Análise e Síntese de Algoritmos

Programação Dinâmica [CLRS, Cap. 15]

2011/2012

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Árvores abrangentes
  - Caminhos mais curtos
  - Fluxos máximos
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Emparelhamento de Cadeias de Caracteres
  - Complexidade Computacional
  - Algoritmos de Aproximação

Resumo

Técnicas para Síntese de Algoritmos

Técnicas para Síntese de Algoritmos

- Dividir para conquistar
  - Exemplo: MergeSort
- Programação dinâmica
  - Exemplo: Floyd-Warshall
- Algoritmos greedy
  - Exemplo: Prim, Dijkstra

Programação Dinâmica

Programação Dinâmica

Passos para a realização de um algoritmo baseado em programação dinâmica:

- Caracterizar estrutura de uma solução óptima
- Definir recursivamente o valor de uma solução óptima
- Calcular valor da solução óptima utilizando abordagem bottom-up
- Construir solução a partir da informação obtida

Motivação

Exemplo

Calcular o n-ésimo elemento da sequência de Fibonacci, sabendo que:

$$fib(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ fib(n-1) + fib(n-2) & \text{caso contrário} \end{cases}$$

Motivação

Exemplo

Calcular o n-ésimo elemento da sequência de Fibonacci, sabendo que:

$$fib(n) = \begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ fib(n-1) + fib(n-2) & \text{caso contrário} \end{cases}$$

Solução Recursiva

fib(n)

```
1  if (n ≤ 1)
2      then return n
3      else return fib(n − 1)+fib(n − 2)
```

Motivação

Solução Programação Dinâmica

fib(n)

```
 1  if (n ≤ 1)
 2      then return n
 3      else
 4           Alocar vector f com n+1 posições
 5           f[0] = 0
 6           f[1] = 1
 7           i = 2
 8           while (i ≤ n)
 9                do f[i] = f[i−1] + f[i−2]
10                   i = i+1
11           return f[n]
```

Motivação

Exemplo

$$C_k^n = \begin{cases} 1 & \text{se } k = 0 \text{ ou } k = n \\ C_{k-1}^{n-1} + C_k^{n-1} & \text{se } 0 < k < n \\ 0 & \text{caso contrário} \end{cases}$$

Motivação

Solução Recursiva

C(n,k)

1 **if** $(k = 0$ or $k = n)$
2 **then return** 1
3 **else return** C$(n-1,k-1)$+C$(n-1,k)$

- Cada chamada a $C(n,k)$ retorna 1 ou invoca o cálculo de dois sub-problemas
- Solução é calculada somando 1's !
- Tempo de execução é: $\Omega(C_k^n)$

Motivação

Exemplo

- Número de $C(n,k)$ distintos é apenas $n \times k$
- Complexidade da solução recursiva deriva do cálculo repetido de sub-problemas
  - C(5,3) = C(4,2) + C(4,3)
  - C(4,2) = C(3,1) + C(3,2)
  - C(4,3) = C(3,2) + C(3,3)
- Solução: solução construtiva (bottom-up)
  - Preencher tabela ($n \times k$) (triângulo de Pascal)

Motivação

Comentários

- Problema artificial
- Formulação definida à partida
- Ilustra eliminação de cálculo repetido de sub-problemas

Multiplicação de cadeias de matrizes

Multiplicação de cadeias de matrizes

- $A_1, A_2, \ldots, A_n$ tal que A_i tem dimensões $(p_{i-1} p_i)$
- Tempo para multiplicar as n matrizes é dominado pelo tempo para realizar as multiplicações escalares necessárias
  - Para multiplicar duas matrizes $(r \times s)$ e $(s \times t)$, o número de multiplicações escalares é: $r \times s \times t$
- Número de produtos depende do modo como os produtos de matrizes são organizados
  - Colocação de parêntesis define organização da multiplicação de matrizes

Multiplicação de cadeias de matrizes

Caso concreto

Caso concreto: $A(13 \times 5)$; $B(5 \times 89)$; $C(89 \times 3)$; $D(3 \times 34)$

