Range Minima

Algoritmos Paralelos

10/05/2002

Ancestral Comum Mais Baixo

Definição 1 *O* ***ancestral comum mais baixo*** *(Lowest Common Ancestor - LCA)de dois vértices u e v de uma árvore com raiz é o vértice w que é um ancestral de u e v e que está mais distante da raiz. Denotamos* $w = LCA(u, v)$.

Definição 2 *O* ***problema do ancestral comum mais baixo*** *consiste em pré-processar uma árvore com raiz de forma que consultas* $LCA(u, v)$*, para quaisquer vértices u e v da árvore, possam ser respondidas em tempo seqüencial constante.*

Range Minima

Definição 3 *Dado um vetor de n números reais* $A = (a_1, a_2, \ldots, a_n)$*, definimos* $MIN(i, j) = \min\{a_i, \ldots, a_j\}$*. O problema de* **range minima** *consiste em pré-processar o vetor A de forma que consultas* $MIN(i, j)$*, para qualquer* $1 \leq i \leq j \leq n$*, possam ser respondidas em tempo constante.*

Algoritmos Seqüenciais

- O algoritmo para o problema de *range minima*, no modelo CGM, utiliza dois algoritmos seqüenciais, que são executados utilizando os dados locais em cada processador.
- Estes algoritmos para o problema de *range minima* foram apresentados por Gabow *et al* (1984) e Alon e Schieber (1987).

Algoritmo de Gabow *et al*

Este algoritmo seqüencial utiliza a estrutura de dados **árvore Cartesiana** (1980).

Definição 4 *A* ***árvore Cartesiana*** *de um vetor* $A = (a_1, a_2, \ldots, a_n)$*, de* n *números reais distintos, é uma árvore binária cujos nós têm como rótulos os valores do vetor* A*. A raiz da árvore tem como rótulo* $a_m = \min\{a_1, a_2, \ldots, a_n\}$*. Sua subárvore esquerda é uma árvore Cartesiana para* $A_{1,m-1} = (a_1, a_2, \ldots, a_{m-1})$*, e sua subárvore direita é uma árvore Cartesiana para* $A_{m+1,n} = (a_{m+1}, \ldots, a_n)$*. A árvore para um vetor vazio é a árvore vazia.*

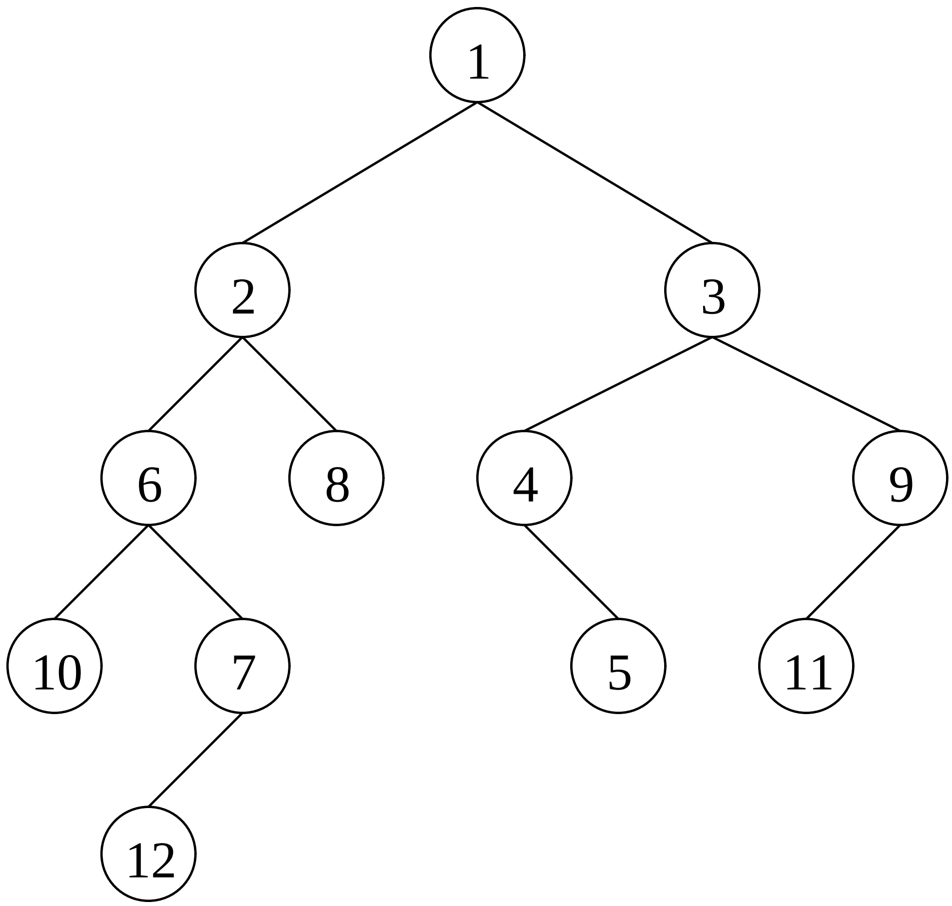


Figura 1: Árvore Cartesiana correspondente ao vetor $(10, 6, 12, 7, 2, 8, 1, 4, 5, 3, 11, 9)$.

