

Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2014–2015 учебном году
10 класс

Время выполнения задач — 4 часа
Ограничение по времени — 2 секунды на тест
Ограничение по памяти — 64 мегабайта

Муниципальный этап
Всероссийской олимпиады школьников
по информатике

в 2014–2015 учебном году

Разборы решений и идеи тестов

10.1. «Приятная покупка». На Новый Год родители пообещали Пете Торпыжкину купить не более двух предметов из трёх: (1) новый телевизор; (2) новый ноутбук; (3) новую игровую приставку. Петя оценил удовольствие, которое он получит от обладания каждым предметом в отдельности, а также от обладания каждой парой предметов (эти значения не учитывают удовольствие от каждой вещи в отдельности). Основываясь на этих знаниях, он высказал свои пожелания по покупке. Напишите программу, которая автоматизирует этот выбор.

Формат входа: В единственной строке перечислены через пробел шесть целых чисел — удовольствия от обладания телевизором, ноутбуком, приставкой, телевизором и ноутбуком, телевизором и приставкой, ноутбуком и приставкой (в указанном порядке). Каждое число принадлежит диапазону от -1000 до $+1000$, включая крайние значения (и да, удовольствие может быть отрицательным!).

Формат выхода: Выведите один или два номера предметов, которые Петя рекомендует купить. Если наилучшая покупка будет состоять из двух предметов, разделите номера пробелом. Если покупок с одинаковым суммарным удовольствием несколько, выведите любую из них. Кроме того, хотя бы один предмет Петя выбрать должен, чтобы не обижать родителей.

Пример 1

Вход:

10 10 10 -20 -20 -20 1

Выход:

1

Пример 2

Вход:

10 10 10 -20 5 10

Выход:

3 2

10.2. «Наибольшее подчисло». Назовём натуральное число k *подчислом* натурального числа n , если десятичная запись числа k входит в десятичную запись числа n . Например, 123 является подчислом 671235, а 27 не является подчислом 529725. Напишите программу, которая по заданному натуральному числу находит его наибольшее двузначное подчисло. Напомним, что ведущие нули в записи числа не допускаются.

Формат входа: Задано единственное целое число из диапазона от 10 до 10^9 .

Формат выхода: Выведите наибольшее двузначное подчисло исходного числа.

Пример

Вход:

529725

Выход:

97

10.3. «Подготовка к олимпиаде». Готовясь к областной олимпиаде по информатике, Петя Торопыжкин решает задачи. Пока что он взялся за задачи не слишком сложные, так что решение каждой задачи занимает ровно t минут. Но помимо решения задач у него есть и другие дела, расписание которых известно. К сожалению, решение задачи должно идти непрерывно, так что если промежуток между двумя делами равен 40 минутам, а решение одной задачи занимает 30 минут, то в этот промежуток удастся решить только одну задачу, а ещё 10 минут потратить на другие дела. По имеющемуся расписанию дел Пети найдите, сколько задач в день он сможет решить. Гарантируется, что в начале суток и в конце их Петя занят — он спит (в расписании обязательно присутствует дело, начинающееся в 00:00, и присутствует дело, оканчивающееся в 23:59).

Формат входа: В первой строке через пробел заданы длительность t (в минутах) решения одной задачи (t — целое число из диапазона от 1 до 1440) и n — количество дел у Пети в расписании ($1 \leq n \leq 1000$). В следующих n строках заданы пары моментов времени в формате `hh:mm` ($0 \leq hh \leq 23, 0 \leq mm \leq 59$) — времена начала и окончания очередного дела. Дела заданы, вообще говоря, в произвольном порядке. Никакие два дела из расписания по времени не пересекаются между собой. Если час или минуты меньше десяти, то они могут снабжаться ведущим нулём, то есть допустимы записи `9:00`, `09:00`, `9:0` и `09:0`. Петя считается занятым с начала минуты момента начала и до конца минуты момента конца.

Формат выхода: Выведите единственное целое число — количество задач, которое Петя сможет решить при указанном расписании.

Пример

<u>Вход:</u>	<u>Выход:</u>
5 3	11
11:00 23:59	
00:00 9:59	
10:30 10:30	

10.4. «Разбор предложения». Петя Торопыжкин наполовину выполнил домашнее задание по английскому языку: набрал (естественно, латиницей) и распечатал предложения, которые нужно разобрать по составу. Набор осуществлялся в одну строку моноширинным шрифтом, то есть все символы имеют одинаковую ширину, включающую в себя свободное пространство вокруг символа и равную в данном случае 6 мм. Слова в предложении разделены в точности одним пробелом; в начале и конце текста пробелов нет. Каждое предложение завершается точкой.

