【位运算】响应报文时间

题目描述与示例

题目描述

1. IGMP 协议中，有一个字段称作最大响应时间(Max Response Time)， `HOST` 收到查询报文，解析出 `MaxResponseTime` 字段后，需要在 `(0，MaxResponseTime]` 时间(s)内选取随机时间回应一个响应报文，如果在随机时间内收到一个新的查询报文，则会根据两者时间的大小，选取小的一方刷新回应时间。
2. 最大响应时间有如下计算方式:

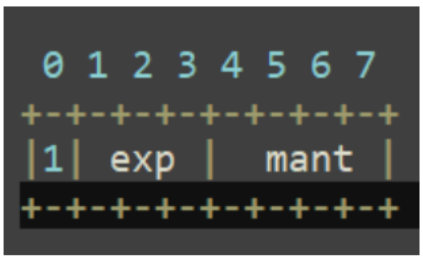
当 `MaxRespCode < 128`， `MaxRespTime = MaxRespCode`;

当 `MaxRespCode ≥ 128`， `MaxRespTime = (mant | 0x10) << (exp + 3)`;

注： `exp` 最大响应时间的高 `5~7位`； `mant` 为最大响应时间的低 `4` 位

其中接收到的 `MaxRespCode` 最大值为 `255`，以上出现所有字段均为无符号数。

现在我们认为 `HOST` 收到查询报文时，选取的随机时间必定为最大值，现给出 `HOST` 收到查询报文个数 `C`， `HOST` 收到该报文的时间 `T`，以及查询报文的最大响应时间字段值 `M`，请计算出 `HOST` 发送响应报文的时间。



输入描述

第一行为查询报文个数 `C`，后续每行分别为 `HOST` 收到报文时间 `T`，及最大响应时间 `M`，以空格分割。

输出描述

`HOST` 发送响应报文的时间。

示例一

输入

```
3
0 20
1 10
8 20
```

输出

```
11
```

说明

收到 `3` 个报文

第 `0` 秒收到第 `1` 个报文，响应时间为 `20` 秒，则要到 `0+20=20` 秒响应

第 `1` 秒收到第 `2` 个报文，响应时间为 `10` 秒，则要到 `1+10=11` 秒响应，与上面的报文的响应时间比较获得响应时间最小为 `11` 秒

第 `8` 秒收到第 `3` 个报文，响应时间为 `20` 秒，则要到 `8+20=28` 秒响应，与第上面的报文的响应时间比较获得响应时间最小为 `11` 秒

最终得到最小响应报文时间为 `11` 秒

示例二

输入

```
2
0 255
200 60
```

输出

```
260
```

说明

收到 `2` 个报文。

第 `0` 秒收到第 `1` 个报文，响应时间为 `255` 秒，则要到 `(15 | 0x10) << (7 + 3) = 31744` 秒响应(`mant = 15` ， `exp =7`)

第 `200` 秒收到第 `2` 个报文，响应时间为 `60` 秒，则要到 `200+60 = 260` 秒响应，与第上面的报文的响应时间比较获得响应时间最小为 `260` 秒

