H So Many Problems——李晨豪&户建坤

Description

这是 Thor 与 Arthur 之间的博弈。
Thor 经常和 Arthur 比对出题事业的贡献程度。但是题目考点是有限的，只有 n 个，于是他们规定轮流出题。
对于第 i 个题目，他们规定了一种题目质量指数 a[i]。然后他们发现题目之间都是有联系的，比如如果题目 A 和题目 B 是一个人出的，那么这两个题目的质量都会有所上升。他们发现了 m 条这样的关系，对于第 j 条关系所能增加的题目质量为 b[j]。
所以，两个人都想比对方的贡献值多尽量多，在这个决策标准下，问如果两个人都足够聪明，且在 Arthur 先选的情况下，Arthur 的贡献程度-Thor 的贡献程度是多少。

Input

多组数据。
每组数据第一行两个整数 n,m(0<=n<=100000,0<=m<=100)表示有 n 个题目考点，m 个关系。
接下来 n 行。每行一个整数 a[i](0<=a[i]<=10000)表示第 i 个题目的贡献。
接下来 m 行。每行三个整数 x[i],y[i],b[i](0<=b[i]<=200)表示如果第 x[i]和 y[i]个题都是同一个人出的可以会为这个人增加 b[i]的贡献。保证 b[i]为偶数。

Output

对于每组数据输出一个数，即 Arthur 的贡献程度-Thor 的贡献程度。

Input Sample

```
5 2
1
2
3
4
5
1 2 4
2 3 2
1 0
```

```
2
3 1
1
1
2
1 2 4
```

Output Sample

```
3
2
2
```

Hint

仔细阅读样例，明确决策标准。
就是给一堆点一些边（无重边），两个人博弈。点有点权，边有边权，选了一边的端点可以得到边权。问两人都取最优的情况下，两个人得分的差值。两个人决策的标准是让自己的得分比对方的得分尽量高。
比如第3组样例，Arthur 选了1点，那么 Thor 会去选择2点而不是3点，然后 Arthur 选择3点，如此答案为2。

分析 1：

这是一种博弈的思想，通过使对方的得分尽可能低来实现自身的胜利。这道题中困扰我们的是 b[i]的加成。我们分析如下情形：

作为 Thor，我们有 a[k] a[m]两种选择，其中 a[k]+b1(对应的) > a[m] + b2(对应的),b1 和 b2 可能为 0。但是 a[k] < a[m] + b2，（我们假设 b1,b2 是确定的值）这时我们首先要考虑的是 a[k]能否对 thor 有加成（因为就结果来看，若是不对 Thor 有加成那么即对 Authur 有加成）

如果有加成的话，那么无论 a[m]是对自己的加成还是 Arthur 的，我们作为机智的 Thor 都会选择 a[k]。

如果没有加成的话，但是 a[m]却有加成。我们可能会了让自己的分大而选择 a[m]。但是我们应该这样想，我们不要 a[k]，a[k]就要去 Authur 那里了啊！！！ 于是乎，

Ans_Aurthur+=a[k]+b1,Ans_Thor+=a[m]+b2,Ans_Delta += a[k]-a[m]+b1-b2;

选 a[k]的话：

Ans_Aurthur+=a[m],Ans_Thor+=a[k],Ans_Delta += a[m]-a[k];

于是 Thor 令 Ans_delta 尽可能小，

选 a[m] a[m]-a[k]>a[k]-a[m]+b1-b2 即 2*a[m]+b2>2*a[k]+b1;即 a[m]+1/2b2>a[k]+1/2b1

选 a[k] a[m]-a[k]<a[k]-a[m]+b1-b2 即 2*a[m]+b2<2*a[k]+b1;即 a[m]+1/2b2<a[k]+1/2b1

咦？！貌似有规律，于是我们回到第一种情况：

我们同样得到了如果 a[m]+1/2b2>a[k]+1/2b1 就选 a[m],否则就选 a[k]的结论。

方案选取的正确性得到了证明，然后我们再看答案的正确性：

令 a[i] = a[i] + $\sum b$ /2

若有 x[i],y[i],b[i],那么只有两种情况：

x[i]与 y[i]不在一起，那么因为 Thor 和 Authur 两人都被多加了 b[i]/2，所以答案并不影响。

x[i]与 y[i]在一起，那么其中一人存在加成 b[i]/2+b[i]/2=b[i]，仍然正确。

所以我们修改各题目原值后，只需进行一次排序即可。然后由大到小按顺序抽取。

这时候我们发现数据规模 100000.。。。冒泡排序肯定是过不去的，所以我们可以用更快的排序，例如快速排序，堆排序，归并排序。

然后这次大家都还没学过这些，就用函数 sort（#include<algorithm>），或者 qsort(#include <cstdlib>)吧.