Дальше нужно было сделать разбор этих предложений по составу, что не являлось слишком сложной задачей, поскольку все предложения имели стандартную структуру: в начале предложения шли определения, за ними — подлежащие, потом — сказуемые, а остальные слова до конца предложения были дополнениями. При этом однородные члены предложения разделялись запятой, то есть если в

конце предыдущего слова стояла запятая, то следующее слово являлось тем же членом предложения, что и предыдущее. Пробелы между предыдущим словом и запятой не ставятся, а после запятой ставится в точности один пробел.

Тут Петю заинтересовал вопрос: а какова суммарная длина линий, которые ему нужно будет нарисовать?

Напомним, что подлежащее подчёркивается одной прямой линией, сказуемое — двумя, дополнение — штриховой линией так, что штрихи длиной в полсимвола центрированы относительно символов. Определение подчёркивается волнистой линией, но Петя вместо неё рисует «прямоугольную волну»: в начале слова рисует вертикально вниз отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вверх отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вниз отрезок на 1 мм, и т.д. Прямоугольная волна всегда заканчивается вертикальным отрезком. Знаки препинания не подчёркиваются. Например:

Beautiful flowers smell tasty, delicately.

(Везде между словами стоит ровно по одному пробелу.)

Формат входа: Единственная строка содержит текст, удовлетворяющий условию задачи. Длина текста лежит в диапазоне от 0 до 255 символов.

Формат выхода: Выведите единственное целое число — суммарную длину (в миллиметрах) линий, которые нарисует Петя.

Пример

Вход:

Beautiful flowers smell tasty, delicately.

Выход:

211

10.5. «Делимость на 4». У Пети Торопыжкина есть набор из n целых чисел. Он выбирает из них несколько так, чтобы их сумма была кратной 4. Каково максимальное значение такой суммы?

Формат входа: В первой строке через пробел задано целое число n из диапазона от 4 до 10^4 . Во второй строке через пробел задано n целых чисел, не превосходящих по модулю 10^4 .

Формат выхода: Выведите единственное целое число — максимальное значение суммы некоторых исходных чисел, кратной 4.

Пример

<u>Вход:</u>	<u>Выход:</u>
5	8
1 2 3 4 -5	

**Муниципальный этап Всероссийской олимпиады
школьников по информатике
в 2014–2015 учебном году
10 класс. Разбор решений и идеи тестов**

Далее в разборах используются обозначения типов: `short int` — 2-байтовый целый тип (`integer` в TurboPascal/BorlandPascal и `int` в BorlandC++), `int` — 4-байтовый целый тип (`longint` в TurboPascal/BorlandPascal/FreePascal, `long int` в BorlandC++, `integer` в Delphi/PascalABC и `int` в VisualC++/C++ Builder), `int64` — 8-байтный целый (отсутствует в BorlandPascal и BorlandC++, но присутствует в системах программирования под Windows).

Задачи оцениваются по 100 баллов за каждую; стало быть, полная стоимость пакета — 500 баллов. В рамках одной задачи все тесты считать равноценными.

10.1. «Приятная покупка». *На Новый Год родители пообещали Пете Торопыжжину купить не более двух предметов из трёх: (1) новый телевизор; (2) новый ноутбук; (3) новую игровую приставку. Петя оценил удовольствие, которое он получит от обладания каждым предметом в отдельности, а также от обладания каждой парой предметов (эти значения не учитывают удовольствие от каждой вещи в отдельности). Основываясь на этих знаниях, он высказал свои пожелания по покупке. Напишите программу, которая автоматизирует этот выбор.*

Данная задача позиционируется программным комитетом как простая: в основе лежит аккуратный поиск максимума из шести чисел. Единственная тонкость: при подсчёте удовольствия от обладания двумя предметами нужно не забыть учесть удовольствие от обладания каждым предметом в отдельности. Разумнее всего реализовывать проверку большим ветвлением.

В описании тестов символами $P(x)$ и $P(y, z)$ описывается величина удовольствия от обладания предметами, указанными в скобках.