ALGORITMO ***Range Minima*** **- Gabow**

Entrada: um vetor $A = (a_1, a_2, \ldots, a_n)$ com n números reais.

Saída: uma estrutura de dados que responde a consultas $MIN(i, j)$ em tempo constante.

1. Construir uma árvore Cartesiana para A.
2. Aplicar um algoritmo seqüencial linear para o problema do LCA na árvore Cartesiana A.

fim algoritmo

- A construção da árvore Cartesiana leva tempo linear. Existem vários algoritmos seqüenciais lineares para problema de LCA. Assim, qualquer consulta $MIN(i, j)$ é feita como na descrição a seguir.

- **Processando Consultas.**

  A partir da definição recursiva de árvore Cartesiana, o valor de $MIN(i, j)$ é o valor do LCA de a_i e a_j. Assim, cada consulta de *range minima* pode ser respondida em tempo constante através de uma consulta LCA na árvore Cartesiana.

  Logo, o problema de *range minima* é resolvido em tempo linear.

Algoritmo de Alon e Schieber

- O algoritmo de Alon e Schieber tem complexidade de tempo $O(n \log n)$.
- Apesar de sua complexidade não ser linear, este algoritmo é crucial na descrição do algoritmo CGM.
- Na descrição do algoritmo, vamos considerar, sem perda de generalidade, que n é uma potência de 2.

ALGORITMO ***Range Minima*** **- Alon e Schieber**

Entrada: um vetor $A = (a_1, a_2, \ldots, a_n)$ com n números reais.

Saída: uma estrutura de dados que responde a consultas $MIN(i, j)$ em tempo constante.

{ Sem perda de generalidade vamos considerar que n é uma potência de 2. }

1. Construir uma árvore binária completa T com n folhas.

2. Associar os elementos de A às folhas de T.

3. Para cada vértice v de T calculamos os vetores P_v e S_v.
   { P_v e S_v são os vetores que armazenam os vetores de mínimo prefixo e mínimo sufixo, respectivamente, dos elementos das folhas da subárvore com raiz v. }