用法(从大到小)：

```
bool com(int a,int b)
{
    return a>b;
}
Sort(a+1,a+n+1,com);
//////////////////////////////////////////////
int cmp(const void* a,const void* b)
{
        return (*(int*)b-*(int*)a);
}
```

```
qsort(a+1,n,sizeof(int),cmp);
```

分析 2：

主要分为两部分。一，弄清楚博弈的意思，并找到该题的解法。二，在不超时情况下完成代码。

First： **1.考虑增加贡献度对原来每道题贡献度的影响。（结论在下面）**设 a,b,z 分别为第 x[i]和 y[i]个题都是同一个人出的可以会为这个人增加 b[i]的贡献中的想 x[i],y[i],b[i].为了比较，在引入一个 c（任意的一个贡献度）。现在开始讨论并归纳结论：

设 A 先选，并为了方便，不妨设 b>=a;

	第一组		第二组		第三组	
A	a		b		c	
B	b	c	a	c	a	b
A	c	b	c	a	b	a
	a+c-b	a+b+z-c	b+c-a	a+b+z-c	b+c –a	a+c-b

以上是所有可能。A,B 足够聪明，指的是：B 能够在 A 选过后选出两个结果中最小的，而 A 因为知道 B 会选最小的，所以他会选三个最小的中的最大的。结果就是分类讨论各组最小值中的最大值。

经过简单的计算和化简可以知道：

当：z>2*(c – a),最优解为 b+c-a;

当：2*(c-b)<z<2*(c – a),最优解为 a+b+z-c;

当：z<2*(c –b),最优解为 c+a-b;

观察一下，可以整理为：

当：a+z/2>c,最优解为(b+z/2)+c-(a+z/2);

当：a+z/2<c<b+x/2,最优解为(a+z/2)+(b+z/2)-c;

当：c>b+z2,最优解为 c+(a+z/2)-(b + z/2);

将 a+z/2　b+z/2 换元为 A,B；

结论就很明显了：

当：c<A,最优解为 B+c-A;

当：A<c<B,最优解为 A+B-c;

当：c>B,最优解为 c+A-B;

即：**第 x[i]和 y[i]个题都是同一个人出的可以会为这个人增加 b[i]的贡献　等价于　x[i]题贡献度变为原来的加上 b[i]/2，y[i]题贡献度变为原来的加上 b[i]/2。**

2.现在就简单多了，将等价过得序列降序排序，肯定是 Arthur 拿走奇数，Thor 拿走偶数= =不解释。基本的算法就是这样。

Second ：代码的实现，详见参考代码。最重要的是排序快慢会影响超时。

三．参考代码

```
#include <iostream>
#include <stdio.h>
#include <algorithm>
using namespace std;
int a[100010],x,y,b;
bool p ;
bool com(int a,int b)
{
	return a>b;
}
int main()
{     int n,m,t,x,ans1,ans2;
	while ( scanf("%d %d",&n,&m) != EOF )
	{
		for ( int i = 1;i <= n;i++ )
		scanf("%d",&a[i]);
		for ( int i = 0;i < m ;i++ )
		{
		     scanf("%d %d %d",&x,&y,&b);
		     a[x] += b / 2;
		     a[y] += b / 2;
		}
		sort(a+1,a+n+1,com);
		ans1 = 0;
		ans2 = 0;
		for ( int i = 1;i <= n;i = i+ 2)
		ans1 = ans1 + a[ i ];
		for ( int i = 2;i <= n;i = i+ 2)
		ans2 = ans2 + a[ i ];
		printf("%d\n",ans1 - ans2);

	}

	return 0;
}
```