讲评课

TEACHING COURSEWARE POWPOINT

授课时间：2025.08.06

启航舱
中小学拔尖创新人才培养中心

目录

01

A

TEACHING
COURSEWARE

TEACH

A

题目描述

夏天到了，各家各户的用电量都增加了许多，相应的电费也交的更多了。小玉家今天收到了一份电费通知单。小玉看到上面写：据闽价电 [2006]27 号规定，月用电量在 150 千瓦时及以下部分按每千瓦时 0.4463 元执行，月用电量在 $151 \sim 400$ 千瓦时的部分按每千瓦时 0.4663 元执行，月用电量在 401 千瓦时及以上部分按每千瓦时 0.5663 元执行；小玉想自己验证一下，电费通知单上应交电费的数目到底是否正确呢。请编写一个程序，已知用电总计，根据电价规定，计算出应交的电费应该是多少。

输入格式

输入一个正整数，表示用电总计（单位以千瓦时计），不超过 10000。

输出格式

输出一个数，保留到小数点后 1 位（单位以元计，保留到小数点后 1 位）。

输入数据 1

```
267
```

Copy

输出数据 1

```
121.5
```

Copy

A

1. 根据用电量的不同范围，分段计算电费：

若用电量≤150，直接用用电量 ×0.4463

若 150 <用电量≤400，150×0.4463 +（用电量 - 150）×0.4663

若用电量 > 400，150×0.4463 + 250×0.4663 +（用电量 - 400）×0.5663

A

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n;          // 输入的用电量
    double ans;     // 计算得到的电费
    // 读取输入的用电量
    cin>>n;
    // 根据不同用电量范围计算电费
    if (n <= 150) {
        ans = n * 0.4463;
    } else if (n <= 400) {
        ans = 150 * 0.4463 + (n - 150) * 0.4663;
    } else {
        ans = 150 * 0.4463 + 250 * 0.4663 + (n - 400) * 0.5663;
    }

    // 输出结果，保留一位小数
    printf("%.1f\n", ans);

    return 0;
}
```

02

B

TEACHING
COURSEWARE

TEACH

B

前缀串

输入文件 string.in 输出文件 string.out
时间限制 1000ms 空间限制 128MB

题目描述

Alice 现在有一个目标串 S。Bob 现在有一个串集，这个串集中总共有 n 个字符串。
Alice 想要找到在 Bob 的串集中，一个以目标串 S 为前缀，且字典序最小的字符串。
保证 Bob 的串集中至少有一个串的前缀是 S。

输入格式

第一行一个字符串 S，表示 Alice 手中的目标串 S。
接下来一行一个正整数 n，表示 Bob 手中的串集中的字符串数量。
接下来 n 行，每行一个字符串，表示 Bob 手中串集的其中一个串。
所有字符串中均只含有小写字母。

输出格式

一行一个字符串，如题面所求。

样例 #1

样例输入 #1

```
next
2
nextpermutation
nextelement
```

样例输出 #1

```
nextelement
```

大样例见文件 string.in/out

B

数据范围

共 10 个测试点，每个测试点 10 分。

测试点编号	数据范围	其他说明
#1~#4	$1 \le n \le 100$ $1 \le len \le 100$， 其中 len 是字符串长度	目标串 S 长度为 1
#5		Bob 串集中的串均相同
#6~#10		无

B

从一组字符串中筛选出以特定前缀开头的字符串，并找出其中字典序最小的那个。

具体流程为：先输入作为前缀的字符串s和待检查的字符串总数n，然后依次读取n个字符串，通过check函数判断每个字符串是否以s为前缀。对于符合条件的字符串，若为第一个符合条件的，则直接存入结果变量ans；若后续还有符合条件的，则将其与当前ans进行字典序比较，取较小的那个更新ans。最后，输出筛选出的字典序最小的符合条件的字符串。

B


```cpp
#include<bits/stdc++.h>
using namespace std;
// 检查字符串s2是否以s1为前缀
// 如果s2的前s1.size()个字符与s1完全相同，返回1；否则返回0
int check(string s1, string s2)
{
    // 遍历s1的每个字符
    for(int i = 0; i < s1.size(); i++)
    {
        // 若对应位置字符不同，返回0
        if(s1[i] != s2[i]) return 0;
    }
    // 所有字符都匹配，返回1
    return 1;
}
```

B

