Algoritmos Paralelos Eficientes para Alguns Problemas em Processamento de Cadeias de Caracteres

Siang Wun Song

JAI/SBC 2007 - Rio de Janeiro
Parte 3

Problema de Subseqüência Comum Mais Longa

- Dadas duas seqüências de símbolos, a obtenção da *subseqüência mais longa comum* a ambas é um importante problema com aplicações em comparação de seqüências de DNA, compressão de dados, etc.
- Consideramos o problema mais geral denominado *toda-subcadeia subseqüência comum mais longa* e apresentamos um algoritmo paralelo eficiente em tempo e espaço.
- Antes vamos introduzir alguns conceitos e nomenclatura.

Subcadeia, Subseqüência

- Considere uma seqüência ou cadeia de símbolos. Exemplo:
  “helloworld”
- Uma subcadeia de uma cadeia é qualquer fragmento contíguo da cadeia dada. Exemplo:
  “lowor” (“helloworld”)
- Uma subseqüência de uma cadeia é obtida pela remoção de zero ou mais símbolos da cadeia original. Exemplo:
  “hold” (“helloworld”)
- Notação: Dada uma cadeia $Y = y_1 \ldots y_n$ denotamos por Y_i^j a subcadeia de Y desde y_i a y_j.

Subseqüência Comum Mais Longa - LCS

- Dadas duas cadeias X e Y, o problema de *subseqüência comum mais longa* (LCS) acha o comprimento da subseqüência mais longa comum a ambas as cadeias. Exemplo:
  X = “helloworld”
  Y = “hollywood”
  O comprimento da subseqüência comum mais longa é 6 e a subseqüência comum:
  “hllwod” ou
  “hllood”

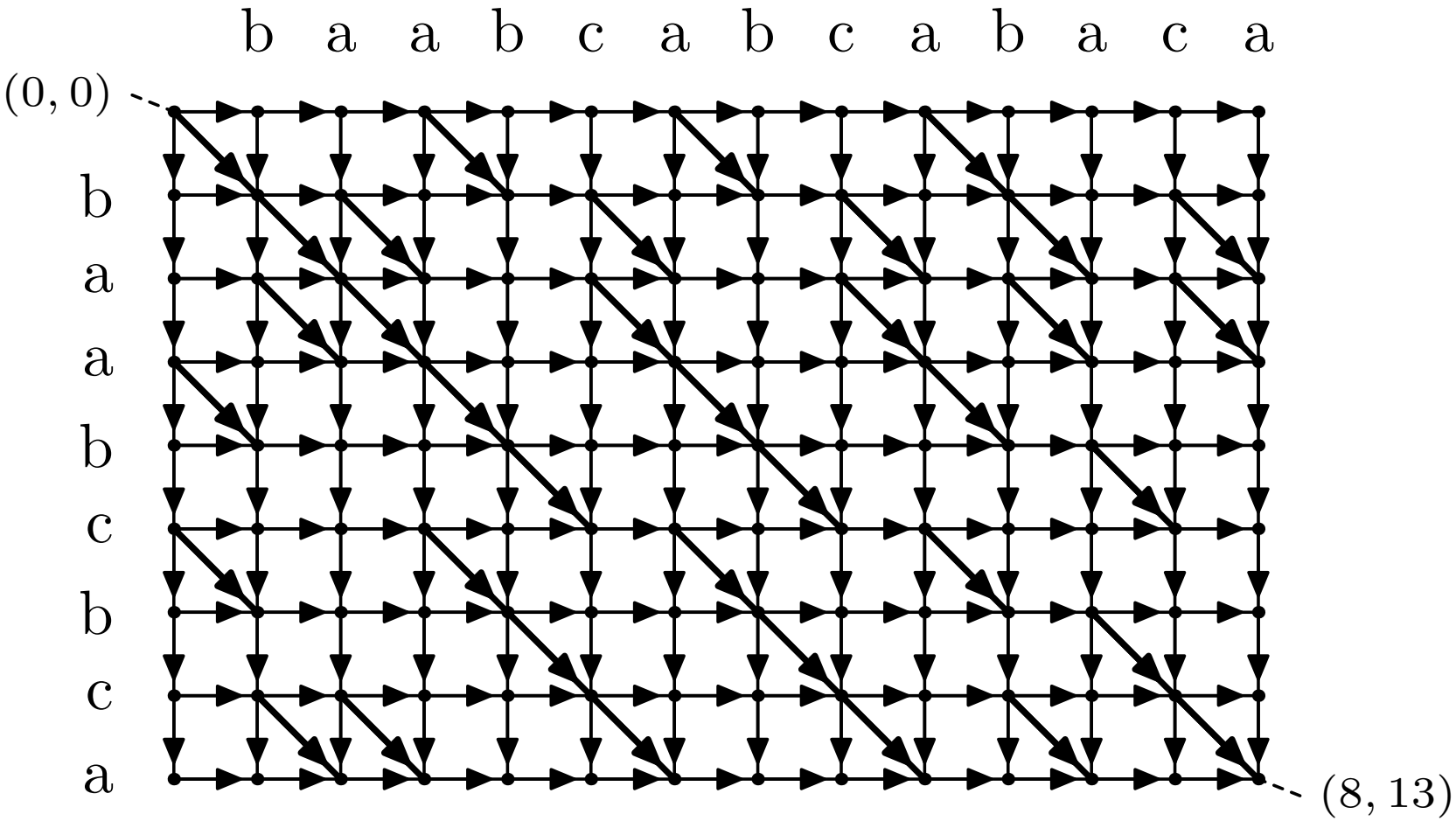
Toda-Subcadeia Subseqüência Comum Mais Longa - ALCS

