思路就是这样的：

- 构造一个二维数组 f ，$f[i][j]$ 表示在 i 分钟时，疲劳度为 j 的情况下，最多能跑几米。

> 举个例子，$f[5][3]$ 表示第5分钟，当疲劳度为3时最多能走多少路。

- 这样定义完之后，我们就可以知道 第0分钟 ，疲劳度为0 （ `f[0][0]` ，也就是还没出发）时最大距离为0。

那怎么算呢?

有两步，最后求 $f[n][0]$（即n$ 秒时，疲劳度为 0 的最大距离）：

(i)：如果是求 $f[i][0]$ 的值，就是这一秒想休息，就有两种情况：

1. 是休息 j 分钟过后得来的：
   这时的距离是 j 分钟前的距离，也就是 $f[i-j][j]$。
   循环遍历一下，求出 $max_{f[i-j][j]}$ 。
2. 或者是上一分钟疲劳度就是 0：
   这样疲劳度的另一种可能就是 $f[i-1][0]$ 。

以上 1、2 两个结果再取 max 就是这一分钟开始前的最大距离.

如果不是 $f[i][0]$ ，就当他这一秒前没休息，因为休息的距离肯定更小，取 max 肯定会被舍弃，索性初始化为 `0` .

(ii)：这一秒肯定是要跑步的，因为休息的情况前面都考虑了，而且跑步的距离肯定会更大，这一秒（ `f[i][j]` ）的最大距离就是 $f[i][j]$（初始化值）或是 $f[i-1][j-1]+D_i$，即前一秒的值+这一秒能跑的距离。
公式是：$max_{(f[i][j],\ f[i-1][j-1]+D_i)}$

状态转移方程：

3个方程，顺序执行：

$$f[i][0] = Max(f[i-j][j], f[i][0]) \tag{1}$$

$$f[i][0] = Max(f[i-1][0], f[i][0]) \tag{2}$$

$$f[i][j] = Max(f[i-1][j-1] + D_i, f[i][j]) \tag{3}$$

下面是我DP部分的代码：

```
for(int i=1;i<=n;i++)
{
    for(int j=1;j<=min(i, m);j++)
        f[i][0] = max(f[i-j][j], f[i][0]);
    f[i][0]=max(f[i][0], f[i-1][0]);
    for(int j=1;j<=m;j++)
        f[i][j] = max(f[i-1][j-1] + d[i], f[i][j]);
}
```

完整代码如下：

```
#include<iostream>
using namespace std;

int n,m,d[11000];
int f[11000][550];

int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        cin>>d[i];

    f[0][0]=0;

    for(int i=1;i<=n;i++){
        for(int j=1;j<=min(i, m);j++)
            f[i][0] = max(f[i-j][j], f[i][0]);
        f[i][0]=max(f[i][0], f[i-1][0]);
        for(int j=1;j<=m;j++)
            f[i][j] = max(f[i-1][j-1] + d[i], f[i][j]);
    }

    cout<<f[n][0];
}
```

注意：计数要从1开始，因为再算0分钟没有意义。