最终得到最小响应报文时间为 `260` 秒

解题思路

题意理解

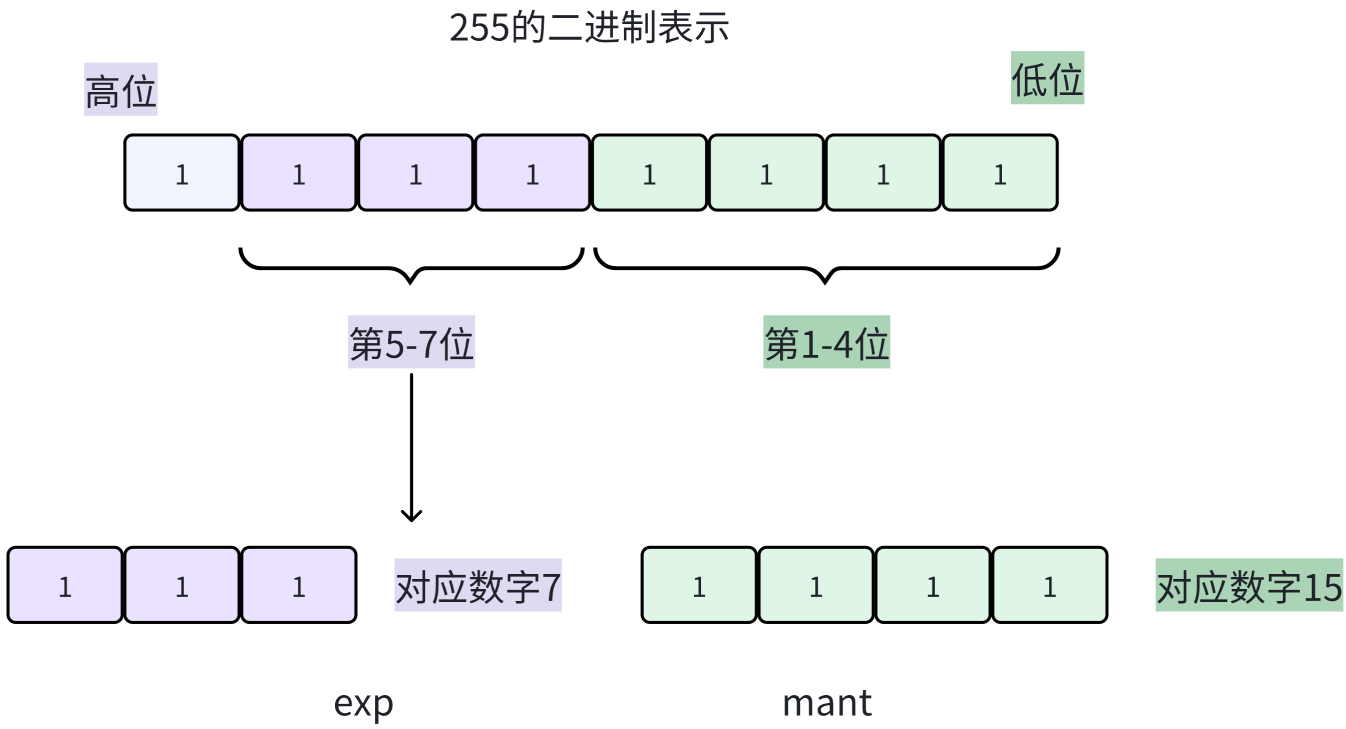
本题的难点在于根据输入的 `MaxRespCode` ，计算每一个报文的 `MaxRespTime` 。

