria CASAC. Seu trabalho consiste em encontrar tal palíndromo mínimo dada uma palavra de entrada.

Paradigmas

O problema deve ser solucionado utilizando diferentes paradigmas de programação. Uma solução ótima deve ser dada utilizando programação dinâmica (PD) e uma solução (não necessariamente ótima) deve ser dada utilizando o paradigma guloso. Para a solução em PD discuta as propriedades de sub-estrutura ótima e sobreposição do sub-problemas. Para a solução gulosa discuta a otimalidade da solução. Tente estimar o fator de aproximação da sua solução empiricamente. Ou seja, para vários casos de entrada calcule quantos porcento a mais de caracteres a solução gulosa utiliza em relação à solução ótima e discuta se existe um teto observável para esse valor.

Entrada e Saída

A entrada deve ser lida de um arquivo passado como argumento e a saída deve ser escrita também em um arquivo passado como argumento. Um exemplo de invocação é dado abaixo:

```
./tp1 input.txt output.txt
```

O arquivo de entrada consiste basicamente de k palavras. A primeira linha possui o valor de k e as k linhas seguintes possuem as palavras de entrada. Considere que cada palavra possui no máximo 5000 caracteres.

A saída consiste de k linhas com o número mínimo de caracteres necessários para gerar um palíndromo com a palavra dada de entrada na linha correspondente. Um exemplo é mostrado abaixo.

Entrada:	Saída:
5	1
BOLO	1
XBOX	1
BANANA	4
SUCESSO	0
WOW	

Entrega

- A data de entrega desse trabalho é **8 de Novembro**. Será fornecido **um dia adicional** para a entrega da **documentação impressa** na secretaria, entretanto a cópia impressa deve ser idêntica à versão digital.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$ onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- **Apenas** os arquivos abaixos devem estar presentes no arquivo zip. Não inclua arquivos compilados ou gerados por IDEs.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua nenhuma pasta. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- **Dois arquivos executáveis devem ser gerados para esse trabalho.** Um executável chamado `tp2g` rodando a solução gulosa e um executável chamado `tp2pd` rodando a solução em programação dinâmica.
- Trabalhos que não seguirem as instruções acima serão penalizados
- Uma cópia impressa da **documentação deve ser entregue na secretaria** (atenção para o horário de funcionamento). Não utilize os escaninhos dos professores. Entregue diretamente para a secretária.
- A documentação na secretaria e o envio pelo *minha.ufmg* receberão **penalização de atraso independentes**, em seguida as notas serão combinadas como descrito no próximo item. Essa cópia

- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.
- Uma planilha será divulgada no *minha.ufmg* instruindo o **agendamento das entrevistas.**

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- Variavéis globais devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanta **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- **Legibilidade e boas práticas** de programação serão avaliadas.