- Considere cadeias X e Y de comprimentos m e n.
- *Toda-subcadeia subseqüência comum mais longa* (ALCS) acha os comprimentos das subseqüências comuns mais longas entre X e *qualquer* subcadeia de Y.
- LCS e ALCS podem ser modelados por meio de um grafo orientado acíclico em grade (*grid directed acyclic graph - GDAG*).
- Vamos apresentar agora este grafo.

Grafo Orientado Acíclico - GDAG

- Considerem
  $X =$ baabcbca (comprimento m)
  $Y =$ baabcabcabaca (comprimento n).
- O GDAG tem $(m+1) \times (n+1)$ vértices.
- As arestas do GDAG tem pesos:
  As arestas verticais e horizontais têm peso 0.
  As aresta do vértice $(i-1, j-1)$ ao vértice (i, j) tem peso 1 se $x_i = y_j$.
  Se $x_i \neq y_j$, a aresta tem peso 0 e pode ser ignorada.

GDAG



Matriz C_G

Para $0 \leq i \leq j \leq n$, $C_G(i,j)$ é o *custo* ou *peso total* do melhor caminho entre vértices $T_G(i)$ e $B_G(j)$, representando o comprimento da subseqüência comum mais longa entre X e a subcadeia Y_{i+1}^{j}. Se $i \geq j$ (Y_{i+1}^{j} é vazia ou não-existente), $C_G(i,j) = 0$.

- As definições de C_G (acima), D_G e V_G (a serem vistos) são devidas a M. Lu e H. Lin 1994.
- T_G primeira linha da GDAG.
- B_G última linha da GDAG.

Matriz C_G

$C_G(i,j)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1	2	3	4	5	6	6	7	8	8	8	8	8
1	0	0	1	2	3	4	5	5	6	7	7	7	7	7
2	0	0	0	1	2	3	4	4	5	6	6	6	6	7
3	0	0	0	0	1	2	3	3	4	5	5	6	6	7
4	0	0	0	0	0	1	2	2	3	4	4	5	5	6
5	0	0	0	0	0	0	1	2	3	4	4	5	5	6
6	0	0	0	0	0	0	0	1	2	3	3	4	4	5
7	0	0	0	0	0	0	0	0	1	2	2	3	3	4
8	0	0	0	0	0	0	0	0	0	1	2	3	3	4
9	0	0	0	0	0	0	0	0	0	0	1	2	3	4
10	0	0	0	0	0	0	0	0	0	0	0	1	2	3
11	0	0	0	0	0	0	0	0	0	0	0	0	1	2
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interpretando a matriz C_G

- Por exemplo $C_G(0, 9) = 8$.
- Isso significa que comprimento da LCS entre X = baabcbca e Y_1^9= baabcabca é 8.
- Lembre-se a notação:
  Dada uma cadeia $Y = y_1 \ldots y_n$
  denotamos por Y_i^j a subcadeia de Y desde y_i a y_j.

