# Разбор задач третьего этапа республиканской олимпиады по программированию 2010 года

## *День 1, задача 1.* Детский сад

**(автор разбора – С.И. Кашкевич)**

Самая простая из предлагавшихся на олимпиаде задач. Для её решения необходимо:

- найти минимальное и максимальное число из записанных на карточке;
- вычислить разность между этими значениями – это и будет ответ.

Сложность решения – O(N).

Как я понял из разговоров участников олимпиады, не решившие эту задачу (к счастью, таких оказалось немного) попросту искали в тексте какой-то подвох, не ожидав встретить столь простое задание…

## *День 1, задача 2.* Анализ изображения

**(автор разбора – Н.О. Лесников)**

Обойдем матрицу сначала по строкам, а затем по столбцам (или наоборот), по ходу выписывая длины всех последовательностей из подряд идущих белых пикселей, принадлежащих одной строке. Для каждой последовательности длины L существует C(L,2)=L*(L-1)/2 возможных различных пар. Нам достаточно просто просуммировать эти значения.

Пример (второй пример из условия задачи):

```
...#
.#..
...#
```

Длины последовательностей по горизонтали: 3, 1, 2, 3

Длины последовательностей по вертикали: 3, 1, 1, 3, 1

Ответ: 3*2/2 + 1*0/2 + 2*1/2 + 3*2/2 + 3*2/2 + 1*0/2 + 1*0/2 + 3*2/2 + 1*0/2 = 3 + 1 + 3 + 3 + 3 = 13

Сложность решения: O(N*M). Целочисленные типы с разрядностью свыше тридцати двух бит не требуются.

## *День 1, задача 3.* Пробное задание

**(автор разбора – С.И. Кашкевич)**

Для решения этой задачи в первую очередь требуется построить упорядоченные массивы интервалов, соответствующие каждой из входных строк. Каждый элемент такого массива соответствует одному блоку строки и содержит два числа a и b – начало и конец интервала. (Обратите внимание на то, что один или оба массива могут быть пустыми!)

Теперь задача свелась к слиянию двух упорядоченных массивов в третий, также упорядоченный. Она решается следующим образом. Пусть $(a_1, b_1)$ и $(a_2, b_2)$ – очередные элементы исходных массивов, $(a_3, b_3)$ – последний построенный элемент результирующего массива. Выбираем тот элемент (a′, b′) из исходных массивов, для которого $a' = \min(a_1, a_2)$. Если $a' <= b_3+1$, полагаем $b_3 := \max(b_3, b')$. В противном случае заносим в результирующий массив новый интервал (a′, b′).

Проиллюстрируем алгоритм слияния на втором примере из условия задачи.

| Исходные массивы | | Первый шаг слияния | | | Второй - третий шаги слияния | | | Четвертый - шестой шаги слияния | | | Седьмой - восьмой шаги слияния | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2, 4 | 1, 9 | 2, 4 | 1, 9 | 1, 9 | 2, 4 | 1, 9 | 1, 9 | 2, 4 | 1, 9 | 1, 66 | 2, 4 | 1, 9 | 1, 66 |
| 7, 7 | 41, 52 | 7, 7 | 41, 52 | | 7, 7 | 41, 52 | | 7, 7 | 41, 52 | | 7, 7 | 41, 52 | 71, 72 |
| 9, 64 | 58, 66 | 9, 64 | 58, 66 | | 9, 64 | 58, 66 | | 9, 64 | 58, 66 | | 9, 64 | 58, 66 | |
| 71, 71 | 72, 72 | 71, 71 | 72, 72 | | 71, 71 | 72, 72 | | 71, 71 | 72, 72 | | 71, 71 | 72, 72 | |
| 75, 81 | 99, 99 | 75, 81 | 99, 99 | | 75, 81 | 99, 99 | | 75, 81 | 99, 99 | | 75, 81 | 99, 99 | |

| Девятый шаг слияния | | | Десятый шаг слияния | | |
|---|---|---|---|---|---|
| 2, 4 | 1, 9 | 1, 66 | 2, 4 | 1, 9 | 1, 66 |
| 7, 7 | 41, 52 | 71, 72 | 7, 7 | 41, 52 | 71, 72 |
| 9, 64 | 58, 66 | 75, 81 | 9, 64 | 58, 66 | 75, 81 |
| 71, 71 | 72, 72 | | 71, 71 | 72, 72 | 99, 99 |
| 75, 81 | 99, 99 | | 75, 81 | 99, 99 | |

## *День 1, задача 4.* **Космодром «Центральный»**
### (автор разбора – Н.О. Лесников)

Используя O(N log N) памяти, построим структуру данных, состоящую из O(log N) массивов-слоев длины N, по следующим правилам:

1) первый слой содержит массив A[i]

2) на i-м (i >=1) слое в позициях $2^i * n$, $2^i * n + 1$, ..., $2^i * n + 2^i - 1$ находятся элементы с тех же позиций (i-1)-го слоя, отсортированные в порядке возрастания

Пример (отсортированные элементы поочерёдно выделены красным и зелёным цветом):

A = 81, 70, 52, 19, 85, 41, 78, 62, 98, 31

i=0 | 81, 70, 52, 19, 85, 41, 78, 62, 98, 31

i=1 | 70, 81, 19, 52, 41, 85, 62, 78, 31, 98

i=2 | 19, 52, 70, 81, 41, 62, 78, 85, 31, 98

i=3 | 19, 41, 52, 62, 70, 78, 81, 85, 31, 98

i=4 | 19, 31, 41, 52, 62, 70, 78, 81, 85, 98

i-й слой, иными словами, содержит подряд идущие отсортированные подмассивы длины 2^i. Такую структуру можно построить за время O(N log N) с помощью алгоритма из шага объединения сортировки слиянием (merge sort). Помимо численных значений, будем запоминать также позиции в оригинальном массиве.

