Prefix Sum

Debojoti Das Soumya

14 January 2022

সূচীপত্র

1 পরিচিতি

ধর তোমাকে একটি প্রোগ্রাম লিখতে হবে যেখানে একটি array $[a_1, a_2, a_3, \cdots, a_{n-1}, a_n]$ দেওয়া হবে এবং তোমাকে q টি প্রশ্নের উত্তর দিতে হবে। প্রতিটি প্রশ্নে একটি pair (l, r) দেওয়া হবে এবং $a_l + a_{l+1} + a_{l+2} + \cdots + a_{r-1} + a_r$ আউটপুট দিতে হবে।

এই প্রব্লেমের সবচেয়ে সহজ solution হবে প্রতি প্রশ্নের জন্য একবার লুপ দিয়ে যোগফলটি বের করা। কিন্তু worst case এ এই প্রোগ্রামের complexity হবে $O(nq)$। যখন $1 \leq n, q \leq 10^5$ এই solution pass করবে না। তাই আমাদের কে প্রোগ্রামেটিকে optimize করতে হবে। এই optimization এর জন্য আমরা prefix sum ব্যবহার করব। প্রথমে আমরা নতুন আরেক্টি array p তৈরি করব এমনভাবে যেনঃ

$$
\begin{aligned}
p_0 &= 0 \\
p_1 &= a_1 \\
p_2 &= a_1 + a_2 \\
&\vdots \\
p_n &= a_1 + a_2 + \cdots + a_{n-1} + a_n
\end{aligned}
$$

অর্থাৎ $p_i = a_1 + a_2 + \cdots + a_{i-1} + a_i$। অনেকে হয়ত ভাবছ এইটা বের করতেই তো $O(n^2)$ লাগবে। কিন্তু খেয়াল কর $p_i = p_{i-1} + a_i$ কারণঃ

$$
\begin{aligned}
p_{i-1} &= a_1 + a_2 + \cdots + a_{i-2} + a_{i-1} \\
p_i &= a_1 + a_2 + \cdots + a_{i-1} + a_i = (a_1 + a_2 + \cdots + a_{i-1}) + a_i = p_{i-1} + a_i
\end{aligned}
$$

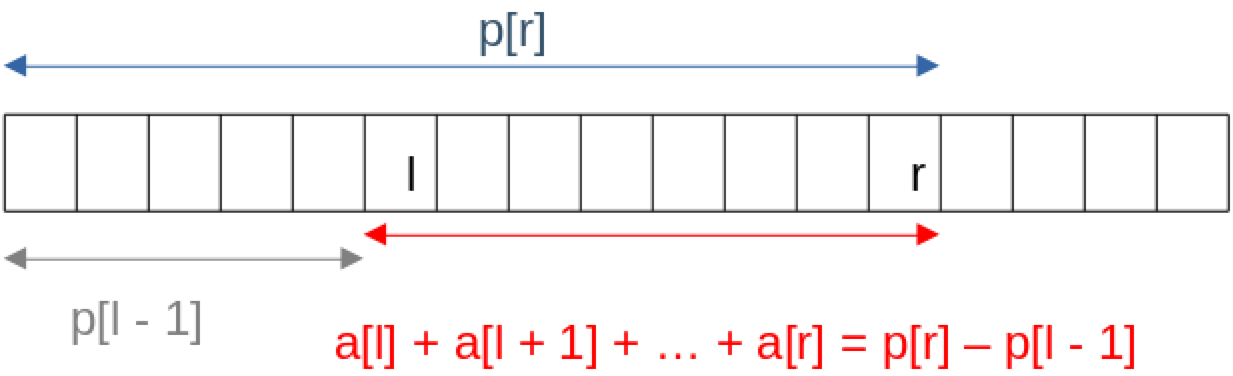
এখন আমরা p array টা $O(n)$ এ তৈরি করতে পারি। নিচে একটি pseudocode দেওয়া হলোঃ

```
// array a uses 1-based indexing
int p[n + 1]
p[0] = 0
for (int i = 1; i <= n; i++) {
  p[i] = p[i - 1] + a[i];
}
```