Идеи тестов:

1. $P(1) > 0$ и больше всех остальных величин;
2. $P(2) > 0$ и больше всех остальных величин;
3. $P(3) > 0$ и больше всех остальных величин;
4. $P(1, 2) > 0$ и больше всех остальных величин;
5. $P(2, 3) > 0$ и больше всех остальных величин;
6. $P(1, 3) > 0$ и больше всех остальных величин;
7. $P(1) < 0$ и больше всех остальных величин;
8. $P(2) < 0$ и больше всех остальных величин;
9. $P(3) < 0$ и больше всех остальных величин;
10. $P(1, 2) < 0$ и больше всех остальных величин;
11. $P(2, 3) < 0$ и больше всех остальных величин;

12. $P(1, 3) < 0$ и больше всех остальных величин;
13. $P(1) = P(2)$ и больше всех остальных величин;
14. $P(1) = P(3)$ и больше всех остальных величин;
15. $P(2) = P(3)$ и больше всех остальных величин;
16. $P(1) = P(1, 2)$ и больше всех остальных величин;
17. $P(1) = P(2, 3)$ и больше всех остальных величин;
18. $P(1) = P(1, 3)$ и больше всех остальных величин;
19. $P(2) = P(1, 2)$ и больше всех остальных величин;
20. $P(2) = P(2, 3)$ и больше всех остальных величин;
21. $P(2) = P(1, 3)$ и больше всех остальных величин;
22. $P(3) = P(1, 2)$ и больше всех остальных величин;
23. $P(3) = P(2, 3)$ и больше всех остальных величин;
24. $P(3) = P(1, 3)$ и больше всех остальных величин;
25. все величины равны.

10.2. «Наибольшее подчисло». *Назовём натуральное число k подчислом натурального числа n , если десятичная запись числа k входит в десятичную запись числа n . Например, 123 является подчислом 671235, а 27 не является подчислом 529725. Напишите программу, которая по заданному натуральному числу находит его наибольшее двузначное подчисло. Напомним, что ведущие нули в записи числа не допускаются.*

Программный комитет считает эту задачу достаточно лёгкой идеологически и не слишком тяжёлой с технической точки зрения.

Понятно, что основной здесь является процедура извлечения двузначного числа из десятичной записи натурального числа. Если число хранится в виде строки, то извлечение просто тривиально: $(\text{str}[k] - \text{ord}('0')) \cdot 10 + (\text{str}[k+1] - \text{ord}('0'))$, где `ord` — функция получения кода символа, а индекс k меняется от 1 до длины строки `str`, уменьшенной на 1 (при отсчёте индексов в строке от 1).

Если же число хранится в целой переменной, то эта процедура становится чуть более сложной. Во-первых, надо вычислить количество цифр в числе, например, так:

```
len := 0;
while m > 0 do begin
  len := len + 1;
  m := m div 10;
end;
```

Единственно, само число при этом «портится», так что надо в переменную `m` скопировать значение анализируемого числа. Теперь $k = 1$ означает позицию единиц числа, а $k = \text{len}$ означает старший разряд числа. Выделить двузначное число, состоящее из цифр в разрядах k и $k - 1$ числа m , можно посредством вычисления $(m \text{ div } 10^k) \bmod 100$. Заметим, что поскольку нам нужно перебирать все значения

индекса k , то степень 10^k можно не вычислять каждый раз, а накапливать, на каждой итерации цикла умножая на 10 соответствующую переменную.

Идеи тестов:

- 1–3. различные двузначные числа без нулей в десятичной записи;
- 4–6. различные трёхзначные числа без нулей в десятичной записи;
7. число 1111111;
8. число 9999999;
9. число 102030409;
10. минимальный тест 10;
11. максимальный тест 1000000000;
- 12–20. случайные тесты.

10.3. «Подготовка к олимпиаде». *Готовясь к областной олимпиаде по информатике, Петя Торопыжкин решает задачи. Пока что он взялся за задачи не слишком сложные, так что решение каждой задачи занимает ровно t минут. Но помимо решения задач у него есть и другие дела, расписание которых известно. К сожалению, решение задачи должно идти непрерывно, так что если промежуток между двумя делами равен 40 минутам, а решение одной задачи занимает 30 минут, то в этот промежуток удастся решить только одну задачу, а ещё 10 минут потратить на другие дела. По имеющемуся расписанию дел Пети найдите, сколько задач в день он сможет решить. Гарантируется, что в начале суток и в конце их Петя занят — он спит (в расписании обязательно присутствует дело, начинающееся в 00:00, и присутствует дело, оканчивающееся в 23:59).*

Программный комитет считает данную задачу имеющей среднюю сложность: идея решения в целом понятна, но для её реализации требуется определённая аккуратность.