- $(((A \times B) \times C) \times D)$:
  - $13 \times 5 \times 89 + 13 \times 89 \times 3 + 13 \times 3 \times 34 = 10582$ produtos
- $((A \times B) \times (C \times D))$:
  - $13 \times 5 \times 89 + 89 \times 3 \times 34 + 13 \times 89 \times 34 = 54201$ produtos
- $((A \times (B \times C)) \times D)$:
  - $5 \times 89 \times 3 + 13 \times 5 \times 3 + 13 \times 3 \times 34 =$ 2856 produtos !
- $(A \times ((B \times C) \times D))$:
  - $5 \times 89 \times 3 + 5 \times 3 \times 34 + 13 \times 5 \times 34 = 4055$ produtos
- $(A \times (B \times (C \times D)))$:
  - $89 \times 3 \times 34 + 5 \times 89 \times 34 + 13 \times 5 \times 34 = 26418$ produtos

Multiplicação de cadeias de matrizes

Caso concreto

- Colocar parêntesis numa cadeia de produtos de matrizes tal que o número de multiplicações escalares é minimizado
- Número de colocações possíveis de parêntesis cresce exponencialmente com número de matrizes:

$$P(n) = \begin{cases} 1 & \text{se } n = 1 \\ \sum_{k=1}^{n-1} P(k)\, P(n-k) & \text{se } n \geq 2 \end{cases}$$

$$P(n) = C(n-1) \quad C(n) = C_n^{2n}/(n+1) = \Omega(4^n/n^{3/2})$$

Multiplicação de cadeias de matrizes

Características da Solução Óptima

Seja $A_{1...n}$ solução com colocação óptima de parêntesis

- Admitir solução óptima com parêntesis em k, $A_{1...k}A_{k+1...n}$
- Facto:
  - Colocação de parêntesis para $A_{1...k}$ é também óptima
- Porquê?
  - Caso contrário seria possível encontrar uma melhor colocação de parêntesis para $A_{1...k}$ e portanto para $A_{1...n}$
- Conclusão:
  - Solução óptima para o problema da colocação de parêntesis é composta por soluções óptimas para os seus sub-problemas

Multiplicação de cadeias de matrizes

Solução Recursiva

- m[i,j]: menor número de multiplicações escalares necessário para calcular matriz $A_{i...j}$
- Solução óptima para $A_{1...n}$ é $m[1,n]$
- $i = j$: $m[i,j] = 0$
- $i < j$:
  - Admitir que solução óptima coloca parêntesis em k:
    - $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} p_k p_j$
  - Mas qual é o valor de k?
    - certamente k tem valor entre i e $j-1$
    - considerar todos os valores de k possíveis
- s[i,j]: define colocação óptima de parêntesis entre i e j

Multiplicação de cadeias de matrizes

Solução Recursiva

$$m[i,j] = \begin{cases} 0 & \text{se } i = j \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{se } i < j \end{cases}$$

$$s[i,j] = k \text{ sse } m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$$

Multiplicação de cadeias de matrizes

Cálculo dos Valores de $m[i,j]$

- Número de sub-problemas distintos:
  - 1 para cada $1 \leq i \leq j \leq n$
  - número de problemas: $\Theta(n^2)$
- Problema:
  - solução recursiva requer tempo exponencial
  - resolução repetida dos mesmos subproblemas
- Solução:
  - solução construtiva (bottom-up)
  - tempo de execução: $O(n^3)$

Caracteristicas

Características da Programação Dinâmica

- Solução óptima do problema composta por soluções óptimas para sub-problemas
- Solução recursiva resolve repetidamente os mesmos sub-problemas
  - Sobreposição de problemas
  - Utilizar solução construtiva para evitar resolver repetidamente o mesmo problema
- Reconstrução da solução óptima
- Memorização

Exemplos

Problema da Mochila

- Definição do problema:
  - Dados n objectos $(1, \ldots, n)$ e uma mochila
  - Cada objecto tem um valor v_i e um peso w_i
  - Peso transportado pela mochila não pode exceder W
  - Objectivo:
    - maximizar o valor transportado pela mochila e respeitar a restrição de peso
- Formalização:
  - $x_i = 1$ se objecto i incluído na mochila; 0 caso contrário

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} v_i x_i \\ \text{sujeito a} \quad & \sum_{i=1}^{n} w_i x_i \leq W \\ & x_i \in \{0, 1\} \end{aligned}$$

Exemplos

Problema da Mochila - Tentativa

- Algoritmo que a cada passo selecciona objecto com maior valor v_i/w_i
- Problema:
  - $v_1 = 8, w_1 = 6$
  - $v_2 = 5, w_2 = 5$
  - $v_3 = 5, w_3 = 5$
  - $W = 10$
- Primeiro objecto seleccionado (objecto 1) impede encontrar solução óptima (com objectos 2 e 3)

