Concepção e Análise de Algoritmos

LeGrep

Turma 2 - Grupo C

José Peixoto	200603103	ei12134@fe.up.pt
Pedro Moura	201306843	up201306843@fe.up.pt

30 de Maio de 2015

Conteúdo

Resumo

No âmbito da unidade curricular de Concepção e Análise de Algoritmos, foi-nos proposto o desenvolvimento de uma aplicação que emulasse as funcionalidades do utilitário da linha de comandos `grep` (**g**lobally search a **r**egular **e**xpression and **p**rint), explorando o conhecimento acerca de algoritmos para pesquisa exacta de strings.

1 Formalização do problema

1.1 Dados de entrada

- $O = \{o_i, ..., o_k\}$ - Conjunto de comandos opcionais para o cálculo e mostra de resultados.
- P - Padrão a procurar.
- F - Ficheiro onde se pretende procurar o padrão.

Lista com os comandos opcionais de entrada

$-n$, uso do algoritmo *naive* na pesquisa exacta

$-m$, uso do algoritmo baseado em autómato finito

$-k$, uso do algoritmo de Knuth-Morris-Pratt (pré-definido)

$-i$, ignorar diferenças de capitalização entre caracteres

$-v$, mostrar linhas sem correspondências

$-h$, mostrar manual de ajuda

$-BN$, imprimir N linhas de contexto anterior

$-AN$, imprimir N linhas de contexto posterior

$-CN$, imprimir N linhas de contexto anterior e N linhas de contexto posterior

1.2 Limites de aplicação

Na procura de um dado padrão, os dados de entrada têm de ser introduzidos obrigatoriamente com a seguinte ordem:

```
[./]legrep[.exe] [Opções] Padrão Ficheiro
```

1.3 Resultados esperados

Os dados de saída variam consoante o padrão, ficheiro e opções seleccionadas pelo utilizador.

L_i, linha de texto

l_i, linha de texto com todos os caracteres em minúsculas

p, padrão com todos os caracteres em minúsculas

$$Output = \begin{cases} \{L_i, \ldots, L_k\},\ L_i \in F \wedge P \subseteq L_i & \text{if } O = \emptyset, \\ \{L_i, \ldots, L_k\},\ L_i \in F \wedge P \not\subset L_i & \text{if “-v” } \subseteq O, \\ \{l_i, \ldots, l_k\},\ l_i \in F \wedge p \subseteq l_i & \text{if “-i” } \subseteq O, \\ \{L_i, \ldots, L_k\},\ (L_i \in F \wedge P \subseteq l_i) \ \vee & \\ \vee\ (L_j \in F \wedge j < i \wedge j \geq i - N) & \text{if “-B N” } \subseteq O, \\ \{L_i, \ldots, L_k\},\ (L_i \in F \wedge P \subseteq l_i) \ \vee & \\ \vee\ (L_j \in F \wedge j \leq i + N \wedge j > i) & \text{if “-A N” } \subseteq O, \\ \{L_i, \ldots, L_k\},\ (L_i \in F \wedge P \subseteq l_i) \ \vee & \\ \vee\ (L_j \in F \wedge j < i \wedge j \geq i - N) \ \vee & \\ \vee\ (L_j \in F \wedge j \leq i + N \wedge j > i) & \text{if “-C N” } \subseteq O. \end{cases}$$

Os resultados esperados cuja cardinalidade $|O| > 1$ envolvem a conjunção dos casos atrás listados, desde que as opções adicionais pertençam a esse conjunto e sejam opções válidas para o programa.

2 Descrição da solução

É feita uma procura de correspondência de um padrão numa cadeia de caracteres “string de texto” a cada linha lida de um dado ficheiro de texto.

2.1 Algoritmos de pesquisa exacta em strings

Na pesquisa de uma correspondência exacta de um dado padrão P num texto T, recorrem-se a três algoritmos diferentes, escolhidos de forma opcional pelo utilizador:

2.1.1 Naive

Para um padrão de tamanho m ou $|P|$ e texto de tamanho n ou $|T|$, encontra todos os deslocamentos s válidos, verificando desde o início do padrão a condição $P[1..m] = T[s+1..m]$ para cada um dos $n-m+1$ valores possíveis de s, quando $|P| > 0$. Este procedimento implica, no pior dos casos, uma complexidade de $O(|P|.|T|)$. Torna-se bastante ineficiente para padrões com tamanho grande, uma vez que se faz uma verificação de equivalência para o tamanho total do padrão a cada novo deslocamento.

2.1.2 Baseado em autómato finito

Cada carácter do texto é verificado apenas uma vez. A análise do texto é feita em tempo constante $|T|$, porque se recorre à modelação de um autómato finito com um conjunto de estados Q de tamanho $|P|$. A função de transição δ possibilita a atribuição de novos estados q a cada novo carácter a lido do texto.

$$\delta(q, a) = \sigma(P_q a) \tag{1}$$

No ciclo principal verifica-se a cada novo carácter lido do texto se se atingiu o estado de aceitação e por conseguinte, uma correspondência exacta do padrão no texto.