এখন p array ব্যবহার করে একটি sub-array এর যোগফল কিভাবে বের করা যায় তা একটু ভেবে দেখ তো।

$$
\begin{aligned}
\sum_{i=l}^{r} a_i &= a_l + a_{l+1} + \cdots + a_{r-1} + a_r \\
&= (a_1 + a_2 + \cdots + a_{r-1} + a_r) - (a_1 + a_2 + \cdots + a_{l-2} + a_{l-1}) \\
&= p_r - p_{l-1}
\end{aligned}
$$

নিচের ছবির মাধ্যমেও বিষয়টি বুঝা যায়।



Prefix sum technique টি আমরা যেকোন reversible operation এ ব্যবহার করতে পারব। এমন operation এর মধ্যে উল্লেখযোগ্য গুন ও bitwise xor। Sub-array এর গুণফল দরকার হলে আমরা prefix এর গুনফল store করবো আর query এর উত্তর হবে $\frac{p_r}{p_{l-1}}$ আর sub-array এর bitwise xor দরকার হলে আমরা prefix এর bitwise xor store করবো আর query হবে $p_r \oplus p_{l-1}$ এখানে $\oplus$ অর্থ bitwise xor।

2 Range add

আরেকটা সমস্যা দিয়ে শুরু করা যাক। আমাদের কাছে একটি array a ছিল যার length n এবং শুরুতে সব $a_i = 0, (1 \leq i \leq n)$। আমাদেরকে এই array এর উপর q টি modification করতে হবে। প্রতি modification এ একটি triplet (l, r, v) দেওয়া হবে এবং আমাদেরকে প্রতি $i (l \leq i \leq r)$ এর জন্য $a_i := a_i + v$ করতে হবে। সব modification শেষে আমাদেরকে array a print করতে হবে।

আমরা শুরুতে একটি array a তৈরি করব যার length n এবং শুরুতে সব $a_i = 0, (1 \leq i \leq n)$। এবার প্রতি modification এ যদি আমরা $a_l := a_l + v$ এবং $a_{r+1} := a_{r+1} - v$ করি এবং শেষে এই array এর prefix sum array p তৈরি করি, তাহলে এই p array টি হবে আমাদের final answer।

অনেকের মাথায় প্রশ্ন আসবে যে এটি কেন কাজ করছে। আমরা যদি শুধু একটি modification চিন্তা করি তাহলে prefix sum নেওয়ার পরে দেখা যাবে যে প্রতি $i (1 \leq i < l)$ এর জন্য কোনো সংখ্যা যোগ হয়নি। এবং প্রতি $i (l \leq i \leq r)$ এর জন্য v যোগ হয়েছে আর শেষে প্রতি $i (r + 1 \leq i \leq n)$ এর জন্য কোনো কিছু যোগ হয়নি। আবারো ছবির মাধ্যমে বিষয়টি visualize করা যাক।

Before doing anything

0	0	0	0	0	0	0	0	0	0	0	0	0

After applying the modification (l = 4, r = 9, v = 3)

0	0	0	3	0	0	0	0	0	-3	0	0	0

After taking prefix sum

0	0	0	3	3	3	3	3	3	0	0	0	0

এভাবে প্রতি modification কে আলাদা ভাবে খেয়াল করলে দেখা যাবে একটি modification এর জন্য শুধু ওই নির্ধিষ্ট range এই যোগ হচ্ছে। তাই এটি কাজ করছে।

3 সমস্যা সমূহ

3.1 Number of 0 sum sub-arrays

আগের সমস্যার মতো আমাদের আবার একটি array a দেওয়া হবে যার মধ্যে n টি সংখ্যা আছে। আমাদের আউটপুট দিতে হবে কয়টি (l, r) আছে যেন $1 \leq l \leq r \leq n$ এবং $a_l + a_{l+1} + \cdots + a_{r-1} + a_r = 0$।

