最优资源分配

知识点数组 贪心
时间限制：1s 空间限制：32MB 限定语言：不限

题目描述：

某块业务芯片最小容量单位为1.25G，总容量为M*1.25G，对该芯片资源编号为1,2,...,M。该芯片支持3种不同的配置，分别为A、B、C。

配置A：占用容量为1.25 * 1 = 1.25G

配置B：占用容量为1.25 * 2 = 2.5G

配置C：占用容量为1.25 * 8 = 10G

某块板卡上集成了N块上述芯片，对芯片编号为1,2,...,N，各个芯片之间彼此独立，不能跨芯片占用资源。给定板卡上芯片数量N、每块芯片容量M、用户按次序配置后，请输出芯片资源占用情况，保证消耗的芯片数量最少。

资源分配规则：按照芯片编号从小到大分配所需资源，芯片上资源如果被占用标记为1，没有被占用标记为0。

用户配置序列：用户配置是按次序依次配置到芯片中，如果用户配置序列中某个配置超过了芯片总容量，丢弃该配置，继续遍历用户后续配置。

输入描述：

M：每块芯片容量为M*1.25G，取值范围为1~256
N：每块板卡包含芯片数量，取值范围为1~32
用户配置序列：例如ACABA，长度不超过1000

输出描述：

板卡上每块芯片的占用情况

补充说明：

用户配置是按次序依次配置到芯片中，如果用户配置序列中某个配置超过了芯片总容量，丢弃该配置，继续遍历用户后续配置。

示例1

输入：

8
2
ACABA

输出：

11111000
11111111

说明：

用户第1个配置A：占用第1块芯片第1个资源，芯片占用情况为：
10000000
00000000
用户第2个配置C：第1块芯片剩余8.75G，配置C容量不够，只能占用第2块芯片，芯片占用情况为：
10000000
11111111
用户第3个配置A：第1块芯片剩余8.75G，还能继续配置，占用第1块芯片第2个资源，芯片占用情况为：
11000000
11111111
用户第4个配置B：第1块芯片剩余7.5G，还能继续配置，占用第1块芯片第3/4个资源，芯片占用情况为：
11110000
11111111
用户第5个配置A：第1块芯片剩余5G，还能继续配置，占用第1块芯片第5个资源，芯片占用情况为：
11110000
11111111

示例2

输入：

8
2
ACBCB

输出：

11111000
11111111

说明：

用户第1个配置A：占用第1块芯片第1个资源，芯片占用情况为：
10000000
00000000
用户第2个配置C：第1块芯片剩余8.75G，配置C容量不够，只能占用第2块芯片，芯片占用情况为：
10000000
11111111
用户第3个配置B：第1块芯片剩余8.75G，还能继续配置，占用第1块芯片第2、3个资源，芯片占用情况为：
11100000
11111111
用户第4个配置C：芯片资源不够，丢弃配置配置，继续下一个配置，本次配置后芯片占用情况保持不变：
11100000
11111111
用户第5个配置B：第1块芯片剩余6.25G，还能继续配置，占用第1块芯片第4、5个资源，芯片占用情况为：
11111000
11111111

解题思路：

本题简单的逻辑题，没有什么难度；
解题关键：使用数组来记录各芯片的占用情况。

```
public class Main{

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int M = sc.nextInt();
        int N = sc.nextInt();
        sc.nextLine();
        String string = sc.nextLine();

        int[] chip = new int[N];       //板卡上的芯片
        int conf; //配置所占容量
        for(int i=0; i<string.length(); i++){
            char c = string.charAt(i);
            if(c == 'A'){
                conf = 1;
            }else if(c == 'B'){
                conf = 2;
            }else {
                conf = 8;
            }
            for(int j=0; j<N; j++){
                int used = chip[j];         //芯片占用的情况
                if(M - used >= conf){    //芯片所剩容量大于等于配置所占容量
                    chip[j] += conf;
                    break;
                }
            }
        }

        for(int i=0; i<N; i++){
            StringBuffer sb = new StringBuffer();
            int used = chip[i];         //芯片占用的情况
            for(int j=0; j<M; j++){
                if(j<used){         //芯片占用的地方为 1，未占用为 0
                    sb.append("1");
                }else {
                    sb.append("0");
                }
            }
            System.out.println(sb);
        }
    }
}
```