Вологодская область
III Областная олимпиада школьников по информатике
2018-2019 учебный год
9-10 классы
Заключительный тур

Разбор задач

Автор разбора: И.А. Андрианов
igand@mail.ru

Задача 1 – Змейка

Требуется получить формулу для элемента A[I][J] матрицы A размера NxN

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16
21	22	23	24	25

Ответ для строк с нечётными номерами: (I - 1) * N + J

Ответ для строк с чётными номерами: I * N - J + 1

Просуммируем эти две формулы, при этом сделав так, чтобы первая давала ноль для строк с чётными номерами, а второй – с нечётными:

((I - 1) * N + J) * (I % 2) + (I * N - J + 1) * ((I - 1) % 2)

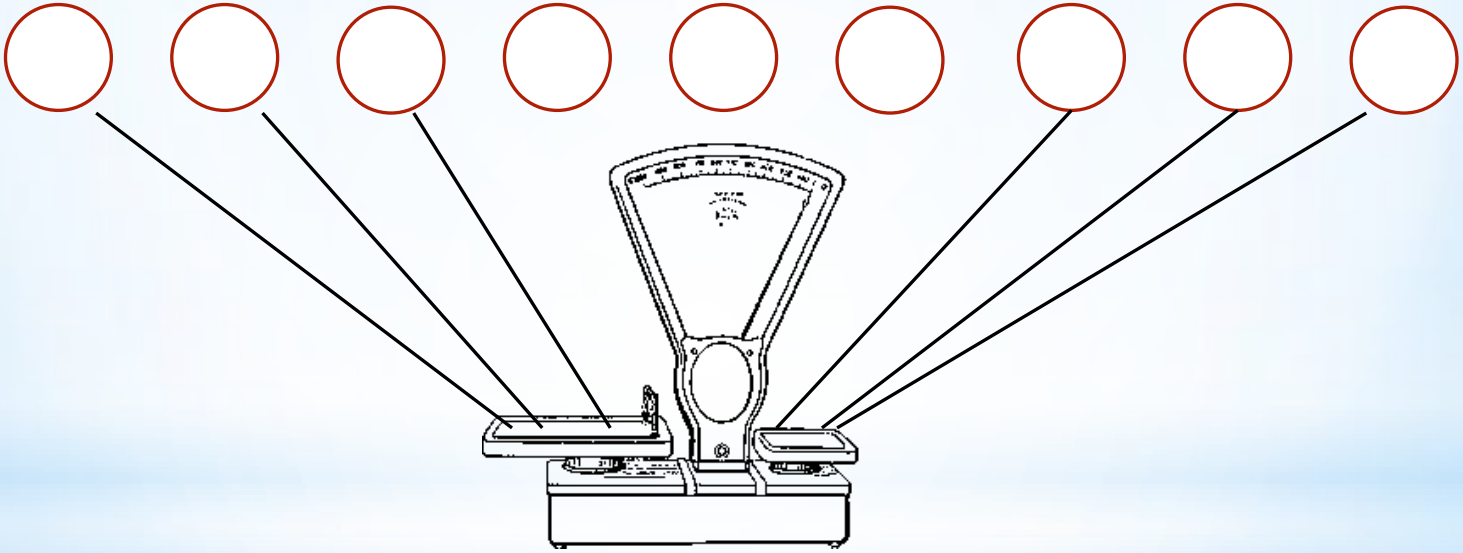
Возможны и другие верные решения

Задача 2 – Взвешивания

Ответ равен $\lceil \log_3(N) \rceil$ (округлённый вверх логарифм по основанию 3 от N).

