单调序列的题目

- <135. 最大子序和 - AcWing题库>
- 思路：

  我们从集合的角度去看，在一连串长度不超过 `m` 的连续子序列的集合中， 集合被划分为更小的n个集合， 考虑第 k 个集合来说， 首先如果想要求连续的子序列的和，我们常用的技巧是先转化为前缀和，考虑以`w[k]`结尾的连续的子序列， 令其长度为`j`那么 这个序列就是`(w[k - j + 1] ~ w[k] )`根据前缀和的性质：

那么这一段区间的和就是： **sum[k] - sum[j]** ；对于当前区间来说 **sum[k]** 为一个定值， 我们只要找到最小的**sum[j]** 就行了，**j**的取值范围是 **1 ~ m** 所以思路可以转化为在长度为m的窗口中找到最小的**sum[j]**，这个就是我们的单调队列的思路了；

注意点：

- 这一行代码中： `i - m > que[hh]` 中间不取等号， 因为 `que[hh]` 存储的是下标, 反过来也就是 `i - que[hh] > m` `que[hh]` 相当于是我们的左端点的下一个位置， 也就是 `j - 1` 那么可以替换一下 `i - (j - 1) > m` 而左边就是我们的连续序列的长度，发现等于的时候是合法的，所以不能将等于的情况出队。

```
if(hh <= tt && ( i - m ) > que[hh] ) hh ++ ;
```

- 与模板不同的是这里:

  ```
  int hh = 0 , tt = 0 ;
  ```

  这里的`tt`的初始值是 0， 而普通的队列中的初始值却是 `-1`，这里为什么是零呢?

  注意在初始化我们的单调队列的时候， 其中是有一个元素的为 0， 我们可以这样想：

  假如从 -1 开始， 如果sum 这个数组是一个单调递增且每一个元素都大于零，那么会先更新一下ans，但是之后这个下标会

  被加入到我们的队列当中，注意这时候 0，就会被覆盖。 那对于下一个位置的元素明显它所对应的左端点的下标的前一个位置应该为 0， 但此时却变为 1 了， 这就出现问题了，所以需要我们首先加入一个零。

  总结一下：

  这个一开始加入的 0 表示以零作为左端点的前一个位置也是一种合法的操作且必须执行的操作。

AC代码：

```
#include <iostream>
using namespace std ;
const int N = 3e5 + 10 ;
```

```cpp
int n , m ;
int sum[N] ;
int que[N] ;
int main()
{
    cin >> n >> m ;
    for(int i = 1; i <= n ; i ++ ) {
        cin >> sum[i] ;
        sum[i] += sum[i - 1] ;
    }
    int ans = -0x3f3f3f3f ;
    int hh = 0 , tt = 0 ;
    for(int i = 1; i <= n ; i ++)
    {
        if(hh <= tt && ( i - m ) > que[hh] ) hh ++ ;
        ans = max(ans , sum[i] - sum[que[hh]] ) ;
        while(hh <= tt && sum[que[tt]] > sum[i] ) tt -- ;
        que[++tt] = i ;
    }
    cout << ans << endl ;
    return 0 ;
}
```