fim algoritmo

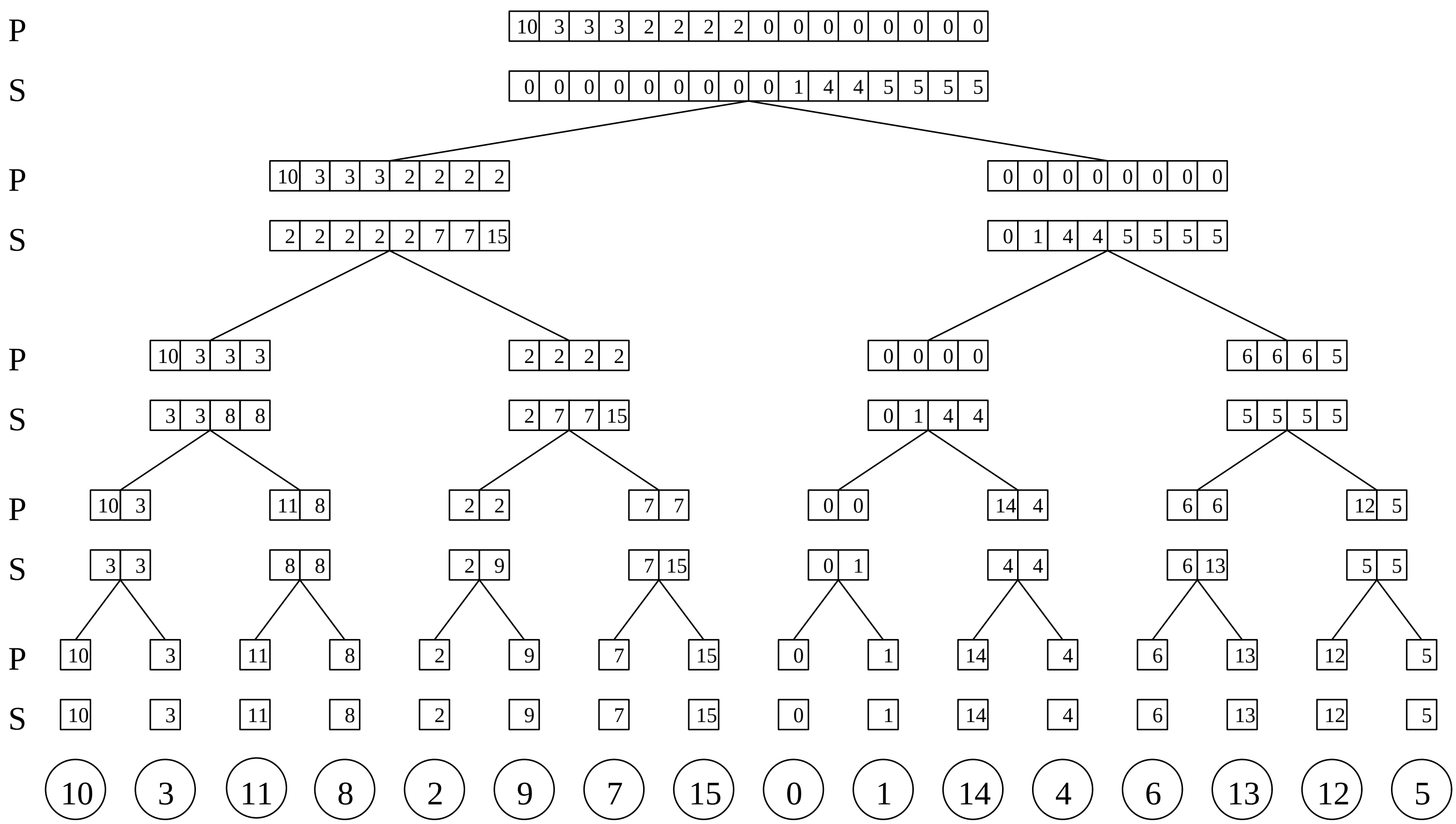


Figura 2: Árvore-PS gerada pelo algoritmo de Alon e Schieber para o vetor $(10, 3, 11, 8, 2, 9, 7, 15, 0, 1, 14, 4, 6, 13, 12, 5)$.

- A árvore T construída pelo algoritmo 1 será denominada **árvore-PS**.
- **Processamento de consultas.**

  Para determinar $MIN(i, j)$, $1 \leq i \leq j \leq n$, encontramos $w = \mathrm{LCA}(a_i, a_j)$ em T. Sejam v e u os filhos esquerdo e direito de w, respectivamente. Então, $MIN(i, j)$ é o mínimo entre o valor de S_v na posição correspondente a a_i e o valor de P_u na posição correspondente a a_j.
- O tempo constante para efetuar uma consulta LCA em uma árvore-PS vem do fato de esta árvore ser binária completa.

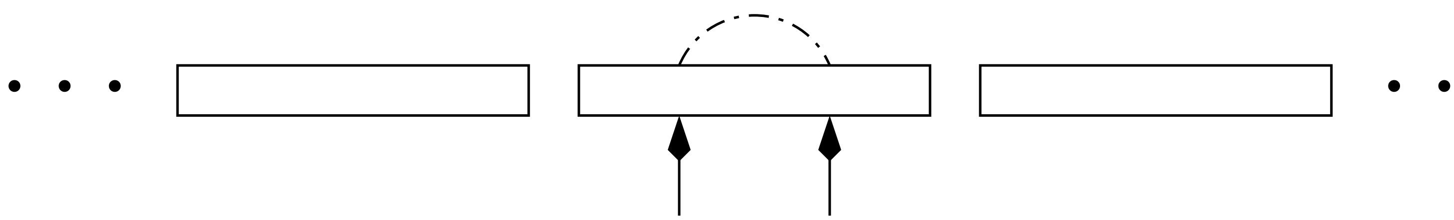
Algoritmo CGM

- Estamos preocupados em diminuir o número de rodadas de comunicação, de forma que nos concentramos em computações com os dados locais dos processadores.
- Utilizamos os algoritmos seqüenciais vistos para construir um algoritmo, no modelo CGM, para o problema de *range minima*.
- Tempo: $O(\frac{n}{p})$
- Rodadas de comunicação: $O(1)$ rodadas de comunicação.
- A maior dificuldade é como armazenar os dados nos processadores para que as consultas sejam feitas em tempo constante, respeitando a limitação de $O(\frac{n}{p})$ posições de memória, imposta pelo modelo CGM.

