# АНАЛИЗ НА ЗАДАЧА СПЕЦИАЛНА РЕДИЦА

Задачата се решава със симулация на действията, описани в условието. Необходими са познания и умения за съставяне условия, последователно обхождане и обработка на числови редици и за прилагане на следните алгоритми от теорията на числата, свързани с делимост:

1) Сума от цифрите на число – стандартният алгоритъм O (log $a$), където $a$ е числото
2) Брой делители на число – със сложност O ($a$) или O ($\sqrt{a}$)
3) Проверка просто ли е число – със сложност O ($a$) или O ($\sqrt{a}$).

**Предвидени са две нива на оценяване:** първото проверява знанията и уменията за правилното имплементиране на алгоритмите, а второто – уменията за реализирането им с оптимална времева сложност. Затова решения с линейна сложност също получават точки.

**Логика на решението:**

Първото картонче задължително се поставя в редичката:

```
int lastSum = sumDigs(a);
int lastCnt = cntDivs(a);
```

Останалите картончета се разглеждат последователно, като за всяко се изчисляват сумата от цифрите `currSum` и броят делители `currCnt`:

```
int cntOut = 0, cntPrimes = 0;      // нулираме броячите
for (int i = 1; i < n; ++i)
{
  cin >> a;
  int currSum = sumDigs(a);
  int currCnt = cntDivs(a);
```

Ако поне едно от двете условия е изпълнено, картончето се добавя отзад в редицата:

```
  if (currSum > lastSum || currCnt >= lastCnt)
  {
     lastSum = currSum;
     lastCnt = currCnt;
  }
```

В противен случай отстраняваме картончето:

```
  else
  {
     cntOut++;                        // броим картончето като отстранено
     if (isPrime(a)) cntPrimes++;     // ако числото a е просто, го преброяваме
  }
```

Забелязваме, че проверката просто ли е числото е след като вече знаем броя `currCnt` на делителите му. Затова е достатъчно само да проверим дали `currCnt` е 2. Така сложността по време за проверката просто ли е числото става O(1).

```
if (currCnt == 2) primes++;          // числото е просто, ако делителите му са 2
```

Накрая, след цикъла за обхождане на всички картончета, извеждаме резултата:

```
cout << removed << " " << primes << '\n';
```

Таблицата представя асимптотичната сложност по време на трите основни операции във всяко от приложените програмни решения, както и общата им времева сложност:

| Сорс код | Сума на цифрите | Брой на делителите | Просто ли е число | Обща сложност |
|---|---|---|---|---|
| `special_author.cpp` | O(log $\boldsymbol{A}$) | O ($\sqrt{\boldsymbol{A}}$) | O (1) | O ($n.\sqrt{\boldsymbol{A}}$) |
| `special_100.cpp` | O(log $\boldsymbol{A}$) | O ($\sqrt{\boldsymbol{A}}$) | O ($\sqrt{\boldsymbol{A}}$) | O ($n.\sqrt{\boldsymbol{A}}$) |
| `special_33.cpp` | O(log $\boldsymbol{A}$) | O ($\sqrt{\boldsymbol{A}}$) | O ($\boldsymbol{A}$) | O (n.$\boldsymbol{A}$) |
| `special_23.cpp` | O(log $\boldsymbol{A}$) | O ($\boldsymbol{A}$) | O (1) | O (n.$\boldsymbol{A}$) |

Тук $\boldsymbol{A}$ е стойността на максималното от числата, а $\boldsymbol{n}$ е техният брой.

*Автор: Юлия Димитрова*