搬运自bilibili专栏。

卖菜问题

假设您有一个数组a[]第i个元素a[i]是第i天给定大头菜的价格。

设法计算最大的利润。您最多可以完成k次交易。（一买一卖算作一次交易）

请注意，无法同时进行多项交易，必须先出售大头菜才能再次购买。

（只能空仓或满仓，一天最多只能切换一次状态）

（不允许结束后还持有大头菜）

多组输入数据：

每组数据第一行两个数n,k[]表示总天数和最多交易次数[](1≤n,k≤10^3)

接下来一行n个数表示大头菜的价格[](1≤a[i]≤10^9)

题解

首先，有一个显然的事实。当最大可交易次数k超过总天数n的一半的时候，相当于不限制交易次数，即可以交易任意多次，因为按照上述规则，无论怎么交易，总次数无论如何也不可能超过总天数n的一半，即总是合法的。

因此，可以买入每一个相对低位，卖出每一个相对高位，即占满每个上升区间。写成代码就是这样的：

```
if(k>n/2)
{
    long long profit=0;
    int i;
    for(i=1;i<n;i++)
    {
        if(a[i]>a[i-1])
        {
            profit+=(a[i]-a[i-1]);
        }
    }
    return profit;
}
```

其他的情形怎么办呢？

可以采用经典的数学归纳法的思想，对天数进行归纳。假设我们已经知道了前一天的情形，考虑下一天。

设置两个数组[]sell和hold[]代表大头菜的空仓和满仓，大小均为[2][1005]。

其中，第一位表示今天或昨天[]cnt表示今天[]1-cnt表示昨天。

那么利用熟知的异或操作，语句“cnt^=1;”表示切换今天与昨天，那么前天的状态就被今天替代了。

第二位，表示已经操作了多少次。每当从sell状态转移至hold的时候，即买入一次的时候，这一位应当更新一次，而其余情况均不应该更新。

数组存储的内容，是该状态下的最大利润，即满足当天“不超过k次交易”的条件下，空仓或满仓的最大利润。

根据题目限定，最终答案状态应该是空仓。所以最终答案应该存储在sell[cnt][k]中。

初始化操作，即最初的一天的状态，显然是确定的。在空仓数组中应当是0，在满仓数组中应该是一个负值，即花钱持有了大头菜，利润为负。

```
for(i=0;i<=k;i++)
{
    sell[cnt][i]=0;
    hold[cnt][i]=-a[0];
}
```

那么状态转移就很容易写了。每次更新时sell[cnt][j]取sell[1-cnt][j]和(hold[1-cnt][j]+a[i])中的较大值，代表两种可能性：本来就是空仓，或者由满仓在当天卖出得到的空仓。同样的hold[cnt][j]取hold[1-cnt][j]和(sell[1-cnt][j-1]-a[i])的较大值。

综上代码已经全部解释完毕。全体代码如下。

代码

```
#include<stdio.h>
#include<string.h>

long long sell[2][1005];
long long hold[2][1005];
long long a[1005];

long long maxProfit(int n,int k)
{
    if(k>n/2)
    {
        long long profit=0;
        int i;
        for(i=1;i<n;i++)
        {
            if(a[i]>a[i-1])
            {
                profit+=(a[i]-a[i-1]);
            }
        }
        return profit;
    }
    memset(sell,0,sizeof(sell));
    memset(hold,0,sizeof(hold));
    int cnt=0;
```

```
    int i;
    for(i=0;i<=k;i++)
    {
        sell[cnt][i]=0;
        hold[cnt][i]=-a[0];
    }
    for(i=1;i<n;i++)
    {
        cnt^=1;
        int j;
        for(j=1;j<=k;j++)
        {
            sell[cnt][j]=(sell[1-cnt][j]>(hold[1-cnt][j]+a[i])?sell[1-
cnt][j]:(hold[1-cnt][j]+a[i]));
            hold[cnt][j]=(hold[1-cnt][j]>(sell[1-cnt][j-1]-a[i])?hold[1-
cnt][j]:(sell[1-cnt][j-1]-a[i]));
        }
    }
    return sell[cnt][k];
}

int main()
{
    int n,k;
    while(~scanf("%d%d",&n,&k))
    {
        memset(a,0,sizeof(a));
        int i;
        for(i=0;i<n;i++)
        {
            scanf("%lld",&a[i]);
        }
        printf("%lld\n",maxProfit(n,k));
    }
    return 0;
}
```