O Crivo de Eratóstenes

Jones Albuquerque
DFM-UFRPE

2004, Recife - PE.

O Algoritmo

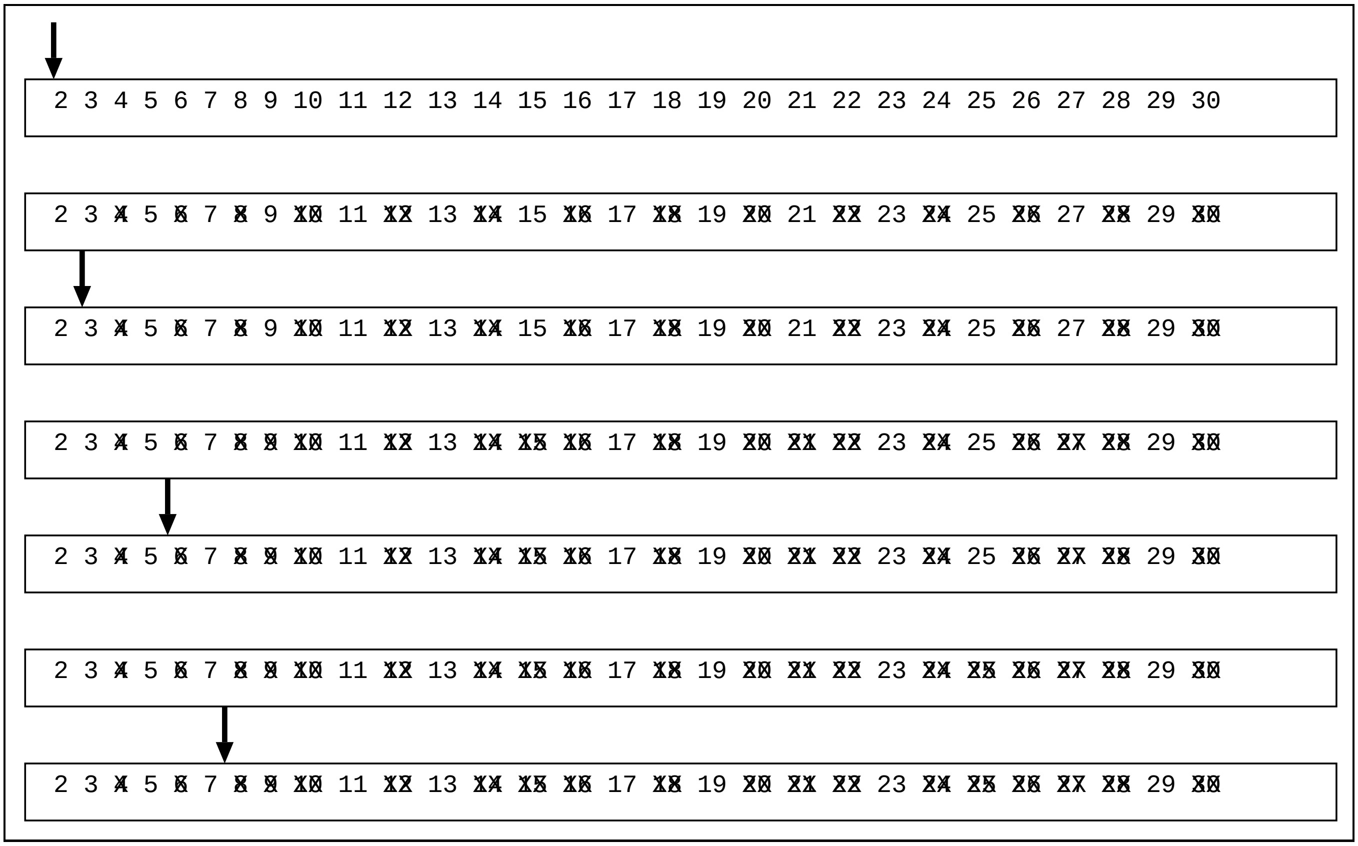
O Crivo de Eratóstenes

- encontra o número de primos menor ou igual a algum inteiro positivo n
- **número primo** é um número que possui exatamente 2 fatores: ele mesmo e 1

Funcionamento

- o crivo começa com uma lista de números naturais $2, 3, 4, \ldots, n$ e remove números compostos da lista descobrindo múltiplos de $2, 3, 5$, e dos demais primos sucessivamente
- o crivo termina quando os múltiplos do maior primo, menor ou igual a $\sqrt{n}$, tenham sido descobertos

