问题 C: Kefa 与伴侣

题目描述

Kefa 想去餐厅庆祝他的第一份高薪。他有 n 个朋友，每个朋友有一定的友谊值和工资。没人想觉得自己穷，所以 Kefa 邀请的朋友中两两工资差小于 d。现在给出朋友的信息，请求出最大友谊值是多少。

输入格式

输入的第一行包含两个空格分隔的整数，n 和 d（$(1\le n\le 10^5, 1\le d\le 10^9)$ ——分别是 Kefa 的朋友数量和令人感到贫穷的最小金额差。

接下来的 n 行包含对 Kefa 的朋友的描述，第（i+1）行包含对第 i 个朋友的 mi,si 的描述（0≤mi, si≤109) – 钱的数量和友谊值。

输出格式

输出可达到的最大总友谊值。

示例

Input1

```
4 5
75 5
0 100
150 20
75 1
```

Output1

```
100
```

Input2

```
5 100
0 7
11 32
99 10
46 8
87 54
```

Output2

```
111
```

在第一个样本测试中，最有效的策略是仅由第二个朋友组建伴侣。在所有其他变体中，友谊的总体程度会更糟。

在第二个样本测试中，我们可以带走所有的朋友。

题解：两两的差其实转化为最大和最小的差。那么我们先对所有人按工资排序，然后枚举最小的人是谁，在他工资上加 d，然后找到小于这个值的人并求和。这可以用二分查找找到小于这个值的最大工资的人，求和的话可以前缀和。
代码如下：

```
#include<bits/stdc++.h>
using namespace std;

const int maxn = 1e5 + 10;
long long n, d, ans, sum[maxn];
struct node {
    long long x, y; // x is money;
}a[maxn];
bool cmp(node a, node b) {
    return a.x < b.x;
}
int main(){
    cin >> n >> d;
    for(int i = 1; i <= n; ++i)
        cin >> a[i].x >> a[i].y;
    sort(a + 1, a + n + 1, cmp);
    for(int i = 1; i <= n; ++i)
        sum[i] = sum[i - 1] + a[i].y; //获取前缀和，注意一定要在排序之后
    int r = 1;
    for(int l = 1; l <= n; ++l) {
        while(r <= n && a[r].x - a[l].x < d) { //不满足退出
            ans = max(ans, sum[r] - sum[l - 1]);
            r ++; //满足条件，继续寻找
        }
    }
    cout << ans;
    return 0;
}
```

方法 2：

```
#include <bits/stdc++.h>
using namespace std;
int n;
long long d;
struct point
{
    long long x, y;
```

```
} a[100005];
long long s[100005], ans;
bool cmp(point x, point y)
{
    return x.x < y.x;
}
int find(long long x)
{
    int l = 1, r = n;
    while (l <= r)
    {
        int mid = (l + r) / 2;
        if (a[mid].x >= x)
        {
            r = mid - 1;
        }
        else
            l = mid + 1;
    }
    return r;
}
int main()
{
    cin >> n >> d;
    for (int i = 1; i <= n; i++)
        cin >> a[i].x >> a[i].y;
    sort(a + 1, a + n + 1, cmp);
    for (int i = 1; i <= n; i++)
        s[i] = a[i].y + s[i - 1];
    for (int i = 1; i <= n; i++)
    {
        long long p = a[i].x + d;
        int u = find(p);
        ans = max(ans, s[u] - s[i - 1]);
    }
    cout << ans;
    return 0;
}
```

问题 E: 午餐费用

题目描述

Watashi 是 ZJU-ICPC 团队的队长，他非常善良。在 ZJU-ICPC 夏令营中，学生被分成几个小组，每天其中一个小组将设计一些题目来举行比赛。今天 C 组的学生被要求设计问题，他们花了一整晚的时间来检查测试数据，这让他们非常疲劳。Watashi 决定给 C 组一些钱作为奖

励，以便他们可以免费购买午餐。

培训日程中有 N 天，所有学生都预订了 N 天的午餐，所以我们知道他们每天要花多少钱。现在，C 组组长需要决定如何使用 Watashi 的钱。由于钱有限，他们不可能每天都有免费的午餐。因此，每天领导者可以选择自己支付整个团队的午餐费用，或者该日午餐费用全部使用 Watashi 的钱支付。当然，领导希望尽可能多地花 Watashi 的钱，但他太忙了，没有时间编写程序来计算他可以从瓦塔西奖励中花费的最大金额。你能帮他吗？

输入格式

输入包含多个测试用例（不超过 50 个测试用例）。

在每个测试案例中，首先有两个整数，N（1<=N<=30），这是训练天数，M（0<=M<=100000），是 Watashi 的奖金。

然后有一行包含 N 个正整数，其中第 i 个整数表示 C 组需要支付第 i 天的午餐。所有这些整数都不超过 10000000，整数之间用空格隔开。

输出格式

对于每个测试用例，输出一行整数，这是 C 组可以从 Watashi 的奖励中花费的最大金额

输入样例

3 10

8 4 5

输出样例

9

题解：要用 dfs+剪枝来写。就剪枝来说，主要有几个方面：一是满足条件则返回；二是每次都更新最优值；三是花费和已经超过奖金数，则返回；四是下标超过则返回；五是加当前的值小于奖金，则下标和值下移，否则回溯。具体在注释已经给出，详见 code。

代码如下：

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

const int MAXN = 30+5;
int n,m,ans,tmp,flag;
int c[MAXN];

void dfs(int i,int d){
	if(flag) return ; //已经满足，则不需要遍历了
	if(d==m){ans=d;flag=1;return ;} //满足则返回
	if(ans<d) ans=d; //更新最优值
	if(d>m) return ; //已经超过 m，则后面的不需遍历了
	if(i>=n) return ; //下标已经超过 n，则后面的不需遍历了
	if(d+c[i]<=m) //当相加和小于 m，则下标和值都加
		dfs(i+1,d+c[i]);
	dfs(i+1,d); //否则回溯，移动下标进行下一次判断
}
```

```
int main(){
	while(~scanf("%d%d",&n,&m)){
		ans=0;tmp=0;flag=0;
		for(int i=0;i<n;i++){
			scanf("%d",&c[i]);
			tmp+=c[i];
		}
		if(tmp<=m){ //所有数之和小于等于 m，则说明必须全部每天都要花掉
			printf("%d\n",tmp);
			continue;
		}
		dfs(0,0);
		printf("%d\n",ans);
	}
	return 0;
}
```