Good subsegments

Xét một tập $S = \{x_1, x_2, ..., x_k\}$, tính P_S:

$$P_S = \sum_{i=1}^{k} 2^{x_i}$$

ta thấy rằng tập S này có thể ghép lại thành một số duy nhất nếu P_S là lũy thừa của 2, hay $P_S == 2^u$. Dựa vào tính chất này ta áp dụng cho bài toán như sau:

- tính $tp[i] = 2^{a[i]}$
- bài toán chuyển về đếm số đoạn $[l, r]$ sao cho

$$\sum_{i=l}^{r} tp[i] == 2^u$$

Gọi $p = \max(a[i]), 1 \leq i \leq n$. Thì ta có thêm nhận xét rằng

$$2^p \leq \sum_{i=1}^{n} tp[i] < 2^{p+\log n+1} \quad (1)$$

Với subtask $a[i] \leq 30$ tổng của tất cả $tp[i]$ sẽ không vượt quá 2^{50}, và do đó ta có thể duyệt lần lượt từng giá trị $E = 2^u$ $(1 \leq u \leq 50)$ và đếm số đoạn $[l, r]$ có $\sum_{i=l}^{r} tp[i] == E$. Như vậy với thuật toán này sẽ chạy trong $O(50 \times n)$ nếu ta sử dụng hash table (unordered_map) để đếm số lượng đoạn $[l, r]$ có tổng bằng E.

Với subtask $a[i] \leq 10^9$, ta sử dụng hàm băm để có thể vẫn sử dụng tính chất trên, để xác suất bị băm trùng là thấp nhất ta nên dùng giá trị hash là một số nguyên tố càng lớn càng tốt. Với giá trị long long ta có thể sử dụng một số nguyên tố có 18 chữ số để làm giá trị hash. Khi tính tích của hai số A, B mod cho $H < 10^{18}$ ta tính như sau:

```
long long mult(long long A, long long B) {
    long long k = (long long)((long double) A * B / MOD);
    long long r = A * B - MOD * k;
    if (r < 0) {
        r += MOD;
    }
    if (r >= MOD) {
        r -= MOD;
    }
    return r;
}
```

Ta định nghĩa lại như sau:

- Gọi H là giá trị hash được chọn
- tính $tp[i] = 2^{a[i]} \mod H$
- bài toán chuyển về đếm số đoạn $[l, r]$ sao cho

$$\sum_{i=l}^{r} tp[i] \mod H == 2^u \mod H$$

Giờ đầu tiên ta cần tìm cách kiểm tra một đoạn $[l, r]$ thỏa mãn, làm sao để biết được tổng của các $tp[i]$ trong đoạn này là lũy thừa của 2? Dựa vào tính chất (1) ta có thể kiểm tra được điều này.

- Ta tính $p = \max_{i=l}^{r} a[i]$
- tính $P_{[l,r]} = \sum_{i=l}^{r} tp[i] \mod H$
- cuối cùng ta chỉ cần duyệt u từ p đến $p + \log n$ và kiểm tra $P_{[l,r]} == 2^u \mod H$

Như vậy ta có thể duyệt từng đoạn $[l, r]$ và dùng cách trên để kiểm tra, thuật toán sẽ chạy trong $O(n^2 \log n)$.

Tất nhiên với độ phức tạp trên ta sẽ không thể AC được bài toán, do đó ta cần có thuật toán khác tốt hơn. Với những bài toán đếm số lượng đoạn liên tiếp thỏa mãn một tính chất nào đó, ta có thể nghĩ đến việc sử dụng chia để trị để giải quyết bài toán đó. Bài toán này ta cũng có thể áp dụng chia để trị vào giải quyết.

- Giả sử ta xét hai đoạn $[l, mid]$ và $[mid+1, r]$, ta đếm số đoạn $[i, j]$ với $l \le i \le mid$ và $mid+1 \le j \le r$ thỏa mãn tổng $tp[z]$ trong đoạn $[i, j]$ là lũy thừa của 2.
- Tính $sl[i]$ là tổng của các $tp[z]$ từ i đế mid hay:

$$sl[i] = \sum_{z=i}^{mid} tp[z] = sl[i+1] + tp[i], l \le i \le mid$$

- Tính $sr[i]$ là tổng của các $tp[z]$ từ $mid + 1$ đến j hay:

$$sr[j] = \sum_{z=mid+1}^{j} tp[z] = sr[j-1] + tp[j], mid + 1 \le j \le r$$

- Nếu với bài toán đếm số đoạn $[i, j]$ với $l \le i \le mid$ và $mid + 1 \le j \le r$ thỏa mãn tổng $tp[z] == E$ thì rất đơn giản ta chỉ cần làm như sau:
  - Cho tất cả các tổng $sr[j]$ với $mid + 1 \le j \le r$ vào một unordered_map cnt ($cnt[x]$ là số lượng giá trị bằng x của các $sr[j]$)
  - Duyệt lần lượt i ($l \le i \le mid$), ta cần đếm số lượng j thỏa mãn $sl[i] + sr[j] == E$, do đó số lượng j thỏa mãn sẽ là $cnt[E - sl[i]]$.
- Tuy nhiên với bài toán này giá trị E không cố định mà bị phụ thuộc vào chính đoạn $[i, j]$ đang xét. Giá trị E được xác định bằng 2^p với $p = \max_{x=i}^{j} a[z]$. Giả sử ta xét đến i trong đoạn $[l, mid]$, ta đi đếm số giá trị j trong đoạn $[mid + 1, r]$:
  - gọi $ml[i] = \max_{z=i}^{mid} a[z]$
  - gọi $mr[j] = \max_{z=mid+1}^{j} a[z]$
  - với mỗi i ta tìm jr lớn nhất sao cho $ml[i] \ge mr[jr]$, như vậy nếu ta chỉ tìm j trong đoạn $[mid + 1, jr]$ thì giá trị p là không đổi và bằng $ml[i]$
  - tuy nhiên nếu chỉ đếm j trong đoạn $[mid + 1, jr]$ thì sẽ bị thiếu các giá trị j trong đoạn $[jr + 1, r]$.
  - để giải quyết vấn đề này ta sẽ cần tính thêm một lần nữa, ta duyệt các giá trị j và đếm số giá trị i trong đoạn $[il, mid]$ với il là giá trị nhỏ nhất mà $ml[il] < mr[j]$.
  - với các cặp (i, j) trong 2 lần đếm khi cố định i đếm j, và cố định j đếm i có bị trùng nhau không? Ta thấy rằng với lần tính đầu tiên giả sử ta tìm được cặp (i_1, j_1), nếu lần thứ hai ta cũng tìm được cặp (i_1, j_1) này, thì rõ ràng i_1 và j_1 phải cùng thỏa mãn $ml[i_1] \ge mr[j_1]$ $ml[i_1] < mr[j_1]$, điều này là không thể. Do đó với hai lần tính như trên ta sẽ không đếm bị lặp lại các cặp (i, j).

- Như vậy ta đã có thể đếm được số lượng đoạn $[i, j]$ với $l \leq i \leq mid$ và $mid+1 \leq j \leq r$ trong thời gian $O((r-l)\log n)$ ($\log n$ là vì với mỗi giá trị i ta cần đếm j với các giá trị $E = 2^u,\ \forall p \leq u \leq p+\log n$). Do đó độ phức tạp của cả thuật toán là $O(n \log^2 n)$, để thời gian chạy là đủ tốt có thể sẽ cần tối ưu thêm với truy vấn $cnt[x]$, nếu dùng unordered_map có thể thời gian truy vấn vẫn là rất lớn.