Intrepretando ainda C_G

- Vimos que $C_G(0,9)$ é 8.
- Mas $C_G(0,10)$ também é 8.
- Isso mostra que há uma posição mais à esquerda (no exemplo 9) para se chegar a um valor de comprimento determinado (no caso 8).
- Essa observação motiva a definição da matriz D_G.

Matriz C_G

$C_G(i,j)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1	2	3	4	5	6	6	7	8	8	8	8	8
1	0	0	1	2	3	4	5	5	6	7	7	7	7	7
2	0	0	0	1	2	3	4	4	5	6	6	6	6	7
3	0	0	0	0	1	2	3	3	4	5	5	6	6	7
4	0	0	0	0	0	1	2	2	3	4	4	5	5	6
5	0	0	0	0	0	0	1	2	3	4	4	5	5	6
6	0	0	0	0	0	0	0	1	2	3	3	4	4	5
7	0	0	0	0	0	0	0	0	1	2	2	3	3	4
8	0	0	0	0	0	0	0	0	0	1	2	3	3	4
9	0	0	0	0	0	0	0	0	0	0	1	2	3	4
10	0	0	0	0	0	0	0	0	0	0	0	1	2	3
11	0	0	0	0	0	0	0	0	0	0	0	0	1	2
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Matriz D_G

Seja G o GDAG para o problema ALCS para as cadeias X e Y. Para $0 \leq i \leq n$, $D_G(i, 0) = i$ e para $1 \leq k \leq m$, $D_G(i, k)$ indica o valor de j tal que $C_G(i, j) = k$ e $C_G(i, j - 1) = k - 1$. Se não há tal valor, então $D_G(i, k) = \infty$.

Matriz D_G

$D_G(i,j)$	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	8	9
1	1	2	3	4	5	6	8	9	∞
2	2	3	4	5	6	8	9	13	∞
3	3	4	5	6	8	9	11	13	∞
4	4	5	6	8	9	11	13	∞	∞
5	5	6	7	8	9	11	13	∞	∞
6	6	7	8	9	11	13	∞	∞	∞
7	7	8	9	11	13	∞	∞	∞	∞
8	8	9	10	11	13	∞	∞	∞	∞
9	9	10	11	12	13	∞	∞	∞	∞
10	10	11	12	13	∞	∞	∞	∞	∞
11	11	12	13	∞	∞	∞	∞	∞	∞
12	12	13	∞	∞	∞	∞	∞	∞	∞
13	13	∞	∞	∞	∞	∞	∞	∞	∞

Significado da Matriz D_G

- Se começarmos da posição i da linha de topo e procedermos para a última linha à uma posição dada por $D_G(i,k)$, então podemos obter um caminho de peso total k.
- Assim, $D_G(i,k)$ fornece a posição mais à esquerda que dá o peso total k.
- Por exemplo, Considere $D_G(0,8) = 9$. Começe com o índice 0 da linha de topo e pega arestas em qualquer uma das três direções: diagonal (peso 1) ou arestas horizontal ou vertical (peso 0). Para obter um peso total 8, a posição mais à esquerda na última linha será 9. Assim, temos $D_G(0,8) = 9$.

Propriedades da matriz D_G

M. Lu e H. Lin (1994) mostram que D_G apresenta as propriedades:

- Para $0 \leq i \leq n-1$, linha D_G^{i+1} pode ser obtida da linha D_G^i pela remoção do primeiro elemento ($D_G^i(0) = i$) e a inserção de apenas um novo elemento (que pode ser ∞).
- Os elementos inteiros de cada linha da D_G estão em ordem crescente estrita (não há elementos repetidos, exceto o valor ∞).

Com isso, temos uma forma econômica de armazenar seus valores: Basta armazenarmos a primeira linha de D_G, mais um vetor contendo os novos elementos nas próximas linhas. Esse vetor será chamado V_G.

