Universidade Federal de Pernambuco
Centro de Informática

IF767 - Processamento de Cadeias de Caracteres 2015.2
Prof. Paulo Fonseca

Projeto 1 — versão 1 (01/10/2015)

- Este documento contém as regras e diretrizes para o primeiro projeto. Leia com atenção todo o conteúdo do documento e tente ater-se às orientações o mais fielmente possível.
- As regras abaixo podem ser modificadas a qualquer tempo pelo professor no melhor interesse acadêmico e didático. As modificações serão comunicadas em tempo útil através do grupo de discussão da disciplina.
- Eventuais omissões serão tratadas de maneira discricionária pelo professor, levando-se em conta o bom senso, a praxe acadêmica e os interesses didáticos.

Objetivo

Neste projeto deve ser desenvolvida uma ferramenta para busca de padrões num arquivo ou conjunto de arquivos, chamada **`pmt`**, similar ao GNU grep [1]. O objetivo é de consolidar o conhecimento dos algoritmos vistos no curso através da implementação de um software com correção, documentação e escalabilidade em nível de produção.

A ferramente deve ser avaliada em termos do desempenho absoluto e relativamente a outros algoritmos/ferramentas, utilizadas como benchmark, sobre diversos tipos de dados.

Equipes

O projeto deve ser feito em equipes de 2 ± 1 integrantes (idealmente, 2). Cada integrante é suposto participar e conhecer em detalhes todas as atividades envolvidas (implementação, documentação e testes).

Data de entrega

O trabalho deve ser entregue por e-mail até **25 de Outubro de 2015** (veja a Seção *Deliverables*).

Funcionamento básico

A ferramenta deve ter uma interface em linha de comando (*command line interface—CLI*) seguindo as diretrizes GNU/POSIX [2]. A sintaxe básica deve ser

```
$ pmt [options] pattern textfile [textfile...]
```

que fará com que o padrão `pattern` seja procurado no(s) arquivo(s) `textfile`. Múltiplos arquivos de texto podem ser indicados utilizando-se *wildcards* (e.g. `livro*.txt`).

[1] `http://www.gnu.org/software/grep/`
[2] `https://www.gnu.org/prep/standards/html_node/Command_002dLine-Interfaces.html`

A ferramenta deve suportar dois modos:

- Busca exata (default)
- Busca aproximada.

A busca aproximada deve ser indicada pela opção:

`-e`, `--edit` e_{max} **:** Localiza todas as ocorrências aproximadas do padrão a uma distância de edição máxima e_{max}

A ferramenta também poderá receber um conjunto de padrões a serem procurados num arquivo, sendo um padrão por linha, o que deve ser feito através da opção

`-p`, `--pattern` ***`patternfile`*****`:`** Realiza a busca de todos os padrões contidos no arquivo *patternfile*.

Implementação

A ferramenta deve ser implementada preferencialmente em C/C++. O objetivo é torná-la a mais eficiente possível. A ferramenta deve ser baseda na plataforma GNU/Linux. Deve-se tentar minimizar as depedências externas para torná-la facilmente portável entre plataformas.

Podem ser utilizadas APIs externas apenas para o *frontend* da ferramenta. Por exemplo, a GNU C Library (glibc)[3] contém funções para o parsing das opções de linha de comando (getopt), e para os nomes de arquivos com wildcards (fnmatch). Entretanto, o *backend* da ferramenta deve consistir *apenas de algoritmos vistos em aula* e (re-)implementados diretamente pelos alunos. *A detecção de cópia de partes substanciais do código desses algoritmos implicará na atribuição da nota 0.0 (zero) ao trabalho como um todo, independente de outras partes*.

Deverão ser implementados *pelo menos* dois algoritmos: um para a busca exata e outro para a busca aproximada.

Testes/Experimentos

Devem ser realizados experimentos para aferir o desempenho prático da ferramenta em termos de tempo/espaço. Para isso deve ser compilado um conjunto de dados de teste composto da textos de diferentes fontes e origem. Como ponto de partida (chegada?) podem ser utilizados os corpora disponíveis em

1. Pizza&Chili (`http://pizzachili.dcc.uchile.cl/texts.html`)
2. SMART (`http://www.dmi.unict.it/~faro/smart/download.php`)

Os resultados dos experimentos para diversas configurações texto/padrão devem ser organizados em tabelas e gráficos. Para além dos simples dados brutos, deve-se tentar caracterizar um padrão de desempenho dos algoritmos em função dos parâmetros e características das entradas que nos permitam, eventualmente, prever o comportamento em cenários não testados diretamente. Ferramentas padrão como o `grep` ou o `agrep`, bem como outros algoritmos e ferramentas disponíveis através da literatura e de software de terceiros podem/devem ser utilizados como benchmark para comparação.