当 `MaxRespCode < 128` 时， `MaxRespTime = MaxRespCode` 。

当 `MaxRespCode >= 128` 时， `MaxRespTime = (mant | 0x10) << (exp + 3)`

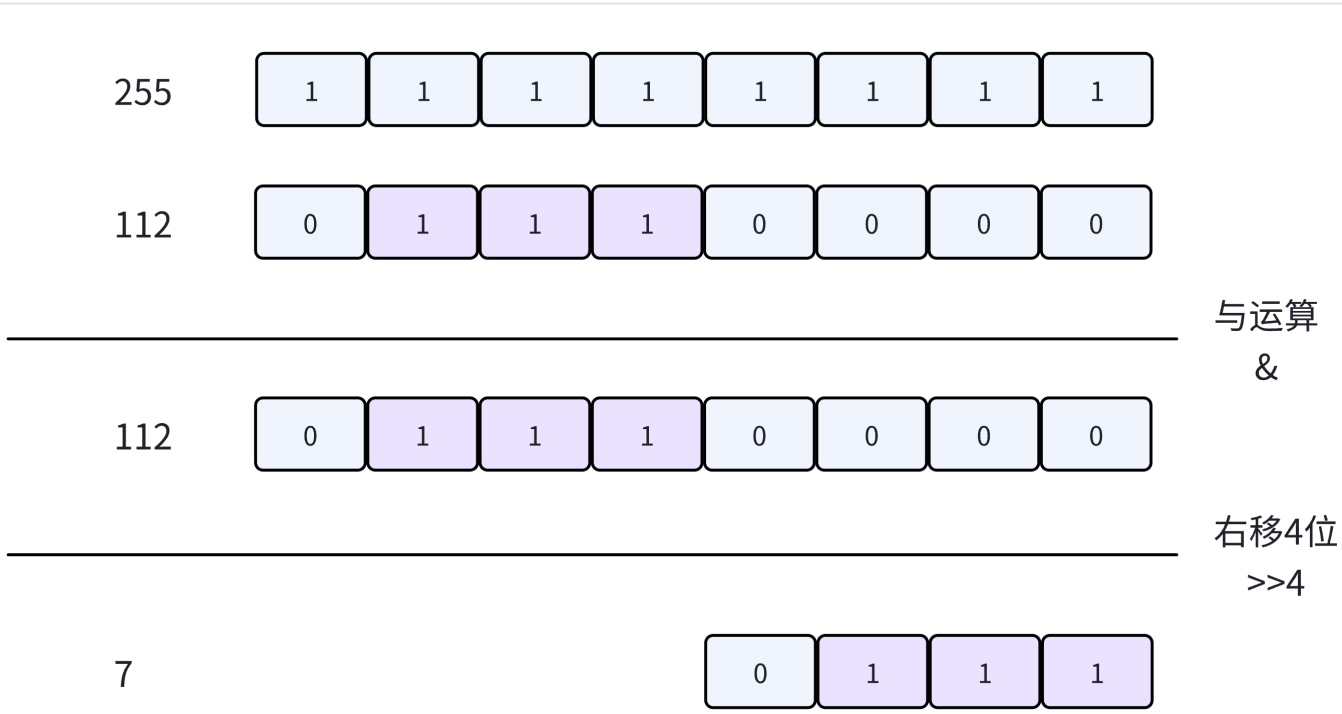
这里的 `exp` 为 `MaxRespCode` 的高 `5~7位` ； `mant` 为 `MaxRespCode` 的低 `4` 位。

