T1 闰年

思路

根据从高到低的优先级进行判断，或者多个条件限制判断

参考代码

```
if (y % 172800 == 0) cout << "Yes";
else if (y % 3200 == 0) cout << "No";
else if (y % 400 == 0) cout << "Yes";
else if (y % 100 == 0) cout << "No";
else if (y % 4 == 0) cout << "Yes";
else cout << "No";
}
```

T2 天然气计价

思路

可以用一个变量 `cnt` 累计总共的用气量，再根据当前用气量计算出从第 1 个月到第 i 个月的花费 `sum2` 。除此之外，定义 `sum1` 保存前 `i-1` 个月的费用，用 `sum2-sum1` 就得到了当前第 i 月的花费。

参考代码

```
for (int i = 1; i <= 12; i++)
{
    cin >> a;
    cnt += a;   // 累计天然气总量
    if (cnt <= 310)
    {
        sum2 = cnt * 3;
    }
    else if (cnt <= 520)
    {
        sum2 = 310 * 3 + (cnt - 310) * 3.3;
    }
    else
    {
        sum2 = 310 * 3 + 210 * 3.3 + (cnt - 520) * 4.2;
    }
    // 用前i个月的费用减去前i-1个月的费用，得到第i个月的费用
    cout << fixed << setprecision(1) << sum2 - sum1 << '\n';
    sum1 = sum2;
}
```

T3 字符串

思路

开两个数组分别统计 `s` 串中每个字母出现的次数和 `t` 串中每个字母的次数。然后遍历两个数组，同一字母取数量较少的累加求和即可，因为不能添加，只能删除。

参考代码

```
for (char c : s) cnts[c-'a']++;
for (char c : t) cntt[c-'a']++;
for (int i = 0; i < 26; i++)
    ans += min(cnts[i], cntt[i]);
cout << ans;
```

T4 覆盖还原

思路

因为每个单词至少贡献一个特征字符，所以每次发现当前连续三个字符是 `boy` 中的一个，boy++，每当发现当前连续的四个字符是 `girl` 中的一个， girl++。

参考代码

```
for (int i = 0; i < s.size(); i++)
{
    if (s[i] == 'b' || s[i+1] == 'o' || s[i+2] == 'y')
    {
        boy++;
    }
    if (s[i] == 'g' || s[i+1] == 'i' || s[i+2] == 'r' || s[i+3] == 'l')
    {
        girl++;
    }
}
```

T5 藏宝洞

思路

定义结构体，按照单位价格进行结构体排序，排好序后，能将当前金币取完就全部累加，不能取完，能取多少取多少即可

参考代码

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 2e6+5;
const int inf = 1e9;
struct Gold
{
    double m, v, mv;
}golds[maxn];
bool cmp(Gold x, Gold y)
{
    return x.mv > y.mv;
}
int n, t;
double ans, res;
int main()
{
    cin >> n >> t;
    for (int i = 1; i <= n; i++)
    {
        cin >> golds[i].m >> golds[i].v;
        golds[i].mv = golds[i].v/golds[i].m;
    }
    sort(golds+1, golds+n+1, cmp);
    for (int i = 1; i <= n; i++)
    {
        if (t < golds[i].m)
        {
            res += t*golds[i].mv;
            break;
        }
        else res += golds[i].v, t -= golds[i].m;
    }
    printf("%.2lf", res);
    return 0;
}
```

复杂度分析

时间复杂度 $O(n \log n)$，空间复杂度 $O(n)$

T6 合理增肥

思路

目标很明确：

在总份数 ≤ m、每类不超过限制的前提下，让脂肪总和最大

解题步骤

① 按脂肪值排序（贪心核心）

```
按 a_i 从大到小排序
```

先选“最肥”的食物，保证收益最大

② 依次尝试选择

遍历排序后的食物：

- 如果还能吃（m > 0）
- 且该类别还没超限制（ `cnt[b_i] > 0` ）就选它！

```
选这个食物：
ans += a_i
m--
cnt[b_i]--
```

③ 直到不能再选

- 选满 m 份
- 或所有能选的都选完

为什么贪心是对的？

因为：

- 每个食物“价值”就是脂肪值，没有复杂组合关系
- 限制只是“数量限制”，不会影响单个食物价值

优先选脂肪大的，一定最优

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
int cnt[105], ans, n, m, k;
struct Food
{
    int a, b;
} foods[205];
bool cmp(Food x, Food y)
{
    return x.a > y.a;
}
int main()
{
    cin >> n >> m >> k;
    for (int i = 1; i <= k; i++) cin >> cnt[i];
    for (int i = 1; i <= n; i++)
    {
        int a, b;
        cin >> a >> b;
        foods[i] = {a, b};
    }
    sort(foods+1, foods+n+1, cmp);
    for (int i = 1; i <= n; i++)
    {
        if (m > 0 && cnt[foods[i].b] > 0)
        {
            ans += foods[i].a;
            m--, cnt[foods[i].b]--;
        }
    }
    cout << ans;
    return 0;
}
```