[3] `http://www.gnu.org/software/libc/`

Deliverables

Deve ser entregue um arquivo comprimido em formato `.tgz` ou `.zip`. Para facilitar a identificação nomeie o arquivo no formato

login-versão.tgz

onde ***login*** corresponde ao primeiro username em ordem lexicográfica da equipe e ***versão*** corresponde a um número sequencial (1,2,3,...) indicativo da versão submetida[4]. Esse arquivo comprimido deve consistir de um diretório com o seguinte conteúdo *mínimo*.

```
pmt/
 |
 +-- doc/
 +-- src/
 +-- README.txt
```

O arquivo `README.txt` deve conter uma identificação da ferramenta, dos autores, e as instruções para compilação (vide seção abaixo). O conteúdo de cada diretório será especificado a seguir.

Código-fonte

Deve ser entegue o código fonte da ferramenta juntamente com um Makefile ou script para compilação no subdiretório `src/`. As instruções para o processo de compilação da ferramenta devem ser dadas no arquivo `README.txt`. Idealmente a compilação deveria consistir apenas na execução de um simples `make`.

O código deve ser o mais *limpo*[5] possível. Entretanto, os objetivos principais são 1) correção e 2) eficiência. Portanto, deve-se evitar o uso exagerado de modelagem por objetos, padrões de projetos, etc. que tornem o programa mais lento. Um programa bem estruturado, com nomes expressivos para funções e variáveis, e com uma separação clara entre interface e motor de busca, deve ser suficiente.

Após a compilação, o arquivo executável deve estar num diretório `bin`, criado dentro do diretório original, isto é, teremos

```
pmt/
 |
 +-- bin/    <=== executável aqui
 +-- doc/
(...)
```

Documentação

Conforme as diretrizes adotadas para a CLI, uma ajuda com as instruções para a utilização básica da ferramenta deve ser obtida através da execução da ferramenta com a opção

[4] É comum que sejam submetidas mais de uma versão, devido a correções de última hora. Nesse caso, apenas a última versão é considerada para avaliação

[5] *RC Martin*. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

`-h, --help`

Além disso, deverá ser entregue um breve relatório dividido em três principais seções:

1. Identificação
   - Identificação da equipe
   - Breve descrição da contribuição de cada membro da equipe ao trabalho
2. Implementação
   - Descrição do funcionamento da ferramenta, incuindo:
     - Algoritmos implementados
     - Situações nas quais cada algoritmo é empregado
   - Detalhes de implementação relevantes, com impacto significativo para o desempenho da ferramenta, incluindo:
     - Estruturas de dados
     - Estratégia de leitura das entradas
     - Heurísticas para combinação do algoritmos
     - etc.
   - Bugs conhecidos e limitações de desempenho notáveis. Se o trabalho não foi integralmente concluído, o que faltou deve ser explicitamente reportado aqui.
3. Testes e Resultados
   - Descrição dos dados e ferramentas de comparação utilizados
   - Descrição do ambiente de testes
   - Descrição dos experimentos realizados
   - Dados e resultados obtidos (tabelas, gráficos, ...)
   - Discussão dos resultados e conclusão

   Dados experimentais brutos muito detalhados e volumosos podem ser submetidos como anexos em arquivos separados. No relatório, deve-se buscar expor dados compilados que favoreçam a visualização e interpretação.

Esse relatório deve estar contido no subdiretório `doc/`, num arquivo `.pdf` (*Não* use MSWord ou qualquer formato proprietário).

Data sets

Os dados utilizados nos testes **NÃO** devem ser submetidos junto com o trabalho em nenhuma hipótese. A inclusão de arquivos de dados será penalizada. Caso seja considerado necessário, deve-se torná-los disponíveis online e indicar o endereço na seção da descrição dos testes do relatório.

Avaliação

A avaliação será feita com base nos seguintes critérios:

1. Implementação (peso 6). Inclui a correção, eficiência e qualidade do código-fonte levando-se em conta a quantidade e dificuldade intrínseca dos algoritmos implementados.
2. Testes (peso 3). Inclui a reprodutiblidade dos experimentos, a abrangência dos dados, a organização e apresentação dos resultados, a correção e profundidade das análises e a exposição das conclusões.
3. Qualidade da documentação (peso 1). Inclui o aspecto geral do relatório, o `README.txt` e a ajuda do programa.

Arguição

A avaliação será feita mediante análise do material submetido e de uma arguição a ser agendada, posteriormente, com cada equipe. Cada integrante deve ter participado de todas as atividades e, portanto, deve conhecer integralmente ser capaz de responder questões sobre qualquer aspecto do projeto.

Extras

Além desse conjunto mínimo de requisitos, cada equipe está livre para implementar recursos extras. Esses recursos devem ser assinalados no relatório e poderão receber alguma bonificação.