Для начала следует отметить, что нужно организовать аккуратный ввод данных. Для тех, кто работает на языках C/C++, Java или Python, тут особой проблемы не будет — встроенные средства достаточно мощные, чтобы разобрать числа из указанного представления или чтобы разобрать входную строку на отдельные подстроки, представляющие числа. Однако участникам, использующих BASIC или Паскаль, тут надо будет немного потрудиться.

Геометрическая интерпретация этой задачи достаточно прямолинейна: имеется набор непересекающихся отрезков на прямой. Вопрос: сколько непересекающихся отрезков заданной длины t можно вставить между ними? Для соседних отрезков $[a, b]$ и $[c, d]$ ($b \leq c$) ответ очевиден: в промежуток между ними можно вставить $\lfloor (c - b) / t \rfloor$ непересекающихся отрезков длины t . Здесь $\lfloor \cdot \rfloor$ — операция округления вниз.

Таким образом, сначала нужно расположить отрезки дел из расписания в порядке слева направо, то есть отсортировать их по возрастанию, например, левых концов. При этом для удобства границы отрезков можно из формата hh:mm «часы–минуты» переводить в минуты $M = 60 \cdot \text{hh} + \text{mm}$. Затем, аккуратно вычисляя длины промежутков между делами (в силу предложенного представления дел из разности начала следующего дела и конца предыдущего надо ещё вычесть 1 для получения количества минут в интервале между этими делами) и находя, как указано выше, количество отрезков длины t , уместящихся в очередном интервале, суммируем эти количества и получаем ответ.

Определённой тонкостью здесь является сортировка сложных объектов (пар целых чисел) по какой-то их составляющей (по моменту начала или конца).

Идеи тестов:

1. минимальный тест, $n = 1$;
2. минимальный тест, $n = 2$; промежуток между делами существенно маленький;
3. минимальный тест, $n = 2$; промежуток между делами равен $t - 1$;
4. минимальный тест, $n = 2$; промежуток между делами равен t ;
5. минимальный тест, $n = 2$; промежуток между делами равен примерно $1.5t$;
6. минимальный тест, $n = 2$; промежуток между делами равен $2t - 1$;
7. минимальный тест, $n = 2$; промежуток между делами равен $2t$;
8. минимальный тест, $n = 2$; промежуток между делами достаточно большой;
9. $n = 3$, суммарная длина промежутков меньше t ;
10. $n = 3$, суммарная длина промежутков равна t ;
11. $n = 3$, суммарная длина промежутков меньше $2t$; один промежуток короткий;
12. $n = 3$, один промежуток имеет длину t , другой $2t$;
13. $n = 3$, оба промежутка имеют длину между t и $2t$;
14. максимальный тест, $n > 500$, нет промежутков длины t или более;
15. максимальный тест, $n > 500$, есть промежутки длины t или более;
16. $t = 1440$;
17. два одноминутных дела в начале и конце суток, $t = 1439$;
- 18–25. случайные тесты.

10.4. «Разбор предложения». *Петя Торопыжкин наполовину выполнил домашнее задание по английскому языку: набрал (естественно, латиницей) и распечатал предложения, которые нужно разобрать по составу. Набор осуществлялся в одну строку моноширинным шрифтом, то есть все символы имеют одинаковую ширину, включающую в себя свободное пространство вокруг символа и равную в данном случае 6 мм. Слова в предложении разделены в точности*

одним пробелом; в начале и конце текста пробелов нет. Каждое предложение завершается точкой.

Дальше нужно было сделать разбор этих предложений по составу, что не являлось слишком сложной задачей, поскольку все предложения имели стандартную структуру: в начале предложения или определения, за ними — подлежащие, потом — сказуемые, а остальные слова до конца предложения были дополнениями. При этом однородные члены предложения разделялись запятой, то есть если в конце предыдущего слова стояла запятая, то следующее слово являлось тем же членом предложения, что и предыдущее. Пробелы между предыдущим словом и запятой не ставятся, а после запятой ставится в точности один пробел.

Тут Петю заинтересовал вопрос: а какова суммарная длина линий, которые ему нужно будет нарисовать?

Напомним, что подлежащее подчёркивается одной прямой линией, сказуемое — двумя, дополнение — штриховой линией так, что штрихи длиной в полсимвола центрированы относительно символов. Определение подчёркивается волнистой линией, но Петя вместо неё рисует «прямоугольную волну»: в начале слова рисует вертикально вниз отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вверх отрезок на 1 мм, затем горизонтальный отрезок на ширину символа, затем вертикально вниз отрезок на 1 мм, и т.д. Прямоугольная волна всегда заканчивается вертикальным отрезком. Знаки препинания не подчёркиваются. Например:

Beautiful flowers smell tasty, delicately.