上面 `exp` 和 `mant` 的计算，单看文字可能较难理解。我们以示例二中的 `255` 作为例子来进行讲解。



所以必须通过各种位运算，将 `MaxRespCode` 中的第 `5-7` 位和低 `4` 位提取出来。

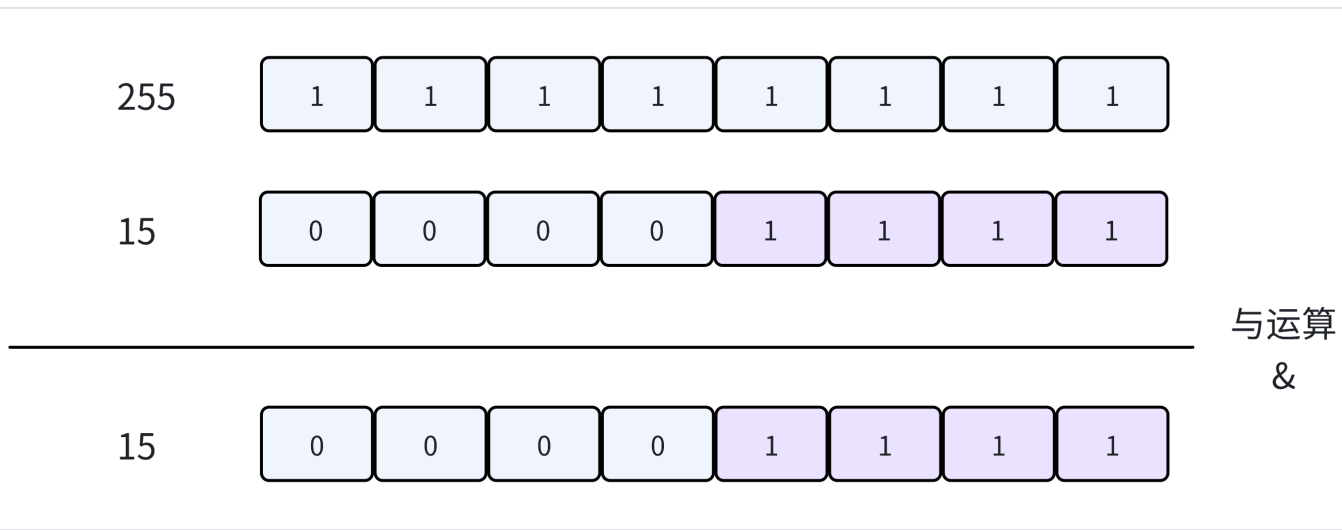
exp的提取



先对 `MaxRespCode` 和 `112` 进行**与运算**，将 `MaxRespCode` 除了 `5-7` 位外的其他位置改为 `0`，再对 `MaxRespCode & 112` 的结果进行右移 `4` 位的操作，最终得到结果。

```
1 exp = (MaxRespCode & 112) >> 4
```

mant的提取



`mant` 的提取相对简单，直接对 `MaxRespCode` 和 `15` 进行**与运算**，即可得到 `MaxRespCode` 低 `4` 位的结果。

```
mant = MaxRespCode & 15
```

代码

Python

```python
# 题目：【位运算】2024E-响应报文时间
# 分值：200
# 作者：闭着眼睛学数理化
# 算法：位运算
# 代码看不懂的地方，请直接在群上提问


from math import inf


n = int(input())
ans = inf
# 循环n次，输入n组数据
for _ in range(n):
    # 每组数据，得到收到报文时间T和最大响应码值MaxRespCode
    T, MaxRespCode = map(int, input().split())
    # 判断MaxRespCode的大小
    if MaxRespCode < 128:
        MaxRespTime = MaxRespCode
    else:
        # 解析exp和mant
        exp = (MaxRespCode & 112) >> 4
        mant = MaxRespCode & 15
        # 根据公式计算MaxRespTime
        MaxRespTime = (mant | 0x10) << (exp + 3)
    # 根据计算得到的MaxRespTime，更新答案
    # 注意此处需要加上收到报文的时间T
    ans = min(ans, T + MaxRespTime)


print(ans)

```

Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int ans = Integer.MAX_VALUE;

        for (int i = 0; i < n; i++) {
            int T = scanner.nextInt();
            int MaxRespCode = scanner.nextInt();
            int MaxRespTime;

            if (MaxRespCode < 128) {
                MaxRespTime = MaxRespCode;
            } else {
                int exp = (MaxRespCode & 112) >> 4;
                int mant = MaxRespCode & 15;
                MaxRespTime = (mant | 0x10) << (exp + 3);
            }

            ans = Math.min(ans, T + MaxRespTime);
        }

        System.out.println(ans);
    }
}

```

C++

```cpp
#include <iostream>
#include <limits>

int main() {
    int n;
    std::cin >> n;

    int ans = std::numeric_limits<int>::max();

    for (int i = 0; i < n; i++) {
        int T, MaxRespCode;
        std::cin >> T >> MaxRespCode;

```

```cpp
        int MaxRespTime;
        if (MaxRespCode < 128) {
            MaxRespTime = MaxRespCode;
        } else {
            int exp = (MaxRespCode & 112) >> 4;
            int mant = MaxRespCode & 15;
            MaxRespTime = (mant | 0x10) << (exp + 3);
        }

        ans = std::min(ans, T + MaxRespTime);
    }

    std::cout << ans << std::endl;

    return 0;
}

```

时空复杂度

时间复杂度：`O(N)`。一共有`N`组数据需要计算。

空间复杂度：`O(1)`。仅需若干常数变量。