

# dp的优化

## 一、单调队列优化

单调队列优化，可以降低诸如  $dp[i]=\max\{dp[j]\}+C[i](1\le j < i)$  这样的转移方程的时间复杂度。主要原理是利用一个单调队列来维护  $dp[i]$  的最值。

### 一道例题

[洛谷p2627](#)

我们不难写出动态转移方程

$$dp[i]=\begin{cases} \text{sum}[i]&i\le k \\ \max\{dp[j-1]+\text{sum}[i]-\text{sum}[j]\}&i-k\le j\le i,i>k \end{cases}$$

答案为  $\max\{dp[i]\}$

但是时间复杂度为  $O(nk)$  显然会超时。

注意到  $\text{sum}[i]$  可以直接提出，即  $dp[i]=\max\{dp[j]\}+\text{sum}[i]$  所以我们只需用一个单调队列来维护  $dp[i]$  的最值，复杂度可以降低为  $O(n)$

代码

```
#include <stdio.h>
#include <iostream>
using namespace std;
int n,k;
long long e[100001];
long long dp[100001];
long long sum[100001];
int q[100001];
int front=0,tail=1;
long long a(int now)
{
    return dp[now-1]-sum[now];
}
int main()
{
    scanf("%d %d",&n,&k);
    int rec=0;
    for (int i=1;i<=n;i++)
    {
        scanf("%lld",&e[i]);
        sum[i]=sum[i-1]+e[i];
    }
    for (int i=1;i<=n;i++)
```

```
{
while (front<=tail&&q[front]<i-k)
front++;
dp[i]=a(q[front])+sum[i];
while (front<=tail&&(q[tail])<=a(i))
tail--;
q[++tail]=i;
}
long long ans=0;
for (int i=1;i<=n;i++)
ans=max(ans,dp[i]);
printf("%lld",ans);
return 0;
}
```

## 练习

[洛谷p3957](#)

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:dp%E7%9A%84%E4%BC%98%E5%8C%96&rev=1590924841](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:dp%E7%9A%84%E4%BC%98%E5%8C%96&rev=1590924841) 

Last update: 2020/05/31 19:34