包来实现，对于每一个字符串，将它转化成几个物品，把每一个字符串切割成全部是1 的花费算出来，这样的话我们就有了几个全部为1 的字符串花费为切割的花费，价值就是1小字符串的长度，前缀1 和后缀1 切割只需要花费1 ，其他花费2 ，但是还有一个问题，如果把中间的一个在能切割的范围内放在最后，那么就少切割一次，也就相当于k+1，意思就是 011001111001 这样的一个串，可以用5 次切出来全部是1 但是实际上只需要切4次就可以，因为可以将11放最后或者将1111放最后，也就是组成的串为1，11,111100，00,0 或者 1,1111,1100，00，0。下面是代码

但是我“就是拼接在最后的也只用1次”想复杂了，以为有什么特殊情况   
今早冷静分析了一下，其实就等价于k++

#include <cstdio>

#include <cstring>

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

struct node

{

int cost,l;

} a[10010];

int dp[10010];

int main()

{

//freopen("input.txt","r",stdin);

int n,k;

char s[10010];

while(~scanf("%d%d",&n,&k))

{

scanf("%s",s);

int l=strlen(s);

bool flag(0);

int l1(0);

int tot(0);

int i;

for(i=0; i<l; i++)

{

if(s[i]=='0' && flag)

{

flag=0;

if(tot==0&&s[0]=='1')

a[tot++]=node{1,l1};

else a[tot++]=node{2,l1};

l1=0;

}

if(s[i]=='1')

{

flag=1;

l1++;

}

}

if(l1!=0)

a[tot++]=node{1,l1};

if(k==0)//要特判,否则后面k+1上去结果就会不一样了

{

if(s[0]=='1')

{

printf("%d\n",a[0].l);

}

else puts("0");

continue;

}

k++;

for(int i=0; i<=k; i++)

dp[i]=0;

for(int i=0; i<tot; i++)

for(int j=k; j>=a[i].cost; j--)

{

dp[j]=max(dp[j],dp[j-a[i].cost]+a[i].l);

}

printf("%d\n",dp[k]);

}

return 0;

}