Notação

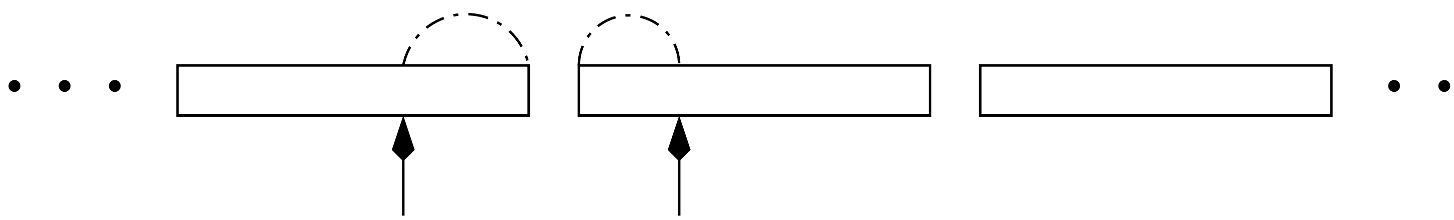
Notação 1 *Dado um vetor* $A = (a_1, a_2, \ldots, a_n)$*, consideramos* $A[i]$*, com* $1 \leq i \leq n$*, o conteúdo da posição* i *do vetor* A*;* $A[i \ldots j]$*, com* $1 \leq i \leq j \leq n$*, o subvetor* $(a_i, \ldots, a_j)$*.*

- A idéia do algoritmo baseia-se na observação de como as consultas $MIN(i,j)$ podem ser feitas.
- Cada processador armazena $\frac{n}{p}$ posições contíguas do vetor.
- Dado $A = (a_1, a_2, \ldots, a_n)$, temos os subvetores $A_i = (a_{i\frac{n}{p}+1}, \ldots, a_{(i+1)\frac{n}{p}})$, para $0 \leq i \leq p-1$.
- Dependo da localização de a_i e a_j nos processadores, temos os seguintes casos:
  1. se a_i e a_j estão em um mesmo processador, o domínio do problema se reduz ao subvetor armazenado no processador. Assim, precisamos de uma estrutura para responder este tipo de consultas em tempo constante. Esta estrutura é obtida, em cada processador, pelo algoritmo de Gabow.

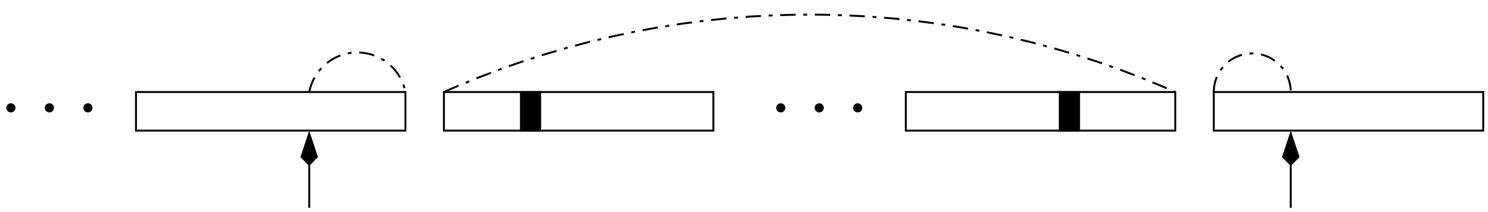


2. se a_i e a_j estão em processadores distintos p_i e p_j (spg, $i < j$), respectivamente, temos dois subcasos:

   (a) se $i = j - 1$, a_i e a_j estão em processadores vizinhos, $MIN(i, j)$ corresponde ao mínimo entre o mínimo de a_i até o fim do vetor A_i e o mínimo do início de A_j até a_j. Estes mínimos podem ser determinados pelo mesmo algoritmo do item anterior. Para determinar o mínimo dos mínimos é necessária uma rodada de comunicação.



(b) se $i < j-1$, $MIN(i,j)$ corresponde ao mínimo entre o mínimo do subvetor $A_i[i \ldots (i+1)\frac{n}{p}]$, o mínimo do subvetor $A_j[j\frac{n}{p}+1 \ldots j]$ e os mínimos dos subvetores $A_{i+1}, \ldots, A_{j-1}$. Os dois primeiros mínimos são obtidos como no subcaso anterior. Os mínimos dos vetores $A_{i+1}, \ldots, A_{j-1}$ são obtidos facilmente pela árvore Cartesiana. O mínimo entre eles corresponde ao problema de *range minima* restrito ao vetor de mínimos dos dados dos processadores. Assim, precisamos de uma estrutura de dados para responder a estas consultas em tempo constante. Como o vetor de mínimos contém apenas p valores, esta estrutura pode ser obtida pelo algoritmo de Alon e Schieber.