Exemplos

Problema da Mochila - Solução

- $v[i,j]$: máximo valor que é possível transportar se o peso limite é j, ($j \leq W$), e se apenas podem ser seleccionados os objectos numerados de 1 a i
- Solução óptima encontra-se em $v[n,W]$
- Definição:

$$v[i,j] = max(v[i-1,j], v[i-1,j-w_i] + v_i)$$

$$v[0,j] = 0, \quad j \geq 0$$
$$v[i,j] = -\infty, \; j < 0$$
$$v[i,0] = 0$$

Exemplos

Problema da Mochila - Análise

- Solução óptima é composta por soluções óptimas para os sub-problemas:

$$v[i,j] = max(v[i-1,j], v[i-1,j-w_i] + v_i)$$

- Se $v[i,j]$ é solução óptima:
  - Se objecto i não é incluído, $v[i-1,j]$ é sub-solução óptima
  - Se objecto i é incluído, $v[i-1,j-w_i]$ é sub-solução óptima
- Caso contrário, seria possível obter solução com valor superior ao da solução óptima; uma contradição !

Exemplos

Problema da Mochila - Análise

- Solução recursiva tem tempo de execução exponencial em n e W
  - Mas: número de sub-problemas distintos é $n \times W$
  - Conclusão: resolução repetida dos mesmos sub-problemas !
- Utilizar solução construtiva (bottom-up)
  - preencher tabela ($n \times W$)
- Tempo de execução: $\Theta(nW)$
  - pseudo-polinomial

Exemplos

Maior Sub-Sequência Comum - Definição Problema

- Dada uma sequência $X =< x_1, \ldots, x_n >$, uma sequência $Z =< z_1, \ldots, z_k >$ é uma sub-sequência de X se existe uma sequência estritamente crescente $< i_1, \ldots, i_k >$ tal que para todo o $j = 1, \ldots, k, x_{i_j} = z_j$

- Dadas as sequências $X =< x_1, \ldots, x_n >$ e $Y =< y_1, \ldots, y_m >$, Z é uma sub-sequência comum se Z é sub-sequência de X e de Y
  Obs: $X_i =< x_1, \ldots, x_i >$

- Problema: Encontrar sub-sequência comum de maior comprimento (LCS) entre duas sequências X e Y

Exemplos

Maior Sub-Sequência Comum - Exemplo

- Um caso concreto:
  - $X =< abefcghd >$
  - $Y =< eagbcfdh >$
  - $Z =< abcd >$ é sub-sequência comum de X e Y
- Uma solução exaustiva é impraticável:
  - Considerar inclusão (ou não) de cada caracter de X e de Y
  - Total de sub-sequências em X: 2^n
  - Total de sub-sequências em Y: 2^m
  - Total de casos a analisar: 2^{n+m}
  - Impraticável para valores elevados de n e m

Exemplos

Maior Sub-Sequência Comum - Exemplo

- Alguns resultados formais:
  - Sejam $X =< x_1, \ldots, x_n >$ e $Y =< y_1, \ldots, y_m >$ duas sequências, e seja $Z =< z_1, \ldots, z_k >$ uma LCS de X e Y
  - Se $x_n = y_m$, então $z_k = x_n = y_m$ e Z_{k-1} é LCS de X_{n-1} e Y_{m-1}
  - Se $x_n \neq y_m$, então se $z_k \neq x_n$ implica que Z é LCS de X_{n-1} e Y
  - Se $x_n \neq y_m$, então se $z_k \neq y_m$ implica que Z é LCS de X e Y_{m-1}
- Abordagem:
  - Se $x_n = y_m$, encontrar LCS W de X_{n-1} e Y_{m-1}
    - Adicionar $x_n = y_m$ a W permite obter Z
  - Se $x_n \neq y_m$, encontrar LCS' s para X_{n-1} e Y e para X e Y_{m-1}
    - Escolher a maior LCS

Exemplos

Maior Sub-Sequência Comum - Formulação

- $c[i,j]$: comprimento da LCS para as sequências X_i e Y_j
- Formulação:

$$c[i,j] = \begin{cases} 0 & \text{se } i = 0 \text{ ou } j = 0 \\ c[i-1,j-1]+1 & \text{se } i,j > 0 \text{ e } x_i = y_j \\ max(c[i,j-1], c[i-1,j)) & \text{se } i,j > 0 \text{ e } x_i \neq y_j \end{cases}$$