Funcionamento - Crivo de Eratóstenes



o crivo começa marcando os últiplos de 2, começando por 2^2; depois marca os múltiplos de 3, começando por 3^2, ...

o crivo termina com os múltiplos de 5, pois $7^2 > 30$

Observações - Crivo de Eratóstenes

O crivo é inviável para teste de primalidade de grandes números (centenas de dígitos), isto porque a complexidade de tempo é $\Omega(n)$, e n cresce exponencialmente com o número de dígitos

Um implementação seqüencial do crivo gerencia três estruturas de dados:

- um vetor boleano cujos elementos correspondem ao números naturais que estão sendo crivados
- um inteiro correspondendo ao último primo encontrado
- um inteiro usado como índice de *loop* incrementado como múltiplos do primo corrente que são marcados como números compostos

Paralelismo de Controle no Crivo

Dois passos:

- encontrar o próximo número primo
- retirar da lista os múltiplos deste primo, começando com o seu quadrado

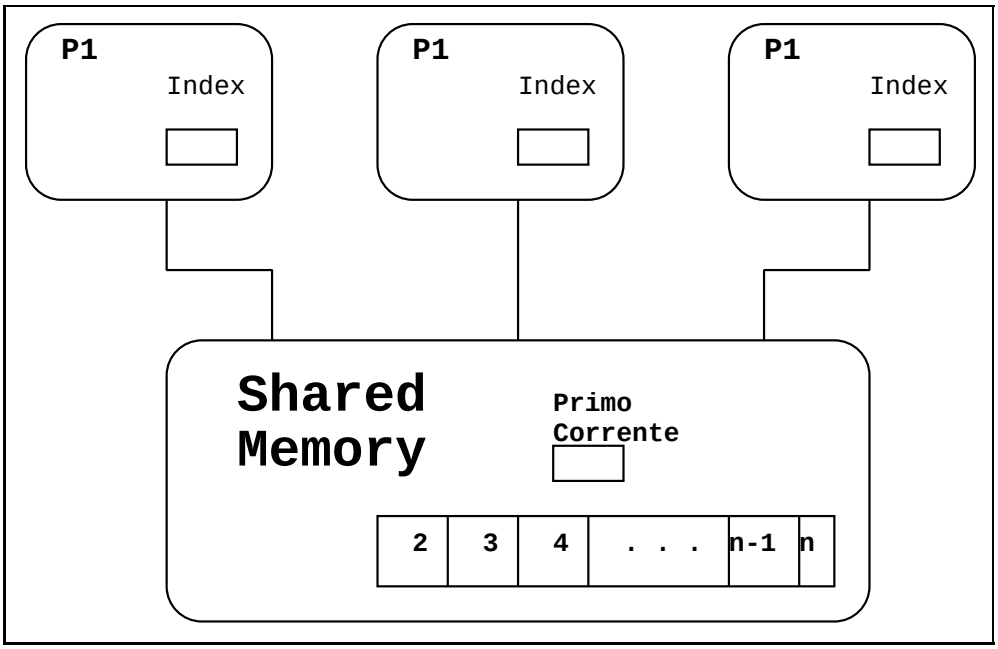
Estratégia:

- processadores diferentes são responsáveis por marcar múltiplos de primos diferentes:
  - processador 1 marca os múltiplos de 2;
  - processador 2 marca os múltiplos de 3;

  . . .

- como?

Memória Compartilhada



Problemas?

Se um grupo de processadores executando assincronamente compartilham acesso à mesma estrutura de dados de uma maneira desistruturada, ineficiências ou erros podem ocorrer

Problema 1 - Mesmo Primo

Pode acontecer de dois processadores usarem o mesmo primo para marcar os seus múltiplos. O algoritmo não executa erradamente, mas desperdiça processamento

Normalmente um processador acessa o valor do primo corrente e começa procurando no vetor até encontrar outra célula não marcada (novo primo)

Se um segundo processador acessa o valor do primo corrente antes do primeiro processador atualizá-lo, então ambos os processadores adotarão o mesmo valor de primo para marcar múltiplos

Problema 2 - Número Composto

Pode acontecer de um processador marcar múltiplos de um número composto (não primo)

Assuma que um processador **A** é responsável por marcar os múltiplos de 2, mas antes dele marcar qualquer célula, um processador **B** encontra o próximo primo 3, e um processador **C** busca pela próxima célula não-marcada

Como a célula 4 ainda não foi marcada, o processador **C** retornará com o valor 4 como o primo mais recente!

