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，如果用户配置序列种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

备注

用户配置是按次序依次配置到芯片中，如果用户配置序列种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。

用例

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

输入	8 2 ACBCB
输	11111000

用例

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

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，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

题目解析

本题输出比较难以理解，我这里以用例1解释一下：

用例1的前两行输入表示：

板卡上有N=2个芯片，而每个芯片有8个单位容量，因此对应如下：

```
00000000
00000000
```

其中每个0代表一个单位容量，而一个芯片有8单位容量，因此第一排8个0代表一个芯片的总容量，第二排8个0代表另一个芯片的总容量。

理解了这个，本题就不难了。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);
```

题目解析

本题输出比较难以理解，我这里以用例1解释一下：

用例1的前两行输入表示：

板卡上有N=2个芯片，而每个芯片有8个单位容量，因此对应如下：

```
00000000
00000000
```

其中每个0代表一个单位容量，而一个芯片有8单位容量，因此第一排8个0代表一个芯片的总容量，第二排8个0代表另一个芯片的总容量。

理解了这个，本题就不难了。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 3) {
    let m = lines[0] - 0;
    let n = lines[1] - 0;
    let sequence = lines[2];

    getResult(m, n, sequence);
    lines.length = 0;
  }
});

function getResult(m, n, sequence) {
  boardCard = new Array(n).fill(0).map(() => m * 1.25);

  dict = { A: 1, B: 2, C: 8 };

  for (let i = 0; i < sequence.length; i++) {
    const need = 1.25 * dict[sequence[i]];
    for (let j = 0; j < n; j++) {
      if (boardCard[j] >= need) {
        boardCard[j] -= need;
        break;
      }
    }
  }

  boardCard.forEach((remain) => {
    unUsed = remain / 1.25;
    used = m - unUsed;
    console.log(
      new Array(used).fill(1).join("") + new Array(unUsed).fill(0).join("")
    );
  });
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;

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

    int m = sc.nextInt();
    int n = sc.nextInt();
    String sequence = sc.next();

    getResult(m, n, sequence);
  }

  public static void getResult(int m, int n, String sequence) {
    double[] boardCard = new double[n];
    Arrays.fill(boardCard, m * 1.25);

    HashMap<Character, Integer> dict = new HashMap<>();
    dict.put('A', 1);
    dict.put('B', 2);
    dict.put('C', 8);

    for (int i = 0; i < sequence.length(); i++) {
      double need = 1.25 * dict.get(sequence.charAt(i));
```

Java算法源码

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;

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

    int m = sc.nextInt();
    int n = sc.nextInt();
    String sequence = sc.next();

    getResult(m, n, sequence);
  }

  public static void getResult(int m, int n, String sequence) {
    double[] boardCard = new double[n];
    Arrays.fill(boardCard, m * 1.25);

    HashMap<Character, Integer> dict = new HashMap<>();
    dict.put('A', 1);
    dict.put('B', 2);
    dict.put('C', 8);

    for (int i = 0; i < sequence.length(); i++) {
      double need = 1.25 * dict.get(sequence.charAt(i));
      for (int j = 0; j < n; j++) {
        if (boardCard[j] >= need) {
          boardCard[j] -= need;
          break;
        }
      }
    }

    for (int i = 0; i < n; i++) {
      int unUsed = (int) (boardCard[i] / 1.25);
      int used = m - unUsed;

      StringBuilder sb = new StringBuilder();
      for (int j = 0; j < used; j++) {
        sb.append(1);
      }
      for (int k = 0; k < unUsed; k++) {
        sb.append(0);
      }
      System.out.println(sb);
    }
  }
}
```

Python算法源码

```python
# 输入获取
m = int(input())
n = int(input())
sequence = input()


# 算法入口
def getResult(m, n, sequence):
    boardCard = [m * 1.25] * n
    mapping = {"A": 1, "B": 2, "C": 8}

    for i in range(len(sequence)):
        need = 1.25 * mapping[sequence[i]]
        for j in range(n):
            if boardCard[j] >= need:
                boardCard[j] -= need
                break

    for remain in boardCard:
        unUsed = int(remain / 1.25)
        used = m - unUsed
        print('1' * used + '0' * unUsed)


# 算法调用
getResult(m, n, sequence)
```