lee215 ★ 47714 Last Edit: October 19, 2019 10:35 PM 8.3K VIEWS

332

Please reply and upvote now.
Don't have prime membership.
Cannot even read and modify my own post later, when it's locked.

Explanation

We want to maximize the minimum sweetness.
Binary search the result between 1 and 10^9.

Don'e explain binary search too much again and again.
Please find more related explanation in **More**.
Also will explain it more in details on youtube lee215.

Time `O(Nlog(10^9))`
Space `O(1)`

Java

```java
    public int maximizeSweetness(int[] A, int K) {
        int left = 1, right = (int)1e9 / (K + 1);
        while (left < right) {
            int mid = (left + right + 1) / 2;
            int cur = 0, cuts = 0;
            for (int a : A) {
                if ((cur += a) >= mid) {
                    cur = 0;
                    if (++cuts > K) break;
                }
            }
            if (cuts > K)
                left = mid;
            else
                right = mid - 1;
        }
        return left;
    }
```

C++

```cpp
    int maximizeSweetness(vector<int>& A, int K) {
        int left = 1, right = 1e9 / (K + 1);
        while (left < right) {
            int mid = (left + right + 1) / 2;
            int cur = 0, cuts = 0;
            for (int a : A) {
                if ((cur += a) >= mid) {
                    cur = 0;
                    if (++cuts > K) break;
                }
            }
            if (cuts > K)
                left = mid;
            else
                right = mid - 1;
        }
        return left;
    }
```

Python:

```python
    def maximizeSweetness(self, A, K):
        left, right = 1, sum(A) / (K + 1)
        while left < right:
            if cur >= mid:
                cuts += 1
                cur = 0
            if cuts > K:
                left = mid
            else:
                right = mid - 1
        return right
```

More Good Binary Search Problems

Here are some similar binary search problems.
Also find more explanations.
Good luck and have fun.

- 1231. 1231. Divide Chocolate
- 1011. Capacity To Ship Packages In N Days
- 875. Koko Eating Bananas
- 774. Minimize Max Distance to Gas Station
- 410. Split Array Largest Sum

Comments: 25 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

liyun1988 ★ 205 Last Edit: October 19, 2019 10:33 PM

Frugal! On another hand I am surprised Leetcode have not reached out to you to award you free premium membership for your contribution to the community.

59 Show 5 replies Reply

SPzhao ★ 51 Last Edit: November 26, 2019 12:43 PM

You can reuse the code from 410 and change only ONE LINE of code.
The most voted answer to LC410 is
https://leetcode.com/problems/split-array-largest-sum/discuss/89817/Clear-Explanation%3A-8ms-Binary-Search-Java

and you only have to change

```
total = num;
```

to

Read More

17 Show 2 replies Reply

zenoflife ★ 25 Last Edit: November 6, 2019 1:10 PM

A good contrast explanation:

In this problem we want to find: Maximum of minimum total sweetness
In "Split Array Largest sum" we want to find: Minimum of maximum largest sum

In both places we do binary search on target answer, the difference is subtle but the key:

Read More

16 Show 2 replies Reply

ycui11 ★ 13 December 22, 2019 10:34 PM

When you decide to do binary search using (l+r)//2 and return l, when use (l+r+1)//2 and return r?
I am really confused.

8 Reply

CodeLenin ★ 47 October 20, 2019 12:17 AM

@lee215 can u explain why we need `(left + right + 1)` instead of `left+right`

8 Show 13 replies Reply

denis631 ★ 426 October 20, 2019 3:17 PM

Maaaan. It felt like a perfect DP problem.
This binary search heuristics approach is killing me

5 Show 2 replies Reply

iamgoodguy2018 ★ 178 Last Edit: October 20, 2019 3:27 AM

I wonder how the result of binary search (i.e., left) is guaranteed to be a sub-array of sweetness? From the following code, we evaluate if the sweetness can be cut into K parts, each of which is larger or equal (!!not EQUAL!!) to mid. I wonder how to justify the correctness, i.e., the returned value (left) must exist as a sub-array of sweetness. Thanks,

```
            int cur = 0, cuts = 0;
            for (int a : A) {
                if ((cur += a) >= mid) {
                    cur = 0;
                    if (++cuts > K) break;
```

Read More

4 Show 3 replies Reply

hzj_joy ★ 2 Last Edit: November 4, 2019 11:18 PM

If you are struggling about why `int mid = (left + right + 1) / 2;` and `left = mid`, hope my thought can help you.
This question asks me to find the maximum sum of subarray and this subarray should be the subarray with minimum sum. So the basic condition is the sum of this subarray is minimum. After satisfied the basic condition, we can find the maximum sum. So `if(condition)`, the sum may be my ans or too smaller(can be larger). That means `if(condition) left = mid.`
And for the 410 Split Array Largest Sum, it ask me to find the minimum sum of subarray and this subarray should be the subarray with maximum sum. The basic condition is the sum of this subarray is maximum. So `if(condition)` the sum may be my ans or too larger(can be smaller), that means `if(condition) right = mid.`

3 Show 1 reply Reply

kitsune89 ★ 8 February 21, 2020 8:14 AM

Excessively commented C++

```cpp
class Solution {
public:
    int maximizeSweetness(vector<int>& sweetness, int K) {
        //start from 1 piece
        int lowSweet = 1;
        //end at excessively large number
        //divide excessively large number by K+1 pieces
        int highSweet = 1e9 / (K+1);
```

Read More

1 Reply

BarOrz ★ 379 October 19, 2019 10:20 PM

410. Split Array Largest Sum is the closest problem I think.

1 Show 1 reply Reply