Giới thiệu:

* Được xuất hiện lần đầu vào năm 1994 bởi Peter M. Fenwick, Binary Indexing Tree (BIT) hay Fenwick Tree được sử dụng để ứng dụng vào thuật toán arithmetic coding.

(Note: arithmetic coding là một kĩ thuật nén dữ liệu thành một chuỗi dữ liệu bằng cách tạo ra một chuỗi mã đại diện cho một phân số trong khoảng từ 0 đến 1).

* Dù đã bị nhiều thuật toán tốt hơn thay thế nhưng thuật toán Fenwick Tree vẫn được sử dụng trong lập trình thi đấu vì khả năng sử dụng ít bộ nhớ mà code không phức tạp.

Nội dung chính:

* Định nghĩa: Fenwick Tree là một cách để biểu diễn một mảng các dãy số dưới dạng mảng các prefix sum.
* Đặt ra bài toán: Cho n phần tử dùng để thực hiện 2 truy vấn:

+ Tính tổng giá trị từ phần tử thứ m đến thứ n.

+ Thêm giá trị vào phần tử thứ i.

* Ví dụ:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| A= | 1 | 3 | 4 | 10 | -1 | 2 | 9 | -7 | 8 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R= | 1 | 4 | 8 | 18 | 17 | 19 | 28 | 21 | 29 | 30 |

Khi này tính sum(i, j) = R[j] – R[i-1]

* Giả sử tính sum(2, 4) = R[4] – R[1] = 17 – 4 = 13.

Quá đơn giản.

Nhưng với câu truy vấn thứ hai: nếu thêm 2 vào phần tử thứ 3 hay add(2, 3) thì sẽ khiến cho toàn bộ giá trị trong mảng từ R[3] trở đi sẽ phải thay đổi.

* Do đó người ta thực hiện chia mảng ra thành nhiều mảng và sử dụng ứng dụng của dãy bit để lưu index của mảng.

Ví dụ:

Khi này: sum(1, 7) = range(1, 4) + range(5, 6) + range(7,7)

(từ đoạn này Fenwick Tree sẽ đánh số index bắt đầu từ 1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | -1 | 2 | 9 | -7 | 8 | 1 |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 4 |  | 18 |  |  |  | 21 |  |  |
| [1,1] | [1,2] |  | [1,4] |  |  |  | [1,8] |  |  |
|  |  | 4 |  | -1 | 1 |  |  | 8 | 9 |
|  |  | [3,3] |  | [5,5] | [5,6] |  |  | [9,9] | [9,10] |
|  |  |  |  |  |  | 9 |  |  |  |
|  |  |  |  |  |  | [7,7] |  |  |  |

Như vậy sum(1, 7) = 18 + 1 + 9 = 28 => đúng như đã làm ở trên.

Và quy đổi sang bit ta có:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | -1 | 2 | 9 | -7 | 8 | 1 |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 4 |  | 18 |  |  |  | 21 |  |  |
| 0001 | 0010 |  | 0100 |  |  |  | 1000 |  |  |
|  |  | 4 |  | -1 | 1 |  |  | 8 | 9 |
|  |  | 0011 |  | 0101 | 0110 |  |  | 1001 | 1010 |
|  |  |  |  |  |  | 9 |  |  |  |
|  |  |  |  |  |  | 0111 |  |  |  |

Ở đây có một kĩ thuật được gọi là số bit cuối (last set bit) được sử dụng để di chuyển giữa các bit: x & (-x)

Ví dụ: x = 7 = (0111)2

-x = -7 = (1001)2

x & (-x) = (0001)2

* Xóa bit cuối: x – (x & (-x))
* Thêm bit cuối: x + (x & (-x))

Trở lại với câu truy vấn thứ 2:

Giả sử t thực hiện: add(2, 5)

(Thêm 2 vào phần tử thứ 5)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 10 | 1 | 2 | 9 | -7 | 8 | 1 |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 1 | 4 |  | 18 |  |  |  | 24 |  |  |
| [1,1] | [1,2] |  | [1,4] |  |  |  | [1,8] |  |  |
|  |  | 4 |  | 1 | 3 |  |  | 8 | 9 |
|  |  | [3,3] |  | [5,5] | [5,6] |  |  | [9,9] | [9,10] |
|  |  |  |  |  |  | 9 |  |  |  |
|  |  |  |  |  |  | [7,7] |  |  |  |

Như vậy là đã thực hiện câu truy vấn xong.