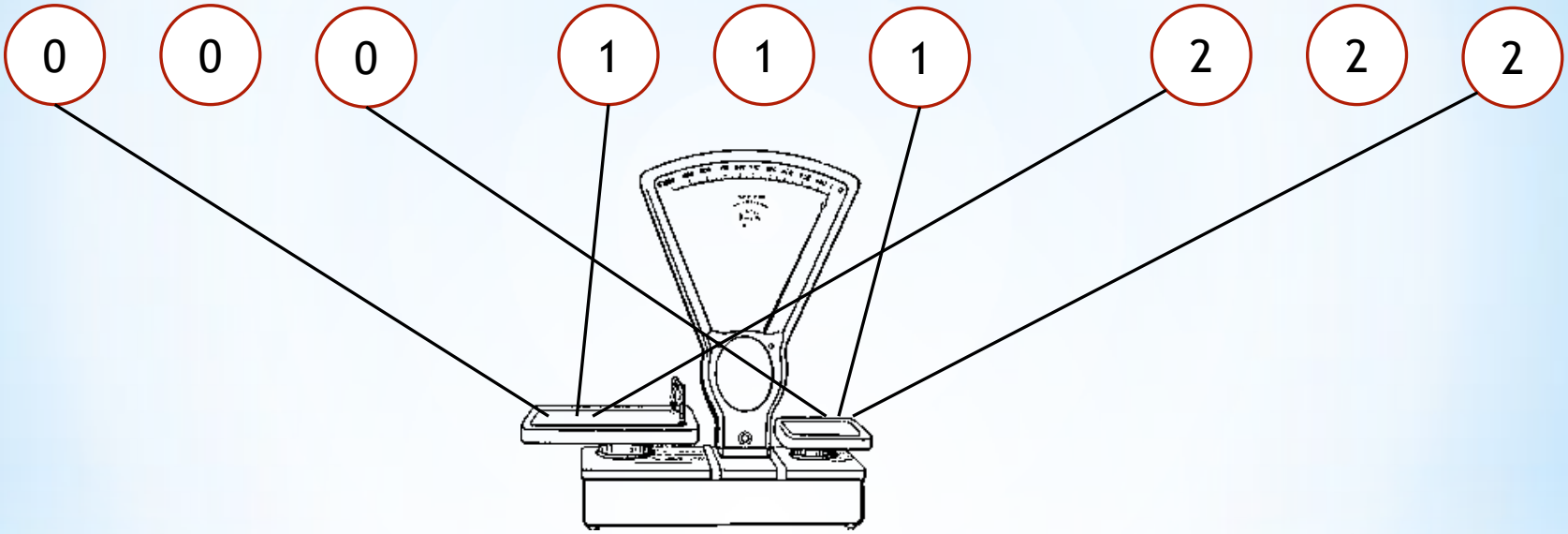
Добавим слева несколько «виртуальных» камней с весом ноль, чтобы число камней стало степенью 3.

Шаг 1. Кладём первую треть камней на левую чашу, последнюю треть – на правую. Так как такая разница весов не может получиться никаким другим способом, то это доказывает, что в первой трети – самые легкие камни, в последней – самые тяжелые.



Подпишем на камнях на левой чаше 0, на правой – 2, не на весах – 1, и вернём камни на стол. Камни разбились на 3 группы:

Шаг 2. Кладём первую треть камней из каждой группы на левую чашу, а последнюю треть из каждой группы – на правую. Поскольку такая разница весов не может получиться никаким другим способом, то первая треть каждой группы содержит самые легкие камни в этой группе, а последняя треть от каждой группы – самые тяжелые в этой группе.



Допишем на камнях на левой чаше цифру 0, на правой – 2, на столе – 1, и вернём камни на стол:

Продолжаем аналогично.

После $\log_3(N)$ таких взвешиваний каждый камень получит уникальный номер в троичной системе, причём чем меньше номер, тем меньше вес.

Почему меньшего числа взвешиваний не хватит?

Пусть проведено k взвешиваний, причём $k < [\log_3(N)]$.

Каждое взвешивание делит камни на три класса: те, что были на левой чаше, на правой и на столе. Допишем к метке каждого камня соответственно 0, 2 или 1. Всего существует 3^k троичных числа длины *k*. Так как $3^k < N$, то найдутся два камня с одинаковыми метками, которые невозможно различить между собой.

Верные ответы для заданных значений N в задаче:

2
2
4
5
17

Идея условия:
http://www.problems.ru/view_problem_details_new.php?id=98259

Задача 3 – Максимальное произведение

Подзадача 1: просто перебрать, какое число вычёркивать, посчитать произведение оставшихся чисел и выбрать лучший вариант.

