

# Árvores de Fenwick

Tópicos Especiais em Algoritmos – Ciência da Computação



Prof. Daniel Saad Nogueira  
Nunes

IFB – Instituto Federal de Brasília,  
Campus Taguatinga



# Sumário

---

- 1 Introdução
- 2 Fenwick Trees



# Sumário

---

## 1 Introdução



# Introdução

---

- Problema: dada uma sequência  $V[0, n - 1]$  e índices  $0 \leq i \leq j < n$ , responder:

$$\sum_{k=i}^j V[k]$$



# Algoritmo Força-Bruta

---

- Um algoritmo força-bruta apenas varre o vetor sobre o intervalo considerado e faz a soma.
- Pior caso:  $\Theta(n)$  por consulta.



# Algoritmo Força-Bruta

---

---

**Algorithm 1:** BRUTE-FORCE( $V, i, j$ )

---

**Input:**  $V[0, n - 1], i, j, 0 \leq i \leq j < n$

**Output:**  $\sum_{k=i}^j V[k]$

- 1  $sum \leftarrow 0$
  - 2 **for**(  $k \leftarrow i; k \leq j; k++$  )
  - 3      $sum += V[k]$
  - 4 **return**  $sum$
-



## Buscando Outra Estratégia

---

- É possível melhorar a nossa abordagem.
- Para simplificar: vamos supor que a nossa sequência está armazenada  $V[1, n]$  agora e que  $V[0] = 0$ .
- Vamos computar uma soma de prefixos da seguinte forma:

$$C[i] = \sum_{k=0}^i V[k]$$



## Buscando Outra Estratégia

---

	0	1	2	3	4	5	6	7	8	9	10
$V$	0	1	3	2	5	2	8	7	3	0	4

	0	1	2	3	4	5	6	7	8	9	10
$C$	0	1	4	6	11	13	21	28	31	31	35





## Buscando Outra Estratégia

---

- Como responder  $\sum_{k=i}^j V[k]$ ,  $1 \leq i \leq j \leq n$  utilizando  $C$ ?
- Simples, como  $C[i] = \sum_{k=1}^i V[k]$ , então podemos calcular o somatório do intervalo  $[i, j]$  da seguinte forma:

$$\sum_{k=i}^j = C[j] - C[i - 1]$$

- $\Theta(1)$ .



## Soma de Prefixos

	0	1	2	3	4	5	6	7	8	9	10
V	0	1	3	2	5	2	8	7	3	0	4

$$\sum_{i=3}^7 V[i] = C[7] - C[2] = 28 - 4 = 24$$

	0	1	2	3	4	5	6	7	8	9	10
C	0	1	4	6	11	13	21	28	31	31	35



## Soma de Prefixos

---

- A computação de  $C$  pode ser feita em  $\Theta(n)$  com uma simples varredura da esquerda para a direita em  $V$ .



## Soma de Prefixos

---

---

### Algorithm 2: PREFIX-SUM( $V, C$ )

---

**Input:**  $V[0, n], V[0] = 0$

**Output:**  $C[0, n]$

- 1  $C[0] \leftarrow 0$
  - 2 **for** ( $i \leftarrow 1; i \leq n; i++$ )
  - 3      $C[i] \leftarrow V[i] + C[i - 1]$
-



## Soma de Prefixos

---

- Com a soma de prefixos, podemos responder qualquer pergunta em tempo constante após calcular  $C$ .
- Tempo de pré-processamento:  $\Theta(n)$ .
- Tempo por consulta:  $\Theta(1)$ .
- $\langle \Theta(n), \Theta(1) \rangle$ .



## Soma de Prefixos

---

- A estratégia utilizando soma de prefixos é excelente!
- Mas existe um problema, só funciona no caso estático, em que  $V$  não muda.
- No caso dinâmico, temos que recomputar  $C$  a cada mudança.
- Levamos  $\Theta(n)$  para atualizar  $C$  no pior caso.
- Precisamos de uma estrutura que funciona bem para o caso dinâmico.
- Alternativa: **Fenwick-Trees**. Tempo  $\langle \Theta(n), \Theta(\lg n) \rangle$  permitindo atualizações em tempo  $\Theta(\lg n)$ .



# Sumário

---

## 2 Fenwick Trees



# Fenwick Trees

---

- As Fenwick Trees foram propostas por Peter Fenwick na década de 90.
- São baseadas em um truque utilizando aritmética computacional e representação binária.
- Na prática conseguem responder as consultas de soma sobre intervalos de uma maneira bem rápida.
- Vamos definir o objeto de cálculo.