আবার আমরা শুরুতে p array টি তৈরি করব। এখন আমাদের বলতে হবে কয়টি (l, r) আছে যেন $1 \leq l \leq r \leq n$ এবং $p_r - p_{l-1} = 0$। এই equation টিকে আমরা সাজিয়ে এভাবে লিখতে পারি $p_r = p_{l-1}$।

আমরা যদি প্রতিটি r এর জন্য কয়টি l আছে তা বের করতে পারি তাহলে প্রতিটি r এর জন্য কয়টি l আছে তার যোগফলই হচ্ছে আমাদের সমস্যার সমাধান। খেয়াল করি $1 \leq l \leq r$ অর্থ $0 \leq l - 1 \leq r - 1$। আমাদের বের করতে হবে কয়টি $l - 1$ আছে যেন $0 \leq l - 1 \leq r - 1$ এবং $p_r = p_{l-1}$। আমরা যদি $p - 1$ কে i দ্বারা বদলে দেই তাহলে condition গুলো হলো $0 \leq i \leq r - 1$ এবং $p_r = p_i$। এবং এটি আমরা খুব সহজেই *map* ব্যবহার করে $O(n \log n)$ এ বের করে ফেলতে পারি। নিচে pseudocode টি দেওয়া হলোঃ

```
int ans = 0;
map<int, int> mp;
mp[0]++; // add 1 to the occurrence of p[0]
for (int r = 1; r <= n; r++) {
  // mp[x] currently stores the number of times x has occurred upto r - 1
  ans += mp[p[r]]; // as mp[p[r]] stores the number of times p[r] has occurred in
  // p[0...r - 1]. we add mp[p[r]] it to the answer
  mp[p[r]]++; // we add 1 to the occurrence of p[r]
}
// number of pairs is stored in ans
```

3.2 Minimum sub-array sum

আবারো আমাদেরকে একটি array a দেওয়া হবে যার length n। এবার আমদের বলতে হবেঃ

$$\min_{1 \le l \le r \le n} \sum_{i=l}^{r} a_i$$

অর্থাৎ সব sub-array এর মধ্যে সবচেয়ে ছোট sub-array sum।

আবার আমরা r fix করব এবং বের করব r এ শেষ হওয়া minimum sub-array sum। এভাবে সব r এর জন্য পাওয়া সংখ্যা গুলার মধ্যে সবচেয়ে ছোট সংখ্যাটি হবে আমাদের উত্তর। তাহলে আমাদের এমন $l-1$ দরকার যেন $p_r - p_{l-1}$ minimum হয়। নির্দিষ্ট r এর জন্য, এখানে p_r constant, তাই p_{l-1} যত বড় হবে $p_r - p_{l-1}$ ততো ছোট হবে। তাই প্রতিটি r এর জন্য আমাদের প্রয়োজন $\max_{0 \le i \le r-1} p_i$। এটাও আমরা আগের মতো $O(n)$ এ বের করতে পারি। আবারো pseudocode টি নিচে দেওয়া হলোঃ

```
int mx = 0; // initializing max prefix sum with p[0]
int ans = INT_MAX; // initializing minimum sub-array sum to be infinity
for (int r = 1; r <= n; r++) {
  ans = min(ans, p[r] - mx); // updating ans with the minimum sub-array sum ending
  // at r
  mx = max(mx, p[r]); // updating max prefix sum with p[r]
}
// minimum sub-array sum is stored in ans
```

3.3 আরো সমস্যা

1. একটি array তে এমন কয়টি sub-array আছে যাদের sum k। (k দেওয়া থাকবে)?

2. একটি array দেওয়া হবে যার length n এমন কয়টি (l, r) আছে যেন $1 \le l \le r \le n$ এবং

$$\sum_{i=l}^{r} a_i = r - l + 1$$

3. একটি array a দেওয়া হবে যার length n। এবার আমদের বলতে হবে

$$\min_{1 \le l \le r \le n} a_l \oplus a_{l+1} \oplus \cdots \oplus a_{r-1} \oplus a_r$$

এখানে $\oplus$ অর্থ bitwise xor। Hint: trie data structure