Подзадача 2. Разобьём числа на отрицательные и неотрицательные, отсортируем те и другие. Рассмотрим несколько случаев:

1. Если отрицательных нечётное количество, вычеркнем самое большое отрицательное:
-3 -2 -1 4 5 → вычёркиваем -1

2. Если все числа отрицательны, и их чётное количество, вычеркнем самое маленькое отрицательное:
-4 -3 -2 -1 → вычёркиваем -4

3. Если отрицательных чисел чётное количество, и есть неотрицательные, вычеркнем самое маленькое неотрицательное:
-4 -3 -2 -1 3 5 → вычёркиваем 3

Подзадача 3.

Разобьём числа на отрицательные и неотрицательные, отсортируем те и другие. Определим знак результата: результат меньше нуля, если все числа отрицательны и (N – K) нечётно.

Если результат < 0, то вычёркиваем K самых маленьких отрицательных.

Если результат ≥ 0, то:

1. Если количество отрицательных чисел нечётно, вычеркнем самое большое из них, уменьшим K на 1 и продолжим.

2. Пока K ≥ 2, смотрим, что выгодней вычеркнуть: два самых больших отрицательных, или два самых маленьких неотрицательных. Вычёркиваем эту пару, уменьшаем K на 2. Повторяем, пока K не уменьшится до 0 или 1.

3. Если K = 1, вычеркиваем самое маленькое неотрицательное

Задача 4 – «Бери больше»

Подзадача 1: можно найти все ответы вручную.

Подзадача 2: динамическое программирование.
Пусть win[i][j] = true, если побеждает игрок, который будет сейчас делать ход. В куче лежит i спичек, предыдущий игрок на последнем ходу брал j.

Рекуррентное соотношение:

win[i][j] := (win[i – j - 1][j + 1] = false) ИЛИ (win[i – j - 2][j + 2] = false)

(если у нас есть хотя бы один ход в проигрышную для соперника позицию, то мы выигрываем)

Реализовать проще всего ”ленивым ДП” – рекурсией с запоминанием ответов в матрице.

Сложность – $O(N^2)$.

Пример решения подзадачи 2 на C++:

```cpp
#include <bits/stdc++.h>

std::vector<std::vector<int> > win;

bool can_win(int n, int prev) {
  if (prev + 1 > n) return false;
  if (win[n][prev] >= 0) return win[n][prev];
  win[n][prev] = 0;
  if (!can_win(n - prev - 1, prev + 1))
    win[n][prev] = 1;
  else if (prev + 2 <= n && !can_win(n - prev - 2, prev + 2))
    win[n][prev] = 1;
  return win[n][prev];
}

int main() {
  int n;
  std::cin >> n;
  win.assign(n + 1, std::vector<int> (n + 1, -1));
  if (can_win(n, 0))
    std::cout << "1\n";
  else
    std::cout << "2\n";
}
```

Подзадача 3: исследование ответов, полученных с помощью решения подзадачи 2. Запишем в файл ответы для N=1,2,...,100. Попробуем найти закономерность - как зависит ответ от N:

1 – 4 (всего **4** числа) - выиграет 1-й
5 – 7 (всего **3** числа) - выиграет 2-й
8 – 15 (всего **8** чисел) – выиграет 1-й
16 – 20 (всего **5** чисел) – выиграет 2-й
21 – 32 (всего **12** чисел) – выиграет 1-й
33 – 39 (всего **7** чисел) – выиграет 2-й
40 – 55 (всего **16** чисел) – выиграет 1-й
56 – 64 (всего **9** чисел) – выиграет 2-й
…

Заметим, что:
Размеры групп с выигрышем 1-го игрока: 4, 8, 12, 16, …
Размеры групп с выигрышем 2-го игрока: 3, 5, 7, 9 …
Размер первых групп увеличиваются каждый раз на 4, вторых - на 2.