- A dificuldade do caso 2.b é que não podemos construir explicitamente a árvore-PS em um processador (ou todos) como na descrição do algoritmo, pois isto gastaria uma memória de tamanho $O(p \log p)$, e a memória no modelo CGM é $O(\frac{n}{p})$, com $\frac{n}{p} \geq p$.
- Para contornar esta dificuldade construímos vetores $\bar{P}$ e $\bar{S}$ de $\log p + 1$ posições cada um, que armazenam algumas informações da árvore-PS T em cada processador.

- Seja um processador i, com $0 \leq i \leq p-1$.
- Seja b_i o valor do mínimo dos valores do vetor A_i.
- Seja v um vértice de T tal que a subárvore com raiz v tem b_i como folha e sejam d_v a **profundidade de v** em T, que é o comprimento do caminho da raiz até v; e l_v o **nível de v**, que é a altura da árvore menos a profundidade de v ($l_v = \log p - d_v$, pois a árvore T tem altura $\log p$).
- O vetor $\bar{P}$ (respectivamente, $\bar{S}$) contém na posição l_v o valor do vetor P_v (respectivamente, S_v), do nível l_v de T, na posição correspondente à folha b_i. Ou seja, temos que $\bar{P}[l_v] = P_v[i \bmod 2^{l_v} + 1]$.