Vetor V_G

Para $1 \leq i \leq n$, $V_G(i)$ é o valor do elemento finito que está presente na linha D_G^i mas não está presente na linha D_G^{i-1}. Se tal elemento finito não existir, então $V_G(i) = \infty$.

Vetor V_G

k	1	2	3	4	5	6	7	8	9	10	11	12	13
$V_G(k)$	∞	13	11	∞	7	∞	∞	10	12	∞	∞	∞	∞

Algoritmo Seqüencial para LCS e ALCS

Alves, Cáceres e Song [Discrete Applied Math. 2007] mostram que:

> *Dadas duas cadeias X e Y de comprimentos m e n, respectivamente, é possível resolver o problema ALCS seqüencialmente em tempo $O(mn)$ e espaço $O(n)$.*

Não vamos mostrar o algoritmo seqüencial, mas sim discutiremos o algoritmo paralelo.

Algoritmo Paralelo para ALCS

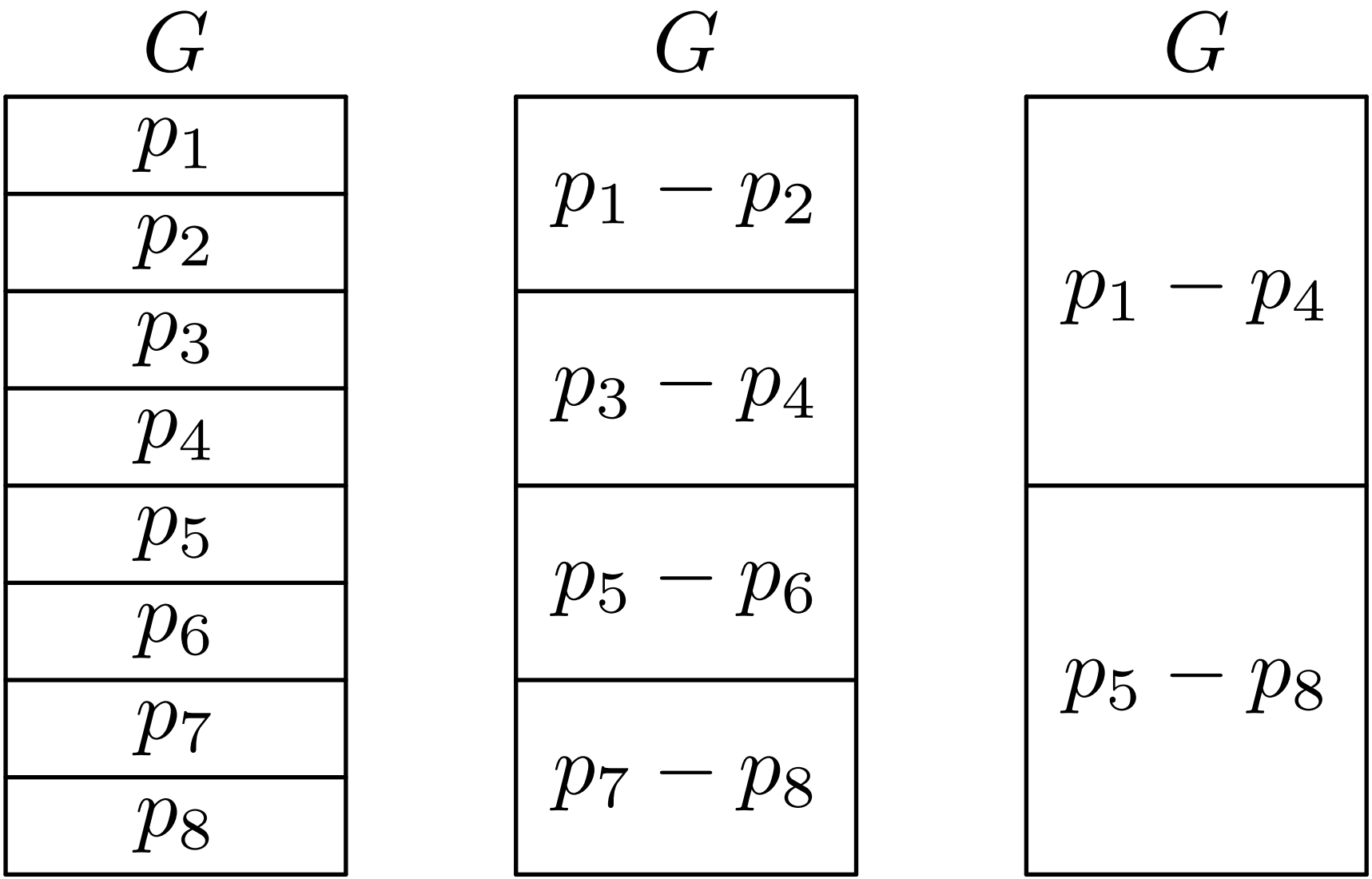
- Dadas duas cadeias X e Y de comprimentos m e n.
- Supor X dividida em p subcadeias, cada uma de tamanho m/p.
- GDAG dividido em p pedaços horizontais, cada um com m/p+1 linhas cada. (Dois pedaços consecutivos compartilham uma linha comum.)
- Cada processador corresponde a um pedaço horizontal e resolve seqüencialmente ALCS para as cadeias $X_{mi/p+1}^{m(i+1)/p}$ e Y, e computa o valor local de D_G.