(Везде между словами стоит ровно по одному пробелу.)

По мнению программного комитета, данная задача имеет сложность чуть выше средней. Для её решения требуется аккуратная работа со строками. По сути, наиболее разумный алгоритм, решающий задачу, представляет собой конечный автомат.

Заметим, что если слово имеет длину в k символов (без учёта знака препинания), то при его подчёркивании будет проведено линий общей длиной в $6k$ мм, если это подлежащее, $12k$ мм, если это сказуемое, $3k$ мм, если это дополнение, и $6k + (k + 1) = 7k + 1$ мм, если это определение. (В последней формуле $6k$ — это длина всех горизонтальных отрезков, а $k + 1$ — длина всех вертикальных отрезков.)

При обработке строки используется переменная, например, `state`, отвечающая за ожидаемый тип следующего слова, которая может принимать значение «определение», «подлежащее», «сказуемое», «дополнение». Например, этим величинами можно сопоставить в соответствие целые числа 0, 1, 2, 3. Тогда взятие следующего состояния осуществляется вычислением `state + 1` (кроме случая «следующий после дополнения», который обрабатывается отдельно).

В начале обработки строки выставляем текущее состояние в «определение».

Суммарную длину линий выставляем в нуль. Цикл:

1. если необработанная часть строки пуста, выходим из цикла;
2. считываем очередное слово;
3. если в конце есть знак препинания (запятая или точка), выкидываем его;
4. по длине полученной строки и типу члена предложения вычисляем длину линий, которые будут нарисованы при подчёркивании этого слова, и прибавляем её к суммарной длине;
5. если в конце слова не было знака препинания, переставляем ожидаемый тип члена предложения в следующее значение; переход на п.1;
6. если в конце слова была точка, переставляем ожидаемый тип члена предложения в «определение» (предложение закончилось, далее идёт новое предложение, которое начинается с определения); переход на п.1;
7. иначе (в конце слова была запятая) просто переходим к п.1 (идут однородные члены предложения, тип члена предложения не меняется).

Разделение строки по словам можно осуществить, пройдя по строке и запомнив в массиве позиции пробелов. Или же использовать функцию поиска символа (пробела) в строке, начиная с позиции, следующей за предыдущим пробелом. В обоих случаях нужна аккуратная обработка последнего слова, чтобы не выйти за границу массива, содержащего строку.

Идеи тестов:

1. минимальный тест, пустая строка;
2. одно предложение, одно слово;
3. одно предложение, два слова-определения;
4. одно предложение, два слова — определение и подлежащее;
5. одно предложение, три слова — определение и два подлежащих;
6. одно предложение, три слова — определение, подлежащее и сказуемое;
7. одно предложение, четыре слова — определение, подлежащее и два сказуемых;
8. одно предложение, четыре слова — все члены предложения по одному;
9. одно предложение, восемь слов — все члены предложения по два;
- 10–17. то же, что в тестах 1–9, только в строке имеются два предложения, одинаковых по составу;
18. одно предложение, длина строки — 255 символов;
19. два предложения, длина строки — 255 символов;
- 20–25. случайные тесты.

10.5. «Делимость на 4». У Пети Торопыжкина есть набор из n целых чисел. Он выбирает из них несколько так, чтобы их сумма была кратной 4. Каково максимальное значение такой суммы?

Данная задача позиционируется программным комитетом как сложная. Для создания полного её решения требуется привлекать идеи метода динамического программирования.

Понятно, что «наивный» подход, связанный с перебором все возможных наборов и поиском среди них того, который даёт максимальную сумму, кратную 4, не работает из-за слишком большого времени вычислений на достаточно больших входных массивах (если быть точным, то на массивах из 31-32 элементов и более длинных такой алгоритм, вообще говоря, не уложится в ограничения по времени).

Предлагается следующее решение. Заведём двумерный массив **info** с $(n+1)$ -м элементом по первому измерению (с индексами от 0 до n) и 4-мя элементами по второму (с индексами от 0 до 3). Смысл ячейки **info** $[i, j]$ — максимальное значение суммы, составленной из членов исходного массива из его начальной части до i -го элемента включительно и дающей в остатке от деления на 4 остаток j . В начале работы алгоритма занесём во все ячейки массива **info** значение, означающее $-\infty$. Например, это может быть -10^9 .

