LeetCode 235

Abhinav Ganesh

January 30, 2024

209. Minimum Size Subarray Sum; Sliding Window

Prompt: Given an array of positive integers nums and a positive integer target, return the minimal length of a subarray whose sum is greater than or equal to target. If there is no such subarray, return 0 instead.

Constraints:

1. 1 <= target <= 10^9
2. 1 <= nums.length <= 10^5
3. 1 <= nums[i] <= 10^4

Solution: (Use Sliding Window)

Let our input array be A and positive integer target be t. Say A has n elements. Let $P(B)$ be the proposition that the sum of all elements in subarray B is $\geq t$. Let $S(A)$ be the set of all subarrays of A; $S(A) = \{A[i, j] | 0 \leq i < j \leq len(A)\}$ where the notation $A[i, j]$ is the subarray of A from indices i inclusive to j exclusive. Let f be a function that returns the negative length of an inputted array. We want to find

$$max_{s \in \{s \in S(A): P(s)\}} f(s)$$

Take any $B, B' \in S(A)$ s.t. $B \in S(B')$; B is a subarray of B'. Observe the following properties:

1. $f(B) \geq f(B')$; the value of f for a subarray will always be greater than that of the larger array. This is because by definition of subarray $len(B) \leq len(B') \implies f(B) \geq f(B')$.
2. $\neg P(B') \implies \neg P(B)$. The elements in B are a subset of the elements in B' by the definition of subarray. Then $\sum B \leq \sum B'$. If $\sum B' < t$ then $\sum B < t$.

We want the maximum value of $f(s)\ \forall\, s \in S(A)$ s.t. $P(s)$. Note that

$$S(A) = \bigcup_{i=0}^{n-1} \{\text{subarrays of A starting at index i}\}$$

Let the notation $S_i^j(A) = S(A[i:j])$ where $0 \leq i < j \leq len(A)$; this is the set of all subarrays of $A[i:j]$. Then

$$S(A) = \bigcup_{i=0}^{n-1} S_i^n(A)$$

We want to find the maximum value of $f(s)$ across $\{s \in S(A) : P(s)\}$. This is the same as

$$\max(\{f(s) | s \in \bigcup_{i=0}^{n-1} S_i^n(A), P(s)\})$$

$$= \max(\max(\{f(s)|s \in S_0^n(A), P(s)\}), ..., \max(\{f(s)|s \in S_{n-1}^n(A), P(s)\}))$$

In other words, we can first consider all subarrays of A starting at 0, then consider all subarrays of A starting at 1, and so on.

Consider all subarrays of A starting at 0, $S_0^n(A)$. We want to optimize f, which means having the smallest possible subarray. Therefore the smallest $k_0 \geq 1$ for which $P(A[0:k_0])$ will provide the greatest value of f out of all $s \in S_0^n(A)$.

Now consider all subarrays of A starting at 1, $S_1^n(A)$. Note that for some elements $s \in S_1^n(A)$, it is also the case that $s \in S_0^n(A)$. For example, $A[1:k]$. We know that k_0 was chosen so that $A[0:k_0]$ is the smallest subarray starting at 0 for which $P(A[0:k_0])$ is true. Then $P(A[0:k_0-1])$ is false $\implies \forall\, s' \in S(A[0:k_0-1])$, $P(s')$ is false (described in property 2). Note that $\forall\, j \leq k_0 - 1, A[1:j] \in S(A[0:k_0-1])$. Then we do not need to consider any subarray $A[1:j]$ where $j \leq k_0 - 1$ as we already know $P(A[i:j])$ will be false. We can let $x_0 = f(A[0:k_0])$.
Therefore, we only need to consider subarrays $A[1:j]$ where $j \geq k_0$.
We also know that f is optimized for the smallest length array. Then for the first value $k_1 \geq k_0$ for which $P(A[1:k_1])$ is true, we know $\nexists\ k' > k_1$ where $f(A[1:k']) > f(A[1:k_1])$, so we need not check any more subarrays starting at 1. Let $x_1 = f(A[0:k_0])$. Then $\max(f(s)|s \in \bigcup_{i=0}^{1} S_i^n(A), P(s)\})) = max(x_0, x_1)$.

We then consider all subarrays of A starting at 2, and so on and so forth until we reach $n = len(A)$. Then $\max(f(s)|s \in \bigcup_{i=0}^{1} S_i^n(A), P(s)\})) = max(x_0, x_1, ..., x_n)$. This is the value we want to return.

Sliding Window

We have used a technique that is generally known as Sliding Window, which is commonly used in problems where one must optimize across subarrays. By eliminating many of the subarrays from consideration, we are able to avoid an $O(n^2)$ Brute Force solution and instead implement an $O(n)$ solution (with $O(1)$ space complexity).

In a Sliding Window problem, you only want to consider a portion of subarrays of A for which the proposition $P(A)$ is true. Further, you want to optimize using a function f to find the subarray for which f is maximized.
In this problem, we define f to be optimal for the subarray with shortest length, and $P(A)$ means $\sum A \geq t$.

Usually the input constraints provide a relationship between subarray and superarrays for both f and P. In this case, given any subarray A and superarray A', $f(A) \leq f(A')$. Further, if $\neg P(A')$ then $\neg P(A')$. (Let a "superarray" A' of A be an array for which A is a subarray).

These relationships are useful because they help us eliminate certain subarrays from consideration. We only want to consider new subarrays S' s.t. $P(S')$ is true and $f(S') > f(S)\ \forall S$ for which we currently know $P(S)$ is true.

For example, given an array S for which we knew $P(S)$ was false, we eliminated all subarrays of S for consideration because in this question $\neg P(\text{array}) \implies \neg P(\text{subarray})$. Given an array for which $P(S)$ was true, we also ignored all superarrays as in this question $f(\text{array}) > f(\text{superarray})$.

Note that the relationships between subarrays and superarrays for f and P are usually specific to the problem being asked.