# Fenwick Trees

---

- Seja  $lsb(i)$  a posição do bit 1 menos significativo de  $i$  em sua representação binária.
  - ▶  $lsb(10) = lsb(1010_2) = 1$
  - ▶  $lsb(3) = lsb(11_2) = 0$
  - ▶  $lsb(16) = lsb(10000_2) = 4$
- A Fenwick Tree  $FT$  é um vetor  $FT[0, n]$  em que

$$FT[i] = \sum_{k=i-2^{lsb(i)}+1}^i V[k]$$



# Fenwick Trees

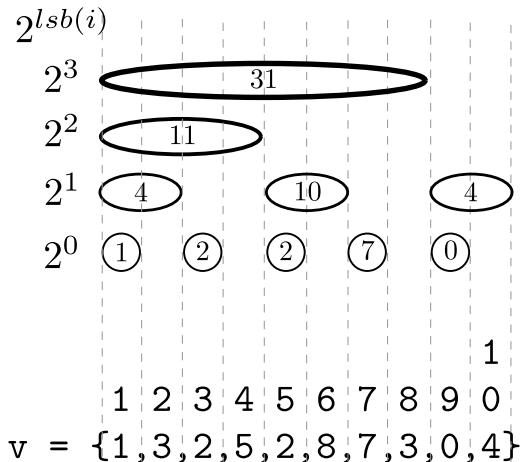
	0	1	2	3	4	5	6	7	8	9	10
$V$	0	1	3	2	5	2	8	7	3	0	4

	0	1	2	3	4	5	6	7	8	9	10
$C$	0	1	4	6	11	13	21	28	31	31	35

	0	1	2	3	4	5	6	7	8	9	10
$FT$	0	1	4	2	11	2	10	7	31	0	4
Intervalos		[1, 1]	[1, 2]	[3, 3]	[1, 4]	[5, 5]	[5, 6]	[7, 7]	[1, 8]	[9, 9]	[9, 10]



# Fenwick Trees





# Sumário

---

- 2 Fenwick Trees
  - Construção
  - Consultas
  - Atualização
  - Análise



## Construção de Fenwick Trees

---

- Para construir a Fenwick Tree, podemos utilizar o vetor  $C$  de soma de prefixos para computar cada intervalo considerado por cada nó.
- Como os inteiros normalmente estão representados em binário através do complemento de dois, o valor de  $2^{lsb(i)}$  pode ser computado de uma maneira muito eficiente.



## Construção de Fenwick Trees

---

- Tome  $x = a1b$ , em que  $a$  representa os bits mais significativos de  $x$ ,  $1$  representa o bit 1 mais à direita de  $x$ , e  $b$  uma sequência de zeros ao final de  $x$ .
- Sabemos que, pela representação computacional usando complemento de dois, temos:  $-x = \sim x + 1$ .
- Como  $\sim(x) = \sim(a1b) = (\sim a)0(\sim b)$ , temos que  $\sim(a1b) + 1 = (\sim a)1b$ .
- Usando o operador de & bit a bit na expressão  $(x \& -x)$  temos exatamente o valor de  $2^{lsb(x)}$ .



## Construção das Fenwick Trees

---

$$\begin{array}{cccc}
 & a & 1 & b \\
 & \sim a & 1 & b \\
 \& \hline
 & 00 \dots 0100 \dots 0
 \end{array}$$



## Construção de Fenwick Trees

---

---

### Algorithm 3: BUILD-FT

---

**Input:**  $C[0, n]$

**Output:**  $FT[0, n]$

- 1  $FT[0] \leftarrow 0$
  - 2 **for**(  $i \leftarrow 1; i \leq n; i++$  )
  - 3    $FT[i] \leftarrow C[i] - C[i - (i \& - i)]$
-





# Fenwick Trees

---

## Construção

- A Fenwick Tree pode ser construída a partir da soma de prefixos  $C$  em tempo  $\Theta(n)$ , uma vez que a soma sobre cada intervalo é respondida em tempo  $\Theta(1)$ .



