LEGGO

Nguồn: 2015-2016 ACM-ICPC Northeastern European Regional Contest - Problem L

https://codeforces.com/gym/100851/problem/L

Trước hết, ta sẽ chặt nhị phân đáp án. Với một độ cao x bất kì, gọi $f(x)$ là số miếng ghép cần đặt để có một cột có độ cao ít nhất bằng x. Nếu $f(x) \leq k$, đáp án cần tìm sẽ lớn hơn x. Ngược lại, đáp án sẽ nhỏ hơn x. Vấn đề còn lại là tính $f(x)$ trong độ phức tạp thời gian cho phép.

Giả sử một cột i nào đó cần đạt độ cao ít nhất bằng x. Khi đó, các cột $i-1$, $i-2$, ... cần có độ cao ít nhất lần lượt là $x-1$, $x-2$, ...; và các cột $i+1$, $i+2$, ... cần có độ cao ít nhất lần lượt là $x-1$, $x-2$. Tuy nhiên, trong quá trình đi về bên trái cột i, nếu cột hiện tại đã có đủ độ cao thì ta không cần xét tiếp các cột bên trái nữa. Tương tự, khi đi về bên phải cột i, nếu cột hiện tại đã có đủ độ cao thì ta không cần xét tiếp các cột bên phải nữa.

Tóm lại, gọi l là chỉ số j lớn nhất ở bên trái i sao cho $a_j \geq x-(i-j)$, r là chỉ số j lớn nhất ở bên phải i sao cho $a_j \geq x-(j-i)$. Khi đó, số miếng ghép cần thiết là:

$$\begin{aligned}
f_i &= \sum_{j=l}^{r}(a_j - (x - |i-j|)) \\
&= \sum_{j=l}^{r} a_j - \sum_{j=l}^{r} x + \sum_{j=l}^{r} |i-j| \\
&= \sum_{j=l}^{r} a_j - (r-l+1)x + \sum_{j=l}^{i}(i-j) + \sum_{j=i+1}^{r}(j-i) \\
&= \sum_{j=l}^{r} a_j - (r-l+1)x + \frac{(i-l)(i-l+1)}{2} + \frac{(r-i)(r-i+1)}{2}
\end{aligned}$$

Giá trị $\sum_{j=l}^{r} a_j$ có thể được thực hiện trong $O(1)$ bằng kĩ thuật mảng cộng dồn. Vấn đề còn lại là với mọi vị trí i, làm thế nào để tìm được hai chỉ số l và r tương ứng với độ phức tạp thời gian cho phép (ta sẽ kí hiệu hai chỉ số này lần lượt là $l(i)$ và $r(i)$).

Trước hết, điều kiện $a_j \geq x-(i-j)$ có thể được viết lại là $a_j + j \geq x - i$. Ta nhận xét rằng, i càng lớn thì điều kiện trên càng dễ thỏa. Do đó, ta sẽ tạo dãy b gồm các cặp $(a_j + j, j)$, rồi sắp xếp dãy này theo thứ tự giảm dần. Sau đó, ta có thể dùng kĩ thuật hai con trỏ (một con trỏ cho các vị trí từ 1 đến n, một con trỏ cho dãy b) để tính các giá trị $l(i)$ với tổng độ phức tạp $O(n)$.

Việc tìm mọi giá trị $r(i)$ trong $O(n)$ có thể được thực hiện tương tự.

Độ phức tạp: $O(n \log k)$ với n là số cột, k là số miếng ghép hiện có.

Tag: Binary Search, Math, Two Pointers