```cpp
int main() {
    string s = "";  // 存储前缀字符串
    cin >> s;       // 读取前缀字符串
    int n;          // 存储输入的字符串数量
    cin >> n;
    string ans = "";  // 存储结果，初始为空
    // 循环读取n个字符串
    while(n--)
    {
        string ss;
        cin >> ss;  // 读取当前字符串

        // 检查当前字符串是否以s为前缀
        if(check(s, ss))
        {
            // 如果是第一个符合条件的字符串，直接赋值给ans
            if(ans == "")
            {
                ans = ss;
            }
            // 否则取当前字符串和ans中的较小者（字典序）
            else ans = min(ss, ans);
        }
    }
    // 输出结果
    cout << ans << endl;

    return 0;
}
```

03

C

TEACHING
COURSEWARE

TEACH

C

好冷好热好冷好热

题目描述

一个餐馆中有个空调，每分钟可以选择上调 1 个单位的温度或下调 1 个单位的温度，当然你也可以选择不变，初始的温度为 m 。有 n 个食客，每个食客会在 t_i 时间点到达，他所能适应的最低温度是 l_i ，最高温度是 h_i ，他只会在 t_i 时刻逗留。

如果温度不在食客的适应范围内，他就会不舒服，请你判断，空调能否使得 n 位来就餐的食客都感到舒服。

输入格式

本题采用多测。

第一行 T 表示测试用例个数。

对于每个测试样例，首先两个正整数 n, m，分别表示来就就餐的人数和初始温度（时间为 0 的时候的温度）。

接下来 n 行，第 i 行三个整数 t_i, l_i, h_i，如题意所述。

输出格式

对于每个测试用例，如果不能让所有食客都满意，则输出 "NO"(不带引号)。

否则，输出 "YES"(不带引号)。

C

样例

Input 1

```
4
3 0
5 1 2
7 3 5
10 -1 0
2 12
5 7 10
10 16 20
3 -100
100 0 0
100 -50 50
200 100 100
1 100
99 -100 0
```

Output 1

```
YES
NO
YES
NO
```

C

提示说明

Constraints

子任务	$\sum n \leq$	$t_i \leq$	特殊性质	分值
1	100	10^9	N/A	10
2	1000			20
4	10^5	10^3		
5		10^9		50

$-10^9 \leq m, l_i, h_i \leq 10^9$

注意温度可负。

C

核心思路

1. 温度范围的连续性：

从时间t_prev到t_curr（t_curr ＞ t_prev），温度变化的最大幅度为时间差

delta_t = t_curr − t_prev（最多升delta_t度或降delta_t度）。

因此，t_curr时刻的可能温度范围由t_prev时刻的范围推导而来：

最低可能温度：prev_low − delta_t

最高可能温度：prev_high + delta_t

2. 范围交集验证：

每个食客到达时，实际允许的温度范围是 “推导的可能范围” 与 “食客适应范围[l_i, r_i]” 的交集。若交集为空，说明无法满足该食客；否则，更新温度范围为交集，继续处理下一个食客。

C

1. 时间排序：

需按食客到达时间t_i升序排序，确保按时间顺序推导温度范围，避免时间混乱导致的错误。

解题步骤

1. 读取食客信息并按t_i排序。

2. 初始化温度范围为[m, m]（时间 0 时的温度），上一时刻tt = 0。

3. 遍历排序后的食客：

计算时间差delta_t = 当前时间 - 上一时刻。

推导当前可能的温度范围：[prev_low - delta_t, prev_high + delta_t]。

与食客适应范围[l_i, r_i]取交集，若交集为空则输出 “NO”。

否则更新温度范围为交集，继续处理。

4. 若所有食客均满足，输出 “YES”。

C


```cpp
#include<bits/stdc++.h>
using namespace std;
int q, n, m, l, r, tt, lt, ll, rr;
// 定义结构体存储每个食客的信息：到达时间t，最低适应温度l，最高适应温度r
struct node { int t, l, r; } a[100010];
// 排序函数：按到达时间t升序排列
bool cmp(node x, node y) { return x.t < y.t; }
int main() {
    cin >> q;  // 测试用例数量
    while (q--) {
        cin >> n >> m;  // n个食客，初始温度m
        l = r = m;      // 初始化温度范围为[m, m]（时间0时）
        tt = 0;         // 上一个时间点（初始为0）
        // 读取每个食客的信息
        for (int i = 1; i <= n; i++) {
            cin >> a[i].t >> a[i].l >> a[i].r;
        }
        // 按到达时间排序
        sort(a + 1, a + 1 + n, cmp);
```

C