- Tempo de execução: $O(n\ m)$

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-									
e									
a									
g									
b									
c									
f									
d									
h									

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0								
a	0								
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0								
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0								
b	0								
c	0								
f	0								
d	0								
h	0								

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0								
c	0								
f	0								
d	0								
h	0								

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0	1							
c	0								
f	0								
d	0								
h	0								

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0							
a	0	1							
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1							
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1							
b	0	1							
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1	1						
b	0	1	2						
c	0	1							
f	0	1							
d	0	1							
h	0	1							

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0						
a	0	1	1						
g	0	1	1						
b	0	1	2						
c	0	1	2						
f	0	1	2						
d	0	1	2						
h	0	1	2						

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1					
a	0	1	1	1					
g	0	1	1	1					
b	0	1	2	2					
c	0	1	2	2					
f	0	1	2	2					
d	0	1	2	2					
h	0	1	2	2					

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1				
a	0	1	1	1	1				
g	0	1	1	1	1				
b	0	1	2	2	2				
c	0	1	2	2	2				
f	0	1	2	2					
d	0	1	2	2					
h	0	1	2	2					

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1				
a	0	1	1	1	1				
g	0	1	1	1	1				
b	0	1	2	2	2				
c	0	1	2	2	2				
f	0	1	2	2	3				
d	0	1	2	2					
h	0	1	2	2					

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

Exemplos

Maior Sub-Sequência Comum - Exemplo

	-	a	b	e	f	c	g	h	d
-	0	0	0	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1
a	0	1	1	1	1	1	1	1	1
g	0	1	1	1	1	1	2	2	2
b	0	1	2	2	2	2	2	2	2
c	0	1	2	2	2	3	3	3	3
f	0	1	2	2	3	3	3	3	3
d	0	1	2	2	3	3	3	3	4
h	0	1	2	2	3	3	3	4	4

Exemplos

Realizar Trocos - Definição Problema

- Dado um conjunto de moedas, denominadas $1, \ldots, n$, com valores $d_1, \ldots, d_n$, calcular o menor número de moedas cuja soma de valores é N
  - Número ilimitado de moedas de cada denominação
- Solução greedy pode não funcionar:
  - $v_1 = 1; v_2 = 5; v_3 = 20; v_4 = 25$
  - Troco de 40?!
- Solução baseada em programação dinâmica

Exemplos

Realizar Trocos - Formulação

- $c[i,j]$: Menor número de moedas necessárias para pagar j unidades, $0 \leq j \leq N$, utilizando apenas moedas com denominação entre 1 e i, $1 \leq i \leq n$
- Objectivo é calcular $c[n,N]$
- Formulação:

$$c[i,j] = \begin{cases} 0 & \text{se } i > 0 \text{ e } j = 0 \\ +\infty & \text{se } i = 0 \text{ ou } j < 0 \\ \min(c[i-1,j], c[i,j-d_i]+1) & \text{caso contrário} \end{cases}$$

- Tempo de execução: $O(n\,N)$

Maior Palindromo - Definição do Problema

- Dada uma sequência $X =< x_1, \ldots, x_n >$, calcular a maior sub-sequência $Z =< z_1, \ldots, z_k >$ de X tal que Z seja um palindromo.
- Exemplo:
  - $X =< abcfdbca >$
  - $Z =< abdba >$

Memorização

Memorização (Memoization)

- Permite obter tempo de execução das soluções bottom-up, mas utilizando abordagem recursiva
  - É necessário memorizar resultados de sub-problemas já resolvidos
- Exemplo: caminhos mais curtos num DAG, com DFS
- Exemplo: cálculo das combinações
  - Não calcular todo o triângulo de Pascal
  - Calcular apenas as entradas necessárias
  - Calcular cada entrada apenas 1 vez

Memorização

Memorização - Exemplo

- Tabela $c[n,k]$:
  - entradas com $k = 0$ ou $k = n$ inicializadas com valor 1
  - restantes entradas inicializadas com valor -1
- Complexidade Solução: $O(n\,k)$

Cm(n,k)

```
1  if (c[n-1,k] < 0)
2     then c[n-1,k] = Cm(n-1,k)
3  if (c[n-1,k-1] < 0)
4     then c[n-1,k-1] = Cm(n-1,k-1)
5  return c[n-1,k-1] + c[n-1,k]
```