Algoritmo Paralelo - Divisão e Conquista

O algoritmo paralelo para ALCS consiste de duas fases:

1. Cada um dos p processadores executa o algoritmo seqüencial ALCS para os seus *pedaços* locais e computa o D_G local.
2. Em $\log p$ rodadas pares de soluções contíguas são juntadas sucessivamente para obter a solução do problema original.

Ilustração - Divisão e Conquistta



$O(\log p)$ rodadas de comunicação são usadas.

Junção de Resultados Parciais

- Em cada rodada, D_G parcial é computado. Resultados parciais precisam ser juntados para a próxima rodada. Essa é a parte mais difícel do algoritmo.
- Em particular o armazenamento de D_G de forma compacta para ser transimitido com eficiência.
- Vamos discutir agora o armazenamento.

Armazenamento Direto de D_G Inadequado

Armazenamento direto de D_G usa espaço $O(mn)$ e permite acesso aos valores de D_G em tempo O(1). Inadeqado pelo espaço grande onde há grande redundância.

$D_G(i,j)$	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	8	9
1	1	2	3	4	5	6	8	9	∞
2	2	3	4	5	6	8	9	13	∞
3	3	4	5	6	8	9	11	13	∞
4	4	5	6	8	9	11	13	∞	∞
5	5	6	7	8	9	11	13	∞	∞
6	6	7	8	9	11	13	∞	∞	∞
7	7	8	9	11	13	∞	∞	∞	∞
8	8	9	10	11	13	∞	∞	∞	∞
9	9	10	11	12	13	∞	∞	∞	∞
10	10	11	12	13	∞	∞	∞	∞	∞
11	11	12	13	∞	∞	∞	∞	∞	∞
12	12	13	∞	∞	∞	∞	∞	∞	∞
13	13	∞	∞	∞	∞	∞	∞	∞	∞

Armazenamento da linha D_G^0 e V_G Inadequado

Usar apenas a primeira linha de D_G e vetor V_G. Espaço usado apenas $O(n)$ mas é inadequado pois demora para obter os valores originais de D_G.