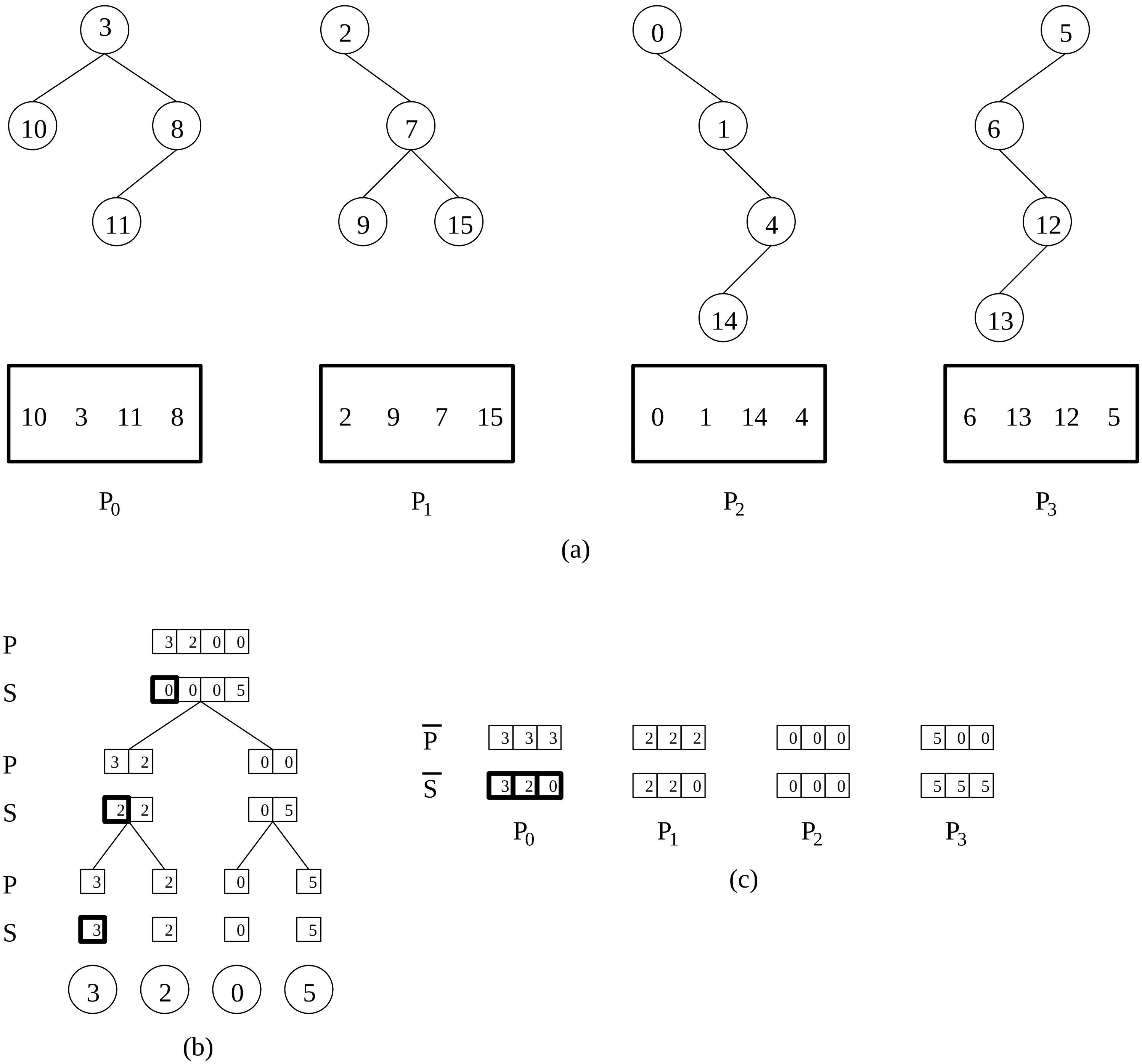


Figura 3: Execução do algoritmo no vetor $(10, 3, 11, 8, 2, 9, 7, 15, 0, 1, 14, 4, 6, 13, 12, 5)$

.

ALGORITMO *Range Minima*

Entrada: um vetor $A = (a_1, a_2, \ldots, a_n)$ com n números reais.

Saída: uma estrutura de dados que responde a consultas $MIN(i, j)$ em tempo constante.

1. Cada processador i executa seqüencialmente o algoritmo de Gabow.
2. { Cada processador constrói um vetor $B = (b_i)$ de tamanho p, que conterá o mínimo dos dados armazenados em cada processador. }
   - 2.1. Cada processador i calcula $b_i = \min A_i = \min\{a_{i\frac{n}{p}+1}, \ldots, a_{(i+1)\frac{n}{p}}\}$.
   - 2.2. Cada processador i envia b_i para os outros processadores.
   - 2.3. Cada processador i coloca o valor recebido do processador k, $k \in \{0, \ldots, p-1\} \setminus \{i\}$, em b_k.
3. Cada processador i executa os procedimentos Constrói_$\bar{P}$ e Constrói_$\bar{S}$.

fim algoritmo

PROCEDIMENTO Constrói_$\bar{P}$.

Entrada: o vetor $B = (b_0, b_1, \ldots, b_{p-1})$ com p números reais.

Saída: o vetor $\bar{P}$ de $\log p$ posições.

$\bar{P}[0] \leftarrow b_i$
$apontador \leftarrow i$
$inordem \leftarrow 2 * i + 1$
para $k \leftarrow 1$ **at** $\log p$ **faça**
    $\bar{P}[k] \leftarrow \bar{P}[k-1]$
    se $\lfloor inordem/2^k \rfloor \bmod 2 = 0$
        então para $l \leftarrow 1$ **at** 2^{k-1} **faça**
            $apontador \leftarrow apontador - l$
            se $\bar{P}[k] > B[apontador]$
                então $\bar{P}[k] \leftarrow B[apontador]$

fim procedimento

Teorema 1 *O algoritmo CGM resolve o problema de* range minima *em tempo* $O(\frac{n}{p})$ *usando* $O(1)$ *rodadas de comunicação.*

Prova.

- O passo 1 é executado em tempo seqüencial $O(\frac{n}{p})$ e não utiliza comunicação.

- O passo 2 roda em tempo seqüencial $O(\frac{n}{p})$ e efetua uma rodada de comunicação.

- O passo 3 é executado em tempo seqüencial $O(\frac{n}{p})$ e não utiliza comunicação.

Logo, o algoritmo CGM resolve o problema de *range minima*, para o vetor inicial, em tempo $O(\frac{n}{p})$ usando $O(1)$ rodadas de comunicação. $\square$

Processamento de Consultas

Com este algoritmo, uma consulta $MIN(i, j)$ é determinada como se segue.

- Se a_i e a_j estão em um mesmo processador então $MIN(i, j)$ pode ser determinado pelo passo 1 do algoritmo.
- Caso contrário, suponhamos que a_i e a_j estejam em processadores distintos $\bar{i}$ e $\bar{j}$, respectivamente, com $\bar{i} < \bar{j}$.
- Seja $direita(\bar{i})$ o índice em A do elemento mais à direita no vetor $A_{\bar{i}}$, e $esquerda(\bar{j})$ o índice em A do elemento mais à esquerda no vetor $A_{\bar{j}}$. Calcular $MIN(i, direita(\bar{i}))$ e $MIN(esquerda(\bar{j}), j)$, usando o passo 1. Temos dois casos:
  1. Se $\bar{j} = \bar{i} + 1$ então $MIN(i, j) = \min\{MIN(i, direita(\bar{i})), MIN(esquerda(\bar{j}), j)\}$.

