Análise de Algoritmos

Segunda Lista de Exercícios

Reconhecimento de padrões em palavras e programação dinâmica

30 de maio de 2008

Prof. Mauricio Ayala-Rincón

Entrega: 14 de maio de 2008 (100%) - Grupos de cinco pessõas
Será disponibilizado um gabarito da lista, depois do dia 14 de maio
Listas entregues entre 15 e 19 de maio de 2008: 50%
Listas entregues após 19 de maio de 2008: 0%

Estágiario de docência: Daniel Lima Ventura: `ventura at mat dot unb dot br`

Reconhecimento de Padrões em Palavras

1. Demonstre que a computação da *função de erro* ou *falha*, denotada por f, para o algoritmo de Knuth-Morris-Pratt (KMP) fornecida na aula é correta. I. e., que o F computado coincide com a f definida. Lembre-se de que a definição dada e o algoritmo de cálculo sugerido desta, são objetos diferentes e por isso não necessáriamente coincidentes.

   Sejam P, Q palavras sobre uma alfabeto Σ. Se $|P| = m$, denotamos os seus símbolos como $P = p_1 p_2 \dots p_m$. P_k, para $k = 0..m$, denota o prefixo k-ésimo de P. Assim, $P_0 = \lambda$, a palavra vazia, e $P_k = p_1 p_2 \dots p_k$, para qualquer $k = 1..m$. Usa-se a notação $Q \sqsubset P$ e $Q \sqsupset P$ para denotar que Q é prefixo de P e que Q é sufixo de P, respectivamente.

   Definição(Função de erro ou falha de KMP) Seja P uma palavra sobre um alfabeto Σ com $|P| = m$. A função de erro de Knuth-Morris-Pratt para P, $f : \{1, \dots m\} \to \{0, \dots m\}$, define-se da seguinte maneira

$$f(i) = \begin{cases} 0 & \text{se } i = 1 \\ \text{máximo}(\{|Q| + 1 \mid Q \sqsupset P_{i-1} \text{ e } Q \sqsubset P_{i-1} \text{ e } p_i \neq p_{|Q|+1}\} \cup \{0\}) & \text{se } i > 0 \end{cases}$$

   Observe que dados um P com $|P| = m$ e algum $i = 1..m$, se não existe Q tal que $Q \sqsupset P_{i-1}$ e $Q \sqsubset P_{i-1}$ e $p_i \neq p_{|Q|+1}$, então o primeiro conjunto na segunda linha da definição de f é considerado vazio e $f(i)$ é definido como 0.

Algoritmo(Cálculo da função de erro ou falha de KMP)

```
Entrada:    PadrAo P com m sImbolos
SaIda: FunCAo de erro ou falha de KMP F para P
 i := 1, j := 0;
 F[i] := 0;
 while (i<m) do
    while (j>0) e (P[i] != P[j])
       j := F[j];
    i++;
    j++;
    if (P[i] = P[j]) then F[i] := F[j];
    else F[i] := j;
```

2. Forneça a função de erro de KMP para os seguintes padrões: AAAB, ABABBABABBAB, ABABBABABBABBABABBAB.

3. Explique como modificar a função de erro de KMP para tratar a busca de todas as ocorrências de um padrão num texto.

4. Forneça a função *matchjump* do algoritmo de Boyer-Moore (BM) para os seguintes padrões: AAAB, ABABBABABBAB, ABABBABABBABBABABBAB.

5. Explique como modificar a função *matchjump* de BM para tratar a busca de todas as ocorrências de um padrão num texto.

6. Opcional. Implemente na linguagem C o algoritmo de Boyer-Moore para reconhecimento de padrões em palavras.

   Compare o comportamento do algoritmo de Knuth-Morris-Pratt (uma implementação deste encontra-se na página `http://www.mat.unb.br/~ayala/academics.html` no arquivo "match_KMP.c" na seção desta disciplina) e sua implementação do algoritmo de Boyer-Moore com palavras em alfabetos binário e inglês.

   Apresentar:

   - Especificação do problema e explicação do método de solução.
   - Descrição e análise da complexidade dos algoritmos implementados.
   - Definição e justificação das estruturas de dados usadas na implementação.
   - Descrição da implementação no mesmo listado.
   - Referências.

7. Explique quando é mais apropriado o uso do algoritmo de KMP e quando o do algoritmo de BM no reconhecimento de padrões. Explique as ventagens do método de reconhecimento de padrões baseado nas árvores de sufixos.