$D_G(i,j)$	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	8	9

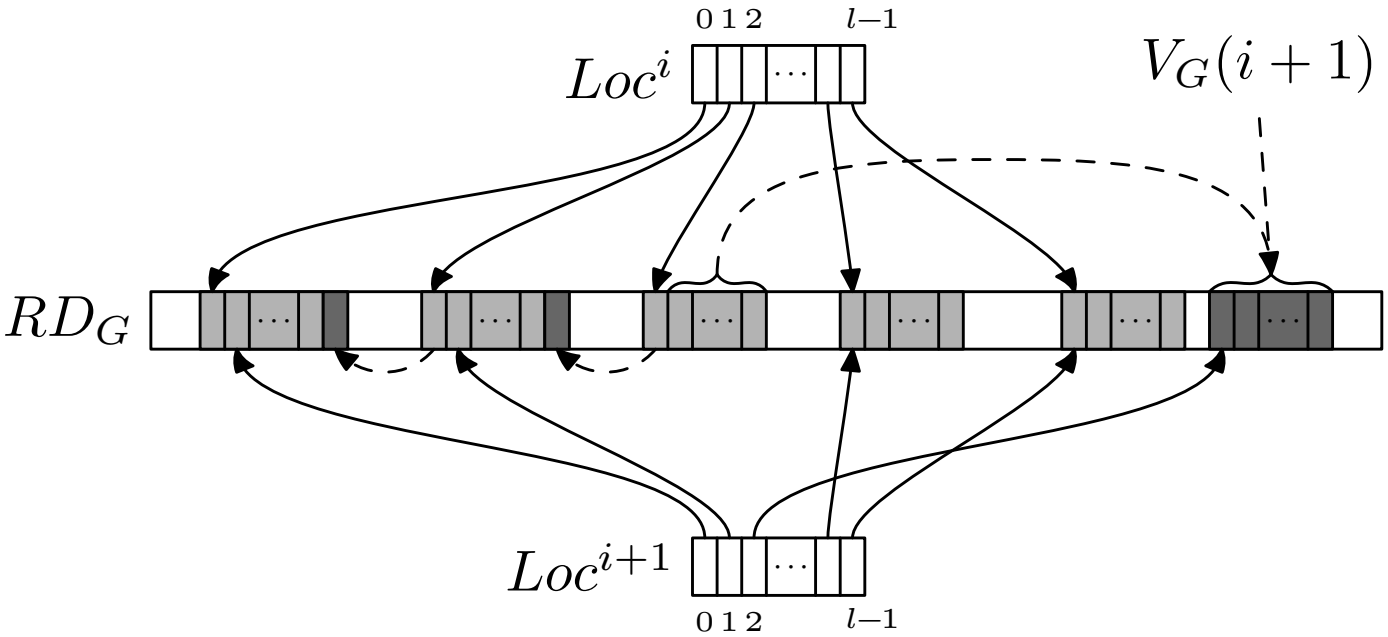
k	1	2	3	4	5	6	7	8	9	10	11	12	13
$V_G(k)$	∞	13	11	∞	7	∞	∞	10	12	∞	∞	∞	∞

Estrutura de Dado Compacta Proposta

Usa espaço $O(n\sqrt{m})$ e permite acesso de valores individuais de D_G em tempo $O(1)$.

- A estrutura compacta é construída a partir de D_G^0 e V_G leva em tempo $O(n\sqrt{m})$.
- A construção é incremental pela adição de uma linha de D_G cada vez.
- Os valores de D_G são armazenados em um vetor chamado RD_G (D_G *reduzido*) de tamanho $O(nl)$, onde $l = \lceil \sqrt{m+1} \rceil$.

Estrutura RD_G



- Linha D_G^i é dividida em no máximo l sub-vetores ($l = \lceil \sqrt{m+1} \rceil$) de tamanho l.
- Os sub-vetores são armazenados em posições separadas de RD_G.
- Temos um vetor adicional denotado por Loc^i de tamanho l para indicar a localização de cada sub-vetor.

Teorema sobre a Estrutura de Dados Compacta

Considere o GDAG G do problema ALCS para as cadeias X e Y. De D_G^0 e V_G podemos reconstruir a representação de D_G em tempo e espaço $O\left(n\sqrt{m'}\right)$ de tal modo que qualquer valor de of D_G possa ser acessado em tempo $O(1)$.

Junção dos Resultados Parciais

- A junção dos resultados envolve vários detalhes complexos e será omitida nesta apresentação.
- O importante é lembrar que foi preciso tomar proveito das redundâncias e das similaridades (como na representação da matriz D_G) a fim de obter representações compactas.

Teorema Principal

Concluímos com o principal resultado que é um algoritmo CGM de *speed-up* linear para o problema ALCS.

Dadas duas cadeias X e Y de comprimentos m e n, respectivamente, o problema ALCS pode ser resolvido por $p < \sqrt{m}$ processadores em tempo $O(mn/p)$, espaço por processador $O(n\sqrt{m})$, e $O(C \log p)$ rodadas de comunicação, para alguma constante escolhida C, em que $O(mp^{1/C} + n)$ dados são transmitidos de/para cada processador.