```
        // 遍历每个食客，验证是否能满足温度要求
        for (int i = 1; i <= n; i++) {
            lt = tt;                    // 上一个时间点
            tt = a[i].t;                // 当前食客到达时间
            ll = a[i].l;                // 当前食客最低适应温度
            rr = a[i].r;                // 当前食客最高适应温
            // 计算当前可能的温度范围，并与食客适应范围取交集
            // 新的最低温度：max(上一范围最低 - 时间差，食客最低要求)
            l = max(l - (tt - lt), ll);
            // 新的最高温度：min(上一范围最高 + 时间差，食客最高要求)
            r = min(r + (tt - lt), rr)
            // 若交集为空，无法满足，标记并跳出循环
            if (l > r) {
                cout << "NO\n";
                // 跳过后续食客的处理（用i = n强制退出循环）
                i = n;
            }
        }
        // 若所有食客都满足，输出YES
        if (l <= r) cout << "YES\n";
    }
    return 0;
}
```

04

D

TEACHING
COURSEWARE

TEACH

D

限制

- 时间限制：$1s$
- 空间限制：$256M$

题目描述

期末考试快到了，文印店正在打印正在打印试题，文印店有n台打印机。第i台打印机每t_i秒可以打印一份试题，但每打印l_i份后必须停机w_i秒防止过热。即第i台打印机的工作计划为：持续工作$t_i \times l_i$秒，然后停机w_i秒，循环进行。老师同时使用所有打印机，求打印k份试题至少需要多少秒。

输入格式

- 第一行输入两个整数n和k （$1 \leq n \leq 100$，$1 \leq k \leq 10^9$）。
- 接下来n行，每行输入三个整数t_i、l_i、w_i（$1 \leq t_i, l_i, w_i \leq 10^9$），表示第$i$台打印机的参数。

输出格式

每组数据输出一行一个整数，表示打印 k 份试题至少需要的秒数。

D

输出格式

每组数据输出一行一个整数，表示打印 k 份试题至少需要的秒数。

样例1

输入

```
3 15
3 4 5
5 7 2
1 2 20
```

Copy

输出

```
25
```

Copy

样例2

输入

```
1 100
1 100 1
```

Copy

输出

```
100
```

Copy

D

样例解释

- 第一组样例中，25 秒内三台打印机分别打印 6、5、4份，总计15份。
- 第二组样例中，仅有一台打印机，打印100份需$1 \times 100 = 100$秒（无需停机，因$l_i = 1$，打印1份后停机，但$k = 100$刚好在一次工作周期内完成）。

数据分布

40% 的样例，保证 $(1 \leq n \leq 10,\ 1 \leq k \leq 1000)$。$t_i$、$l_i$、$w_i$ $(1 \leq t_i, l_i, w_i \leq 1000)$，

100% 的样例，保证 $(1 \leq n \leq 100,\ 1 \leq k \leq 10^9)$。$t_i$、$l_i$、$w_i$ $(1 \leq t_i, l_i, w_i \leq 10^9)$，

D

解题思路

1. 单调性分析：若t秒能完成打印k份，则t+1秒一定也能完成（时间越多，打印量不会减少）。这种单调性使得问题适合用二分查找求解。

2. 二分查找框架：

左边界l：初始化为 0（最少时间从 0 开始）。

右边界r：设置为一个极大值（如 1e18），确保覆盖所有可能的情况。

中间值mid：判断mid秒内是否能打印至少k份，以此收缩边界。

3. 核心验证函数（check）：

对于给定时间mid，计算每台打印机在mid秒内的打印量：

一台打印机的工作周期为T = l_i * t_i + w_i（工作l_i * t_i秒 + 停机w_i秒）。

完整周期数：mid / T，每个周期贡献l_i份，共(mid / T) * l_i份。

剩余时间：mid % T，在此时间内最多可打印min(剩余时间 / t_i, l_i)份（不超过单次连续打印上限l_i）。

所有打印机的打印量之和若≥k，则mid秒可行。

D