# Sumário

---

## 2 Fenwick Trees

- Construção
- Consultas
- Atualização
- Análise



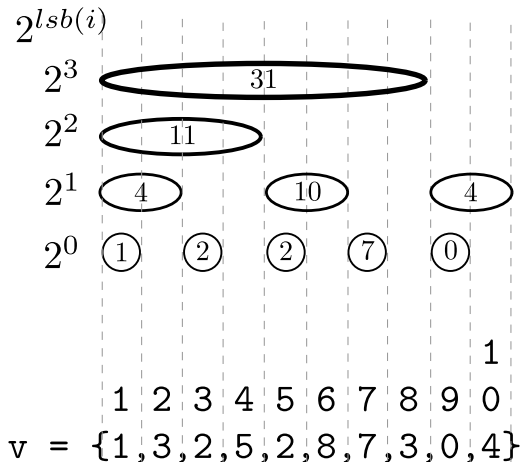
## Consultas sobre Fenwick Trees

---

- Mostraremos agora como utilizar a Fenwick Tree para responder a consulta de soma sobre o intervalo  $V[1, i]$  para algum  $1 \leq i \leq n$ .
- O nó  $FT[i]$  contém a informação da soma de  $V[i - 2^{lsb(i)} + 1, i]$ .
- Seja  $i' = i - 2^{lsb(i)}$ . Recursivamente podemos conseguir mais um pedaço da informação de  $V[1, i]$  olhando para  $FT[i']$ , que possui o valor da soma de  $V[i' - 2^{lsb(i')} + 1, i']$ .
- Caso base:  $i' = 0$ .



# Fenwick Trees





## Construção de Fenwick Trees

---

---

### Algorithm 4: SUM

---

**Input:**  $FT[0, n], i$

**Output:**  $\sum_{k=1}^i V[k]$

- 1  $sum \leftarrow 0$
  - 2 **while**  $i > 0$  **do**
  - 3      $sum+ = FT[i]$
  - 4      $i- = (i \& - i)$
  - 5 **return**  $sum$
-



## Consultas sobre Fenwick Trees

---

- Utilizando a mesma ideia da soma de prefixos, é possível utilizar a Fenwick Tree para responder uma soma sobre um intervalo  $V[i, j]$ .
- Basta considerar subtração das somas sobre  $V[1, j]$  e  $V[1, i - 1]$ .



## Consultas de Fenwick Trees

---

---

**Algorithm 5:** SUM

---

**Input:**  $FT[0, n], i, j$

**Output:**  $\sum_{k=i}^j V[k]$

1 **return**  $FT.SUM(j) - FT.SUM(i - 1)$

---



## Consultas de Fenwick Trees

---

- O tempo para cada consulta sobre uma Fenwick Tree é  $\Theta(\lg n)$ .
- Cada operação  $i- = (i \& x - i)$  efetivamente escreve 0 no bit menos significativo com valor 1 de  $i$ .
- Como gastamos  $w$  bits para representar um inteiro a cota está justificada, pois  $w \in \Theta(\lg n)$ . (Modelo RAM)





# Sumário

---

- 2 Fenwick Trees
  - Construção
  - Consultas
  - Atualização
  - Análise



## Atualização de Valores

---

- Até o momento não ganhamos nada com as Fenwick Trees.
- Utilizar soma de prefixos para responder uma consulta leva tempo  $\Theta(1)$  contra  $\Theta(\lg n)$  da Fenwick Tree.
- A grande utilidade das Fenwick Trees é a habilidade de poder atualizar valores em tempo  $\Theta(\lg n)$ .



## Atualização de Valores

---

- Suponha que queremos atualizar  $V[i]$ .
- As mudanças devem ser propagadas nos nós da Fenwick Tree.
- Sabemos que um nó  $FT[j]$  da Fenwick Tree cobre o intervalo  $[j - 2^{lsb(j)} + 1, j]$ . Temos que atualizar todos os nós  $FT[j]$  cujo intervalo contém  $i$ . Em outras palavras, atualizamos  $FT[j]$  sempre que  $j - 2^{lsb(j)} + 1 \leq i \leq j$ .
- Processo simétrico ao da soma, mas incrementamos  $i$  de  $2^{lsb(i)} = (i \& -i)$  unidades.



## Atualização de Valores

---

---

### Algorithm 6: UPDATE

---

**Input:**  $FT[0, n], i, \Delta$

- 1 **while**  $i \leq n$  **do**
  - 2      $FT[i]_+ = \Delta$
  - 3      $i_+ = (i \& - i)$
-



## Atualização de valores

---

- As atualizações podem ser feitas em tempo  $\Theta(\lg n)$ .



# Sumário

---

## 2 Fenwick Trees

- Construção
- Consultas
- Atualização
- Análise



## Análise das Fenwick Trees

---

- As Fenwick Trees são uma excelente opção para responder a consulta de soma sobre intervalos em tempo  $\langle \Theta(n), \Theta(\lg n) \rangle$ .
- Estrutura dinâmica, suporta atualização em tempo  $\Theta(\lg n)$ .
- Implementação extremamente enxuta e elegante baseada em aritmética computacional.
- Muito rápida na prática.