2. Se $\bar{i} + 1 < \bar{j}$, então calcular $MIN(direita(\bar{i}) + 1, esquerda(\bar{j}) - 1)$, usando o passo 3 do algoritmo. Notemos que $MIN(direita(\bar{i}) + 1, esquerda(\bar{j}) - 1)$ corresponde a $\min\{b_{\bar{i}+1}, \ldots, b_{\bar{j}-1}\}$. Assim, $MIN(i, j) = \min\{MIN(i, direita(\bar{i})), MIN(direita(\bar{i}) + 1, esquerda(\bar{j}) - 1), MIN(esquerda(\bar{j}), j)\}$.

- O valor de $MIN(direita(\bar{i})+1, esquerda(\bar{j})-1)$ é obtido usando o passo 3 em que cada processador $\bar{i}$:
  - calcula $w = \text{LCA}(b_{\bar{i}+1}, b_{\bar{j}-1})$ em tempo constante, e obtém o nível l_w;
  - determina v e u, os filhos esquerdo e direito, respectivamente, de w;
  - o processador $\bar{i}+1$ calcula $S'[l_w-1]$ e envia este valor para o processador 0;
  - o processador $\bar{j}-1$ calcula $P'[l_w-1]$ e envia este valor para o processador 0;
  - o processador 0 calcula o mínimo dos mínimos recebidos.

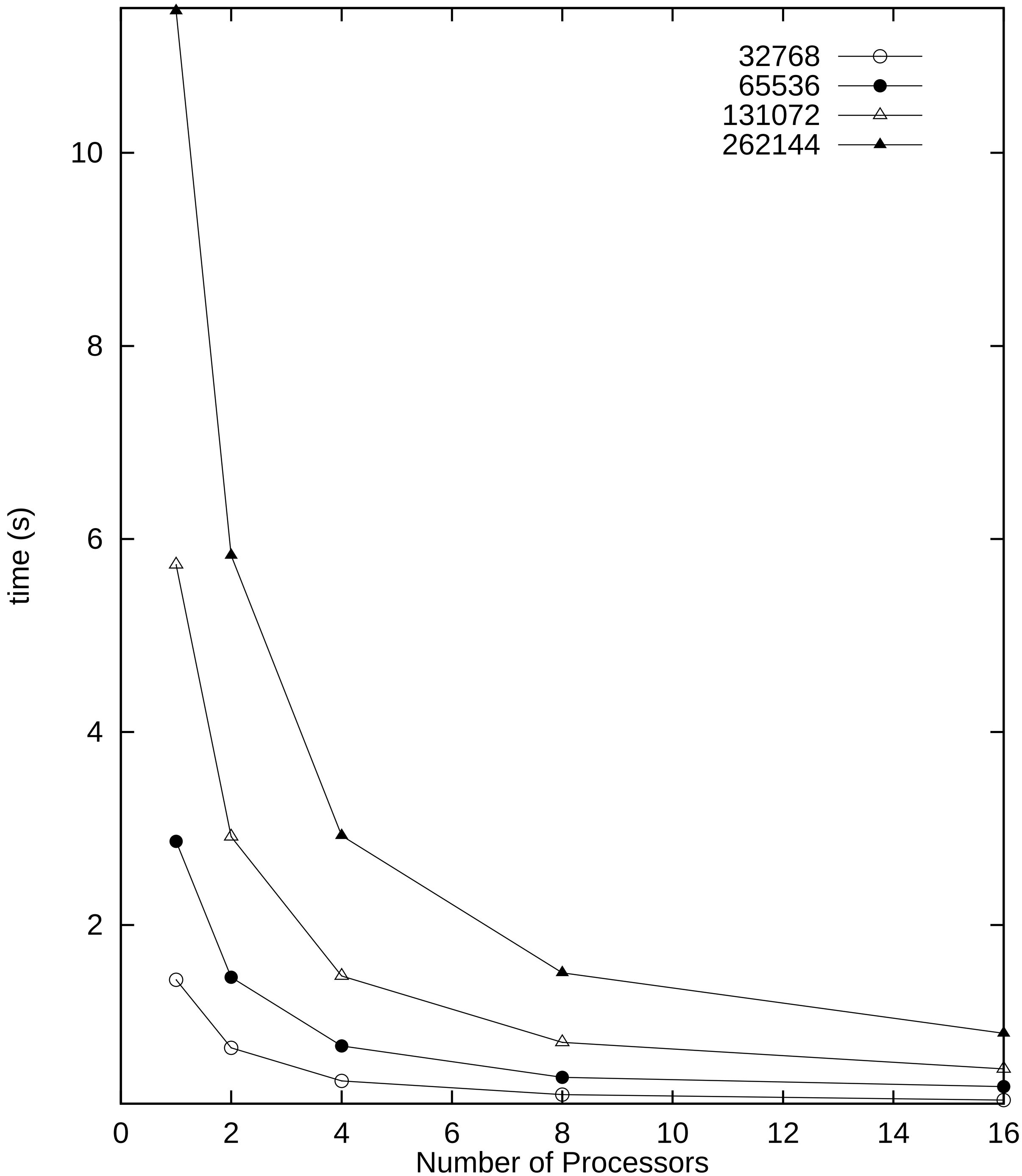


Figura 4: Tempos máximos.