```cpp
#include<bits/stdc++.h>
using namespace std
const int N = 1e5 + 50;
int n, k;
int t[N], l[N], w[N];  // t[i]:每份打印时间; l[i]:连续打印份数; w[i]:停机时间
// 检查mid秒内是否能打印至少k份
bool check(long long mid) {
    long long total = 0;  // 所有打印机的总打印量
    for (int i = 1; i <= n; ++i) {
        long long cycle = 1LL * l[i] * t[i] + w[i];  // 一个工作周期的总时间（工作+停机）
        long long full_cycles = mid / cycle;  // 完整的周期数
        total += full_cycles * l[i];  // 完整周期贡献的打印量
        long long remaining = mid % cycle;  // 剩余时间（不足一个周期）
        // 剩余时间内最多打印的份数：不超过连续打印上限l[i]
        total += min(remaining / t[i], 1LL * l[i]);
        // 提前退出：若已满足k份，无需继续计算
        if (total >= k) return true;
    }
    return total >= k;
}
```

D

```
int main() {
    cin >> n >> k;
    for (int i = 1; i <= n; ++i) {
        cin >> t[i] >> l[i] >> w[i];
    }

    // 二分查找最小时间
    long long left = 0, right = 1e18;
    long long ans = right;
    while (left <= right) {
        long long mid = left + (right - left) / 2;  // 避免溢出
        if (check(mid)) {
            ans = mid;          // 记录可行时间
            right = mid - 1; // 尝试更小的时间
        } else {
            left = mid + 1;  // 时间不足，需要更大的时间
        }
    }

    cout << ans << endl;
    return 0;
}
```

04

冰雹猜想

TEACHING
COURSEWARE

TEACH

冰雹猜想

题目描述

给出一个正整数 n，然后对这个数字一直进行下面的操作：如果这个数字是奇数，那么将其乘 3 再加 1，否则除以 2。经过若干次循环后，最终都会回到 1。经过验证很大的数字（7×10^{11}）都可以按照这样的方式比变成 1，所以被称为“冰雹猜想”。例如当 n 是 20，变化的过程是 $20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$。

根据给定的数字，验证这个猜想，并从最后的 1 开始，倒序输出整个变化序列。

输入格式

输入一个正整数 n。

输出格式

输出若干个由空格隔开的正整数，表示从最后的 1 开始倒序的变化数列。

输入数据 1

```
20
```

Copy

输出数据 1

```
1 2 4 8 16 5 10 20
```

Copy

提示

数据保证，$1 \leq n \leq 100$。

冰雹猜想

1. 模拟过程：从输入的正整数 n 开始，按照冰雹猜想规则迭代计算下一个数字，直到结果为 1 时停止。

2. 存储序列：将迭代过程中产生的所有数字（包括初始数字和最终的 1）存储在数组中。

3. 倒序输出：由于题目要求从 1 回溯到初始数字，因此需要将存储的序列反向输出。

冰雹猜想


```cpp
#include<bits/stdc++.h>
using namespace std;
int a[10005];  // 存储冰雹猜想过程中的数字序列
int cnt = 0;   // 记录序列中数字的个数
int main()
{
    int n;
    cin >> n;      // 输入初始正整数n
    cnt++;         // 计数加1（初始数字加入序列）
    a[cnt] = n;    // 将初始数字存入数组
    // 按照冰雹猜想规则迭代，直到n变为1
```

冰雹猜想

```
    // 按照冰雹猜想规则迭代，直到n变为1
    while(n > 1)
    {
        cnt++;    // 每次迭代计数加1
        if(n % 2 == 0)  // 若n为偶数，除以2
        {
            n /= 2;
            a[cnt] = n;  // 存储新的数字
        }
        else  // 若n为奇数，变为3n+1
        {
            n = n * 3 + 1;
            a[cnt] = n;  // 存储新的数字
        }
    }
    // 倒序输出序列（从1到初始数字）
    for(int i = cnt ; i >= 1 ; i--)
    {
        cout << a[i] << ' ';
    }
    cout << endl;
    return 0;
}
```