Далее проводим цикл по всем элементам входного массива **arr**. Обработка i -го элемента заключается в следующем. К этому моменту мы уже имеем информацию о суммах, выбранных из первых $(i-1)$ элементов. Мы пытаемся прибавить i -й элемент ко всем четырём суммам (со всеми возможными остатками от деления на 4) и смотрим, увеличится ли значение суммы с соответствующим остатком в результате такого прибавления. Отдельно нужно обработать случай, когда сумма с соответствующим остатком ещё не начала копиться (соответствующее значение равно $-\infty$). Если прибавление даёт улучшение, то в соответствующую ячейку **info** $[i, \cdot]$ записываем результат этого прибавления; если не даёт, то переносим значение из соответствующей ячейки **info** $[i-1, \cdot]$. Формально эту процедуру можно записать следующим образом:

1. цикл по i от 1 до n
2. цикл по j от 0 до 3
3. если **info** $[i-1, j] \neq -\infty$ (если сумма с остатком j уже накоплена), то
4. $j' := (j + \text{arr}[i]) \bmod 4$;
 (вычисляем остаток от деления на 4 накопленной суммы
 с остатком j и элемента **arr** $[i]$)
5. если **info** $[i-1, j'] \neq -\infty$
 (если сумма с остатком j' уже накоплена), то (пытаемся улучшить)
6. если **info** $[i-1, j'] < \text{info}[i-1, j] + \text{arr}[i]$, то (можно улучшить)
7. **info** $[i, j'] := \text{info}[i-1, j] + \text{arr}[i]$;
8. иначе (улучшить не получилось, переносим старый результат)
9. **info** $[i, j'] := \text{info}[i-1, j']$;
10. конец если
11. иначе
 (сумма с остатком j' ещё не накоплена, просто записываем результат)
12. **info** $[i, j'] := \text{info}[i-1, j] + \text{arr}[i]$;

13. конец если
14. конец если (сумма с остатком j ещё не накоплена, улучшать нечего)
15. конец цикла по j
16. $j' := \text{arr}[i] \bmod 4$; (пытаемся начать новую сумму)
17. **info** $[i, j'] := \max\{\text{info}[i, j'], \text{arr}[i]\}$;
18. конец цикла по i

Ответом служит величина в ячейке **info** $[n, 0]$.

Смысл идеи решения — классическое понимание принципа динамического программирования: если сумма

$$\text{arr}[k_1] + \text{arr}[k_2] + \dots + \text{arr}[k_\sigma] + \text{arr}[i]$$

даёт наибольшее значение (со своим остатком от деления на 4) по всем наборам суммируемых элементов среди начала массива **arr** до i -го элемента, то сумма

$$\text{arr}[k_1] + \text{arr}[k_2] + \dots + \text{arr}[k_\sigma]$$

должна давать наибольшее значение (со своим остатком от деления на 4) по всем наборам суммируемых элементов среди начала массива **arr** до $(i-1)$ -го элемента.

Заметим, что если у нас есть набор из 4 натуральных чисел, то в нём обязательно найдётся подмножество, сумма элементов которого кратна 4. Действительно, возможны следующие варианты, описывающие все возможности:

1. есть чётное число, кратное 4;
2. есть два чётных числа, каждое из которых не кратно 4;
3. есть одно чётное число, не кратное 4, остальные числа нечётные, хотя бы два из них дают при делении на 4 остаток 1;
4. есть одно чётное, число не кратное 4, остальные числа нечётные, хотя бы два из них дают при делении на 4 остаток 3;
5. все числа нечётные, хотя бы одно даёт при делении на 4 остаток 1 и хотя бы одно даёт остаток 3;
6. все числа нечётные, все четыре из них дают при делении на 4 остаток 1;
7. все числа нечётные, все четыре из них дают при делении на 4 остаток 3.

В каждом описаны те наборы, которые дают сумму, кратную 4.

Идеи тестов:

- 1–7. минимальный тесты, $n = 4$, тесты соответствуют случаям 1–7 из вышеперечисленного;
- 8–14. максимальный тест $n = 10^4$, тесты соответствуют случаям 1–7 из вышеперечисленного;
- 15–20. случайные тесты, $n < 30$;
- 21–25. случайные тесты, $n > 40$.

Тесты таковы, что «наивный» алгоритм, связанный с полным перебором, наберёт чуть больше 50% полного балла. В каждой группе присутствуют как тесты только с положительными числами, так и тесты с числами обоих знаков.