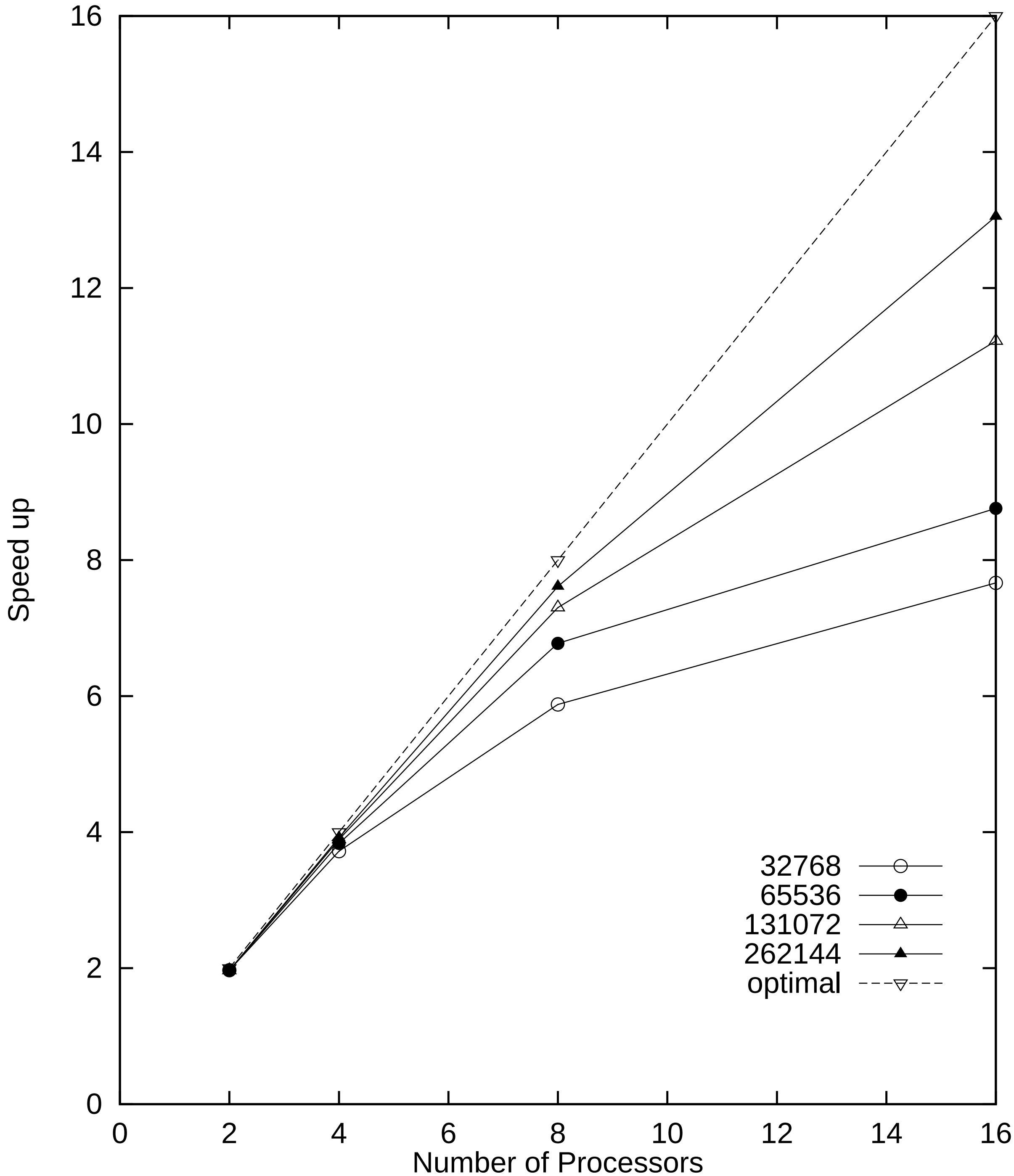


Figura 5: Speed up.