Análise de Complexidade

Considerações

Problemas de ineficiência como os descritos não ocorrem

Tempo para encontrar o próximo primo não será considerado

Concentração da análise nas operações de marcar células (múltiplos de primos)

Algoritmo Seqüencial

Assuma 1 unidade de tempo para 1 processador marcar 1 célula

Suponha k primos menores ou iguais a $\sqrt{n}$. Denotaremos estes primos $\pi_1, \pi_2, \pi_3, ..., \pi_k$ ($\pi_1 = 2, \pi_2 = 3, \pi_3 = 5, ..., \pi_k$)

A quantidade total de tempo que um processador gasta marcando seus múltiplos é:

$$\lceil \frac{(n+1)-\pi_1^2}{\pi_1} \rceil + \lceil \frac{(n+1)-\pi_2^2}{\pi_2} \rceil + \lceil \frac{(n+1)-\pi_3^2}{\pi_3} \rceil + \cdots + \lceil \frac{(n+1)-\pi_k^2}{\pi_k} \rceil$$

Por exemplo, para $n = 1000$ a soma dos tempos é 1411

Algoritmo Paralelo

Sempre que um processador está desocupado, ele pega o próximo primo e marca seus múltiplos

Todos os processadores continuam neste procedimento até que o primeiro primo maior que $\sqrt{n}$ seja encontrado

Encontrando todos os primos menores que 1000:

- 2 processadores gastam 706 unidades de tempo;
- 3 processadores gastam 499 unidades de tempo

Observações

A execução para marcar todos os múltiplos de 2 é o limite inferior para complexidade de tempo

O limite superior para speedup é $(1411/499)$ 2,83 para $n = 1000$

Mais de 3 processadores não implica em tempos menores!

n maior $\Rightarrow$ limite superior de speedup maior para mais processadores?

Paralelismo de Dados (Crivo)

Processadores trabalham juntos para marcar múltiplos de cada novo primo encontrado

Cada processador será responsável por um segmento do vetor representando os números naturais até n

Por que paralelismo de dados?

cada processador aplica a mesma operação (marcar múltiplos de um primo) a sua porção do conjunto de dados

Mecanismo de comunicação:

- memória compartilhada
- troca de mensagens

Crivo com Troca de Mensagens

P processadores

Cada processador recebe $\lceil \frac{n}{p} \rceil$ números naturais

$p << \sqrt{n}$

Todos os primos menores que $\sqrt{n}$ assim como o primeiro primo maior que $\sqrt{n}$ estão na lista de números naturais controlada pelo primeiro processador

Funcionamento: processador 1 encontrará o próximo primo e divulgará seu valor para todos os outros processadores (*broadcast*).
Então cada processador marca na sua lista de números compostos todos os múltiplos do novo primo divulgado. Este processo continua até que o primeiro processador encontre um primo maior que $\sqrt{n}$, então o algoritmo termina

Análise de Complexidade

Considerações

Tempo de encontrar o próximo primo é ignorado

Concentração no tempo gasto para marcar números compostos

Consideração do tempo de comunicação para divulgar o primo do primeiro processador para os demais (*broadcast*)

Assuma que são necessárias $\aleph$ unidades de tempo para um processador marcar um múltiplo de um primo como sendo um número composto

Suponha k primos menores ou iguais a $\sqrt{n}$

Análise de Complexidade

Considerações

Denotemos estes primos como $\pi_1, \pi_2, \ldots \pi_k$. O tempo total que um processador gasta para marcar os números compostos é:

$$\left(\lceil \frac{\lceil n/p \rceil}{2} \rceil + \lceil \frac{\lceil n/p \rceil}{3} \rceil + \lceil \frac{\lceil n/p \rceil}{5} \rceil + \cdots + \lceil \frac{\lceil n/p \rceil}{\pi_k} \rceil \right) \aleph$$

Assumimos que o processador 1 encontra o novo primo e comunica aos demais $(p-1)$ processadores um por vez. Se o processador 1 gasta λ unidades de tempo cada vez que ele passa um número para outro processador, então o tempo total de comunicação para todos os k primos é:

$$k(p-1)\lambda$$

Exemplos

Supondo $n = 1000000$

Supondo $\lambda = 100\aleph$ (comunicação = 100 * marcar múltiplos)

Máximo speedup é alcançado com 11 processadores (Figura 1-11 do livro texto)

Tempo de comunicação é o fator de queda de speedup (Figura 1-12 do livro texto)