

题意：（太容易理解了）



30 分：

对于 20%的数据， $K=1$ ；
对于额外的 10%的数据， $K=N$ ；

直接对特殊数据进行骗分处理就行了

骗分处理部分：

```
int
```

```

main1()
{
    int L=0,R=0;
    for(int i=1;i<=n;i++) L=max(L,a[i]),R=max(R,b[i]);
    LL ans=0,sum=L*R;
    for(int i=1;i<=n;i++) ans+=(LL)c[i]*(sum-a[i]*b[i]);
    printf("%lld\n",ans);
    return 0;
}

int
main2()
{
    printf("0\n");
    return 0;
}

```

60 分： dfs

使用状态压缩，预处理一个信封中所有情况的花费，然后暴力 dfs

设 $cost_i$ 表示状态为 i 时的花费，为了方便转移，设 L_i 表示状态为 i 时的长度， R_i 表示状态为 i 时的宽度， num_i 表示状态为 i 时的信封总数(可以理解为厚度)，由小学的数学知识可知，状态为 i 时，有以下式子：

$$L_i = \max_{j \in i} a_j$$

$$R_i = \max_{j \in i} b_j$$

$$num_i = \sum_{j \in i} c_j$$

$$cost_i = L_i R_i num_i - \sum_{j \in i} a_j b_j c_j$$

用上述式子进行更新就行了。

CODE:

```

all=(1<<n)-1;
for(int i=0;i<=all;i++)
    for(int j=1;j<=n;j++)

```

```

        if(!(i&1<<j-1)) //未包含，可以扩展
        {
            L[i|1<<j-1]=max(L[i],a[j]);
            R[i|1<<j-1]=max(R[i],b[j]);
            num[i|1<<j-1]=num[i]+c[j];
            cost[i|1<<j-1]=cost[i]+(LL)a[j]*b[j]*c[j];//让cost[i]变为
a[j]*b[j]*c[j]
            //这样方便下一步的更新
        }
        for(int i=0;i<=all;i++) cost[i]=(LL)num[i]*L[i]*R[i]-cost[i];

```

然后.....暴力就行了。

Code:

```

LL ans=Inf;
void
dfs(int now,LL sum,int step)
{
    if(step==k+1)//递归返回
    {
        if(now==all)//状态满足要求
            ans=min(ans,sum);//更新答案
        return ;
    }
    for(int i=1;i<=all;i++)//枚举全集，其实还可以优化
    {
        if(sum>=ans) return ;//剪枝
        if(sum+cost[i]>=ans) continue;//剪枝
        if((now&i)==0)//可以扩展
        {
            dfs(now|i,sum+cost[i],step+1);//dfs 下一层
            if(sum>ans) return ;// 剪枝
        }
    }
}

```

只要剪枝做得足够好，暴力还是能拿很多分的。

100 分(正解): Dp

设 $f_{i,j}$ 表示状态为 i , 用了 j 个信封时的最小花费。状态转移方程:

$$f_{i,j} = \min_{l, s \in i, l \cap s = \emptyset, l \cup s = i} (f_{l,j-1} + cost_s)$$

边界: $f_{0,j} = 0, 1 \leq j \leq k$

Code:

```
for(int i=1;i<=all;i++) f[i][1]=cost[i]; //赋初值
for(int i=1;i<=all;i++) //枚举第一维
    for(int j=1;j<=k;j++) //枚举第二维, 注意边界
        for(int hh=i;hh;hh=(hh-1)&i) //枚举子集
            f[i][j+1]=min(f[i][j+1],cost[i^hh]+f[hh][j]); //状态更新
printf("%lld\n",f[all][k]); //Perfact print.
```

那么, 本题就这么完了

Show all the code:

```
#include<cstdio>
#include<algorithm>
#include<iostream>
#include<cstring>
using namespace std;
```

```

typedef long long LL;
const int N=15+10;
const LL Inf=0x3f3f3f3f3f3f3f;

LL f[1<<15|1][N+1],cost[1<<15|1];
int L[1<<15|1],R[1<<15|1],num[1<<15|1];
int n,k,a[N+1],b[N+1],c[N+1];
int all;

char buf[1<<23|1],*p1=buf,*p2=buf;
#define getchar() (p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<23,stdin),p1==p2)?EOF:*p1++)
template<class _T>inline void//致命快读
read(_T &x)
{
    x=0;char ch=getchar();
    for(;!isdigit(ch);ch=getchar());
    for(;isdigit(ch);ch=getchar()) x=(x<<3)+(x<<1)+(ch&15);
}

int
main()
{
    freopen("envelope.in","r",stdin);
    freopen("envelope.out","w",stdout);
    read(n);read(k);
    for(int i=1;i<=n;i++) read(a[i]),read(b[i]),read(c[i]);
    all=(1<<n)-1;
    for(int i=0;i<=all;i++)
        for(int j=1;j<=n;j++)
            if(!(i&1<<j-1)) //未包含, 可以扩展
            {
                L[i|1<<j-1]=max(L[i],a[j]);
                R[i|1<<j-1]=max(R[i],b[j]);
                num[i|1<<j-1]=num[i]+c[j];
                cost[i|1<<j-1]=cost[i]+(LL)a[j]*b[j]*c[j];//让cost[i]变为
a[j]*b[j]*c[j]
                //这样方便下一步的更新
            }
    for(int i=0;i<=all;i++) cost[i]=(LL)num[i]*L[i]*R[i]-cost[i];
    for(int i=0;i<=all;i++)//手动刷Inf,反正常数也不大,可以用memset
        for(int j=0;j<=k;j++)
            f[i][j]=Inf;
    for(int i=1;i<=all;i++) f[i][1]=cost[i];//赋初值
    for(int i=1;i<=all;i++)//枚举第一维

```

```
        for(int j=1;j<k;j++)//枚举第二维, 注意边界
            for(int hh=i;hh;hh=(hh-1)&i)//枚举子集
                f[i][j+1]=min(f[i][j+1],cost[i^hh]+f[hh][j]); //状态更新
    printf("%lld\n",f[all][k]); //Perfact print.
    return 0;
}
```