Programação Dinâmica

8. Multiplicação ótima de sequências de matrizes.

   (a) Demonstre que o número de operações aritméticas necessárias para calcular diretamente a relação de recurrência

   $$M(i,j) = min_{i \leq k \leq j-1}[M(i,k) + M(k+1,j) + d_{i-1}d_k d_j], \text{ para } 1 \leq i < j \leq n,$$
   $$M(i,i) = 0,$$

   resultante do tratamento puramente recursivo ao problema da fatoração ótima de multiplicação de sequências de matrizes, é de ordem exponencial; i. e., tem um límite inferior exponencial em n.

   (b) Explique passo a passo o funcionamento do algoritmo dinâmico para ordem ótima de multiplicação de sequências de matrizes, para matrizes A_1, A_2, A_3, A_4 de dimensões

   $$20 \times 2,\ 2 \times 15,\ 15 \times 40,\ 40 \times 4,$$

   respectivamente.

   (c) Sejam $A_1, A_2, \ldots, A_n$ matrizes de dimensões $d_0 \times d_1, d_1 \times d_2, \ldots, d_{n-1} \times d_n$, respectivamente. Análise a seguinte técnica voraz para o problema de ordem ótima de multiplicação de sequências de matrizes:

   em cada passo, selecione a dimensão restante maior (entre $d_1, \ldots, d_{n-1}$) e multiplique as duas matrizes adjacentes eliminando tal dimensão.

   - Qual é a ordem do algoritmo correspondente a tal técnica?
   - Encontre um exemplo para o qual tal técnica voraz não funciona.
     Ajuda: aplique a técnica sugerida ao exemplo anterior.

   (d) Na página `http://www.mat.unb.br/~ayala/academics.html` no arquivo "matmultseq.c" na seção desta disciplina, encontra-se uma implementação na linguagem `C` deste método. Inclua uma função para imprimir simbolicamente a sequência ótima de multiplicação.

9. Na página `http://www.mat.unb.br/~ayala/academics.html`, na seção desta disciplina no arquivo "arvore_binaria_otima_dinamica.c", encontra-se uma implementação na linguagem `C` da solução dinâmica do problema de construção de árvores binárias de consulta ótimas para chaves com probabilidade de busca. Utilize a bibliografia da disciplina para formular o problema e a solução dinâmica deste.

10. Sejam $u = u_1 \ldots u_n$ e $v = v_1 \ldots v_m$ palavras no alfabeto $\{0, 1\}$. u é uma *subsequência* de v se existem $1 \leq s_1 < s_2 < \cdots < s_n \leq m$, tais que $u_i = v_{s_i}$, $1 \leq i \leq n$.
    Dadas duas sequências x e y, u é uma *subsequência comum* de x e y, se u é uma subsequência de x e também de y.
    Seja:

$$maxcom(x, y) = max\{\ |u|\ \mid\ u \text{ é uma subsequência comum de } x \text{ e } y\}$$

(a) Sejam $x, y \in \{0,1\}^*$, $\delta, \sigma \in \{0,1\}$.
Descreva $maxcom(x\delta, y\sigma)$ em termos de

$maxcom(x\delta, y)$,
$maxcom(x, y\sigma)$ e
$maxcom(x, y)$.

(b) Desenhe um algoritmo $O(n^2)$ para calcular $maxcom(x, y)$, onde $|x| = |y| = n$.
Ajuda: use técnicas de programação dinâmica.

(c) Modifique seu algoritmo de tal forma que no cálculo de $maxcom(x, y)$ sejam geradas as subsequências correspondientes de x e y.

11. O *reconhecimento aproximado de padrões em palavras* é uma generalização do problema de reconhecimento de padrões em palavras. O problema pode-se estabelecer da seguinte maneira:

> dadas palavras **padrão** e **texto**, p e t, respectivamente, sobre um alfabeto Σ, com $|p| = m > 0$ e $|t| = n > 0$ e dado um inteiro $k \geq 0$, encontrar subpalavras s de t tais que a **distância de edição** entre s e p, $d(s, p)$, é menor ou igual que k.

A distância de edição entre duas palavras x e y é o custo total mínimo possível de uma sequência de *passos de edição* para converter x em y. Denotando com ϵ a palavra vazia, em cada passo de edição aplica-se uma *regra de reescrita* da forma:

- $a \to \epsilon$, $a \in \Sigma$ (eliminação);
- $\epsilon \to a$, $a \in \Sigma$ (inserção);
- $a \to b$, $a, b \in \Sigma$ (substituição).