Теперь мы можем последовательно брать запросы и вычислять ответы для них. Пусть текущий запрос имеет параметры S, F, R. Найдем все подмассивы верхнего слоя i (с длиной <=2^i), полностью входящие в отрезок [S; F]. Нетрудно видеть, что их будет либо нуль, либо один. Если такой подмассив найдется, вычтем отрезок, соответствующий ему, из отрезка [S; F] (при этом может образоваться нуль, один или два фрагмента), и для получившихся фрагментов осуществим подобную процедуру, используя подмассивы слоя i-1. Таким образом, мы за время O(log N) покроем отрезок [S; F] не более чем 2 log N отрезками, для которых у нас уже построены массивы с отсортированными значениями (в слоях описанной выше структуры данных). Для каждого из таких отрезков мы можем найти ближайшее к R значение с помощью двоичного поиска за время O(log L), где L - длина отрезка. Из полученных 2 log N значений мы можем выбрать оптимальное за время O(log N). Таким образом, на один запрос мы итого потратим O(log^2 N) операций (т.к. по времени исполнения доминируют O(log N) двоичных поисков по O(log L) каждый).

Пример:

```
4 9 81
Отрезок: 19, 85, 41, 78, 62, 98
Разбиение отрезка: 19 (i=0), 41 62 78 85 (i=2), 98 (i=0)
Найденные двоичным поиском позиции: 19 *, 41 62 78 * 85, * 98
Ближайшие значения: 19, 78 (|81-78| < |81-85|), 98
Ближайшее значение для всего отрезка: 78 в позиции 7
```

## *День 2, задача 1.* **Бинарные близнецы**
**(авторы разбора – С.И. Кашкевич, Н.О. Лесников)**

Для решения этой задачи достаточно построить все бинарные близнецы исходного числа N и найти наименьшее среди них.

Построим массив B из двоичных цифр числа N, дополненных ведущим нулем. Количество элементов в этом массиве равно `log(N)+1`. Для получения очередного близнеца сравниваем элементы B[i] и B[i+1] для всех допустимых i. Если они не равны, переставляем их местами, смтроим число по его двоичному представлению, восстанавливаем исходный массив.

Построим описанным алгоритмом всех бинарных близнецов для числа 11:

Исходный массив:

| 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|

Возможные перестановки:

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 13 |
| 0 | 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 1 | 1 | 19 |

(красным цветов выделены переставляемые значения, зелёным – найденный минимум).

Множество всех бинарных близнецов можно также задать формулой

```
(N & ~(3 << i)) | ((N & (1 << i)) >> 1) | ((N & (1 << (i-1))) << 1)
```

для всех i от нуля до ceil(log(N)/log(2))-1 (используется нотация языка C). В этом случае, очевидно, построение массива не нужно…

Трудоёмкость алгоритма – O(log(N)).

## *День 2, задача 2.* **Железная дорога**
**(автор разбора – С.И. Кашкевич)**

Строим массив M из 200000 элементов и вводим два индекса a и b. Вначале полагаем a=100001, b=100000. Номер каждого прицепляемого выгона заносим (используя нотацию языка C) в элемент `M[--a]` , если вагон прицепляется в начало состава, и в элемент `M[++b]` в противном случае. Тогда номер вагона, в котором находится начальник поезда, всегда хранится в элементе массива `M[(b+a)/2]`

## *День 2, задача 3.* **Лабиринт**
**(автор разбора – Н.О. Лесников)**

Рассмотрим частные случаи:

1) Задано одно направление (U, D, L или R). Если S и F лежат в одной строке или столбце, причем так, что из S в F мы попадаем после некоторого числа L шагов в разрешенном направлении по безопасным клеткам, то ответ на этот тест L 1 (очевидно, других путей нет). Согласно условию, ситуация, когда решение не существует, в тестах не встречается.

2) Задано два направления, оба горизонтальных (LR) или вертикальных (UD). Аналогично предыдущему случаю, решение единственно и представляет собой последовательность из L одинаковых ходов. Ответ - L 1.

3) Задано два перпендикулярных (UL, UR, DL, DR) или три направления (UDL, UDR, ULR, DLR). Выберем направление, по которому нам не разрешено возвращаться назад (например, для UL это либо U, либо L, для ULR - U, а для UDR - R). Предположим без ограничения общности, что мы индексируем лабиринт двумя целыми i и j, причем выбранному направлению соответствует возрастание координаты j. Тогда ответ задачи можно вычислить при помощи следующего рекуррентного соотношения:

P[i,j,l] - количество различных путей из S в (i,j) длины l без повторов клеток

(Si, Sj) - координаты S, (Fi, Fj) - координаты F

*База:*

P[i, Sj, |i-Si|] = 1, если все клетки (Si, Sj), (Si + sgn(i-Si), Sj) ... (i, Sj) безопасные, и из (Si, Sj) в (i, Sj) можно попасть, воспользовавшись разрешенными направлениями

P[i, Sj, l] = 0, для всех остальных i и l, не удовлетворяющих приведенным выше условиям

*Шаг:*

Имея значения P[i, j, l] для всех i и l для соответствующего j, можно вычислить P[i', j+1, l'] для всех подходящих i' и l' по правилу:

P[i', j+1, l'] = сумма P[i, j, l] для всех таких i и l, что клетки (i, j+1), (i+sgn(i'-i), j+1) ... (i', j+1) безопасны и доступны при помощи разрешенных направлений, а l' = l + |i' - i| + 1

***Ответ задачи:***

P[Fi, Fj, l] , для максимального l такого, что P[Fi, Fj, l] > 0