A construção do autómato é feito numa primeira fase de pré-processamento e envolve a determinação do alfabeto Σ do padrão. Na determinação da função de transição recorre-se a uma função auxiliar para o sufixo $\sigma(x)$ que retorna, para um dado conjunto de caracteres do alfabeto, Σ^* o maior comprimento (estado) que seja prefixo do padrão.

$$\sigma(x) = max\{k : P_k \sqsupset x\} \tag{2}$$

2.1.3 Knuth-Morris-Pratt

Cada carácter do texto é verificado apenas uma vez. A análise do texto é feita em tempo constante $|T|$, recorrendo à função auxiliar π, vector calculado numa fase de pré-processamento. O vector π contém informação acerca dos deslocamentos do padrão, evitando a comparação de todos os caracteres em todas as situações. P_q é o comprimento do maior prefixo de P que é também sufixo de P_q:

$$\pi[q] = max\{k \ : \ k < q \wedge P_k \sqsupset P_q\} \tag{3}$$

2.1.4 Complexidade empírica

Na análise da complexidade de forma empírica, fez-se uma pesquisa 5 vezes sobre a mesma string de texto e aumentou-se sucessivamente o tamanho do

texto por um factor de 5.

Tempo em segundos da execução do algoritmo *naive*:

Tamanho do texto	Tempo total	Média do tempo por iteração
568	0.000446	8.92e-05
2840	0.002769	0.0005538
14200	0.01356	0.002712
71000	0.06682	0.013364
355000	0.342743	0.0685486
1775000	1.71114	0.342227
8875000	8.31921	1.66384

Tempo em segundos da execução do algoritmo baseado num autómato finito:

Tamanho do texto	Tempo total	Média do tempo por iteração
568	0.000173	3.46e-05
2840	0.001024	0.0002048
14200	0.004816	0.0009632
71000	0.02306	0.004612
355000	0.117364	0.0234728
1775000	0.573843	0.114769
8875000	2.82262	0.564524

Tempo em segundos da execução do algoritmo Knuth-Morris-Pratt:

Tamanho do texto	Tempo total	Média do tempo por iteração
568	0.000103	2.06e-05
2840	0.000595	0.000119
14200	0.002732	0.0005464
71000	0.012749	0.0025498
355000	0.066263	0.0132526
1775000	0.32453	0.064906
8875000	1.56998	0.313995

Pela análise das tabelas chega-se à conclusão expectável de que o algoritmo menos eficiente é o *naive* e que a fase de pré-processamento do algoritmo baseado num autómato finito, com maior complexidade temporal e espacial, é suficiente para se traduzir em tempos mais lentos de execução comparativamente ao algoritmo de Knuth-Morris-Pratt.

3 Lista de casos de utilização

- Pesquisa das linhas de um dado ficheiro onde há pelo menos uma ocorrência do padrão.
- Pesquisa inversa das linhas de um dado ficheiro onde há pelo menos uma ocorrência do padrão.
- Mostra do contexto com linhas que antecedem e/ou precedem as linhas onde ocorrem as correspondências.
- Modo de correspondência que despreza a capitalização dos caracteres.
- Selecção do algoritmo usado na procura de correspondências exactas.
- Mostra de texto de ajuda na utilização do programa com lista de comandos opcionais.
- Multi-plataforma, testado em sistemas Linux e Windows.

4 Relato das principais dificuldades encontradas no desenvolvimento do trabalho

Sentiram-se dificuldades, em casos específicos, na detecção e mostra correcta da mudança de contexto, como acontece no `grep`. A sinalização da mudança do contexto, implica a selecção de pelo menos uma das opções de mostra de linhas de contexto (anterior ou posterior).

Houve também dificuldades na criação da tabela com as transições de estados, mais precisamente na escolha de uma estrutura de dados que permitisse guardar uma transição, a partir de um dado estado e de um carácter do alfabeto, possibilitando também tempo constante de acesso. Embora inicialmente se tenha optado por guardar estes dados numa tabela de dispersão, mesmo experimentando com diferentes funções de dispersão, a performance era bastante pior, pelo que se acabou por recorrer a um vector para guardar os dados. O tempo de acesso a este vector foi optimizado para ser no pior dos casos o tamanho do alfabeto $|\Sigma|$.

5 Indicação do esforço dedicado por cada elemento do grupo

5.1 José Peixoto

- Tratamento dos dados de entrada.
- Mostra dos comandos de ajuda.

- Concepção dos algoritmos "naive" e baseado em autómato finito.
- Autor do relatório.

5.2 Pedro Moura

- Concepção do algoritmo Knuth-Morris-Pratt.
- Correcção de vários erros no código.
- Melhoria da performance no tratamento das opções.

Referências

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to algorithms Third Edition*, Cambridge, MA [etc.] : The MIT Press, cop. 2009.

N.B. Este relatório não foi escrito ao abrigo do novo Acordo Ortográfico.