Diz-se que a palavra x é uma *k-aproximação* de y, se se pode converter x em y em k passos de edição. Por exemplo, existe uma ocorrência 1-aproximada do padrão *abb* na primeira posição do texto *abcdbdb* e uma ocorrência 2-aproximada na quarta posição, como se ilustra no seguinte esquema.

$$\begin{array}{cccccccc} a & b & b & & a & b & \epsilon & b \\ & & \downarrow & & \downarrow & & \downarrow & \\ a & b & \epsilon & c & d & b & d & b \end{array}$$

O mecanismo de solução mais "popular" do problema de reconhecimento aproximado de padrões é baseado em técnicas de *programação dinâmica* e tem complexidade $O(nm)$. Sejam $p = p_1 \ldots p_m$ e $t = t_1 \ldots t_n$ um padrão e um texto, respectivamente. Basicamente o método consiste em construir uma $m + 1 \times n + 1$-tabela D tal que para $0 \leq i \leq m$,

$0 \leq j \leq n$, $D[i,j]$ é a mínima distância de edição de $p_1 \ldots p_i$ à subpalavra do texto que termina em t_j. Existem aproximações do padrão com distância de edição $\leq k$ terminando em t_j se e solamente se $D[m,j] \leq k$. A tabela se constroi segundo as seguintes regras:

- $D[0,j] = 0, \;\; 0 \leq j \leq n$;
- $D[i,0] = i, \;\; 0 \leq i \leq m$;
- $D[i,j] =$ mínimo entre:
  - $D[i-1,j] + 1$,
  - $D[i-1,j-1] +$ se $p_i = t_j$ então 0 se não 1,
  - $D[i,j-1] + 1$,

  $1 \leq j \leq n, 1 \leq i \leq m$.

Note que $D[i,j]$ depende unicamente de $D[i-1,j], D[i-1,j-1]$ e $D[i,j-1]$ o que permite construir a tabela coluna por coluna e fila por fila em ordem ascendente.
Considere, por exemplo, a tabela construida para $p = cab$, $t = abcabdab$:

		a	b	c	a	b	d	a	b
	0	0	0	0	0	0	0	0	0
c	1	1	1	0	1	1	1	1	1
a	2	1	2	1	0	1	2	1	2
b	3	2	1	2	1	0	1	2	1

(a) Descreva em detalhe o algoritmo sugerido usando técnicas de programação dinâmica.

(b) Indique com precisão como pode ser usada a tabela gerada pelo anterior algoritmo para produzir as "aproximações do padrão" achadas.

12. Considere o problema de *soma de subconjuntos*: dado um conjunto de inteiros positivos $S = \{s_1, \ldots, s_n\}$ e um inteiro positivo C, determinar se existe um subconjunto S' de S tal que

$$\sum_{s_i \in S'} s_i = C$$

Uma solução dinâmica do problema consiste em construir uma tabela M de tamanho $(n+1) \times (C+1)$, tal que $M[i,j] = 1$ se existe um subconjunto de $\{s_1, \ldots, s_i\}$ com soma j e se não existe $M[i,j] = 0$.
Note que a tabela pode ser construida coluna por coluna ascendentemente usando o fato de que um subconjunto de $\{s_1, \ldots, s_i\}$ soma j sse

- algum subconjunto S' de $\{s_1, \ldots, s_{i-1}\}$ soma j ou
- se algum subconjunto S' de $\{s_1, \ldots, s_{i-1}\}$ soma $j - s_i$ (sempre que $j - s_i \geq 0$). Neste caso o conjunto $S' \cup \{s_i\}$ soma j.

Assim, para determinar se o valor de $M[i, j]$ é 1 basta verificar se

- $M[i-1, j] = 1$ ou
- $M[i-1, j-s_i] = 1$ $(j - s_i \geq 0)$.

Como exemplo considere a tabela para $S = \{3, 2, 4\}$ e $C = 7$:

	0	1	2	3	4	5	6	7
	1	0	0	0	0	0	0	0
3	1	0	0	1	0	0	0	0
2	1	0	1	1	0	1	0	0
4	1	0	1	1	1	1	1	1

Observe que para todo $0 \leq i \leq 3$, $M[i, 0] = 1$ e para todo $1 \leq j \leq 7$, $M[0, j] = 0$, já que o conjunto vazio soma 0.

(a) Descreva o algoritmo para construir M.

(b) Calcule a complexidade do algoritmo dinâmico sugerido em termos do tamanho da entrada.
Ajuda: Não é preciso desenvolver a primeira parte para calcular a complexidade. A complexidade não se deve expressar como $\theta(n \times C)$ devido a que o tamanho da entrada do problema é $n + 1$, os n inteiros do conjunto S e o inteiro C!