Теперь несложно построить алгоритм для эффективного вычисления ответа. Для *N* до 10^9 достаточно в цикле пропускать целые группы, так как их размеры достаточно быстро растут, и в конце определить, внутри какой группы мы в итоге окажемся.

Полное решение на C++:

```
#include <bits/stdc++.h>

int solve() {
  int n;
  std::cin >> n;
  int gsize = 4, gdiff = 3;
  for(;;) {
    if (n <= gsize) return 1;
    n -= gsize;
    if (n <= gdiff) return 2;
    n -= gdiff;
    gsize += 4;
    gdiff += 2;
  }
}

int main() {
  std::cout << solve() << std::endl;
}
```

Задача 5 – Мобильная связь

Подзадача 1 - перебрать все пары деревень и станций. Не забываем, что координаты до 10^9, то есть для возведения в квадрат типа int не хватит. Нет необходимости извлекать корень, лучше сравнивать квадрат расстояния с квадратом радиуса – это быстрее и не даёт погрешностей.

Подзадача 2 ($R \le 500$**).** Переберём все деревни.

Для очередной деревни (x_i, y_i) с помощью двоичного поиска найдём самую правую станцию не правее её (то есть с координатой $x \le x_i$).

Используя её в качестве правой границы, ещё одним двоичным поиском найдём самую левую станцию s_l, ещё дотягивающуюся до данной деревни.

Аналогичными двумя шагами найдём самую правую станцию s_r, ещё дотягивающуюся до деревни.

Поскольку в подзадаче 2 радиус не превышает 500, то всего между s_l и s_r не более 1000 станций. Для каждой такой станции *s* увеличиваем счётчик cnt[*s*] на 1.

Число итераций в худшем случае – порядка 100 миллионов.

Подзадача 3.

Ускорим решение подзадачи 2 – для этого устраним цикл от s_l до s_r, который увеличивает счётчики станций. Вместо того, чтобы прибавлять единицу к $cnt[s_l]$, $cnt[s_l+1]$, $cnt[s_l+2]$, …, $cnt[s_r]$, мы будем делать только две операции:

$$cnt[s_l]{+}{+}$$

$$cnt[s_r+1]{-}{-}$$

Восстановить ответ по такому массиву по окончании вычислений очень просто:

```
int cur = 0;
for (int i = 0; i < n; i++) {
  cur += count[i];
  std::cout << cur << '\n';
}
```

Сложность решения - $O(M \cdot \log(N))$

Литература и web-ресурсы для подготовки

1. Дистанционный практикум по программированию ВоГУ: http://avt.vogu35.ru/acm
2. Вологодские олимпиады студентов и школьников: https://olympiads.vogu35.ru/
2. Школа программиста: http://acmp.ru/
3. Алгоритмы на e-maxx: http://e-maxx.ru/algo/
4. Базовые алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/997/313/info
5. Базовые и "продвинутые" алгоритмы для школьников (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/998/312/info
6. "Продвинутые" алгоритмы для школьников" (учебный курс, видеолекции): http://www.intuit.ru/studies/courses/975/311/info
7. Олимпиадное программирование с нуля на Java: https://vk.com/ol_prog_0
8. Проект "3.5 задачи в неделю“: http://codeforces.com/blog/entry/20066
9. Соревнования по программированию на Codeforces: http://codeforces.com
10. Шень, А. “Программирование: теоремы и задачи”: http://www.e-academy7.narod.ru/COURSES/PROGRAM/LITERATURA/01shen.PDF
11. Меньшиков, Ф. В. Олимпиадные задачи по программированию/ Меньшиков, Федор Владимирович. - Москва: Питер, 2006. - 315 с.
12. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В. М. Гуровица и В. А. Матюхина—М.: МЦНМО, 2006.— 256 с
13. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен , Ч. Лейзерсон, Р. Ривест, К. Штейн; пер. с англ.; 3-е изд. - Москва: ООО "И.Д. "Вильямс", 2013. - 1328 с.
14. Скиена С.С., Ревилла М.А. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. – М.: Кудиц-образ, 2005. – 416 с.

А также многое-многое другое…