XXVI командный чемпионат школьников Санкт-Петербурга по программированию

28 октября 2018 года

Задача А
Валя и письмо

Автор задачи:
Степан Филиппов
Разработка задачи:
Степан Филиппов

Постановка задачи

- Есть письмо со сторонами n и m, конверт со сторонами w и h.
- Письмо можно сгибать пополам по вертикали или по горизонтали
- Нужно согнуть письмо наименьшее число раз, чтобы его можно было поместить в конверт

Решение задачи

- Письмо можно повернуть перед началом складывания
- При фиксированном повороте письма, нужно сгибать по горизонтали, пока высота письма больше высоты конверта, и по вертикали, пока ширина письма больше ширины конверта
- Вместо того, чтобы делить длину стороны письма пополам, можно умножать длину стороны конверта на два
- Посчитаем ответы для двух начальных положений письма и выберем из них минимальный

Задача B
Экскурсия

Автор задачи:
Андрей Станкевич
Разработка задачи:
Роман Коробков

Постановка задачи

- Есть n людей, m городов и k пожеланий
- Каждое пожелание имеет вид человек X хочет/не хочет посетить город Y.
- Требуется составить план экскурсии, чтобы для каждого человека были выполнены все пожелания, кроме, может быть, одного.

Решение задачи

- Рассмотрим пожелания любого человека. Сопоставим каждому пожеланию v две вершины v_{true} и v_{false}.
- v_{true} означает, что пожелание v будет выполнено, v_{false} – не будет выполнено.
- Если не выполнено пожелание v_i, то должны быть выполнены все остальные.
- Проведем рёбра-следствия из $v_{i,\ false}$ в $v_{j,\ true}$ для всех других пожеланий v_j этого человека.
- Получилось слишком много ребер, давайте уменьшим

- Чтобы не проводить ребра во все пожелания кроме текущего v, воспользуемся следующей идеей:
- Пусть P_i вершина, из которой есть ребра-следствия в первые i пожеланий $v_1, v_2, \ldots, v_i$. Тогда из P_i можно провести всего два ребра: $P_{i+1} \to P_i$ и $P_{i+1} \to v_{i+1}$, true
- Аналогично S_i – вершина, из который есть ребра-следствия во все пожелания с номером больше либо равным i.
- Тогда для пожелания нужно провести всего два ребра: $v_{i,\ false} \to P_{i-1}$ и $v_{i,\ false} \to S_{i+1}$

- Кроме того, если есть пожелание v_1 посетить город a и пожелание v_2 не посещать город a, то оба пожелания одновременно не смогут быть выполнены.
- Чтобы выполнить это условие, для каждого города t создадим две дополнительных вершины t_{true} и t_{false}.
- Тогда для пожелания v, в котором требуется посетить город t, добавим ребра $v_{true} \Leftrightarrow t_{true}$, $v_{false} \Leftrightarrow t_{false}$.
- Для пожелания v, в котором не требуется посещать город t добавим ребра $v_{true} \Leftrightarrow t_{false}$, $v_{false} \Leftrightarrow t_{true}$.
- Заметим, что суммарное число проведенных ребер $O(k + m)$

- После проведения всех ребер нужно проверить, можно ли выбрать выполненные и невыполненные пожелания так, чтобы все ребра-следствия выполнялись. Это можно сделать алгоритмом, который используется при решении задачи 2-SAT за $O(k + m)$

Задача C
Как проиграть контест

Автор задачи:
Степан Филиппов
Разработка задачи:
Степан Филиппов

Постановка задачи

- Есть n предметов с весами w_i и стоимостями c_i
- Нужно выбрать множество предметов с суммарным весом, не превосходящим W, так что нельзя добавить ни один из оставшихся предметов, не превысив ограничение на суммарный вес
- Требуется минимизировать суммарную стоимость

Решение задачи

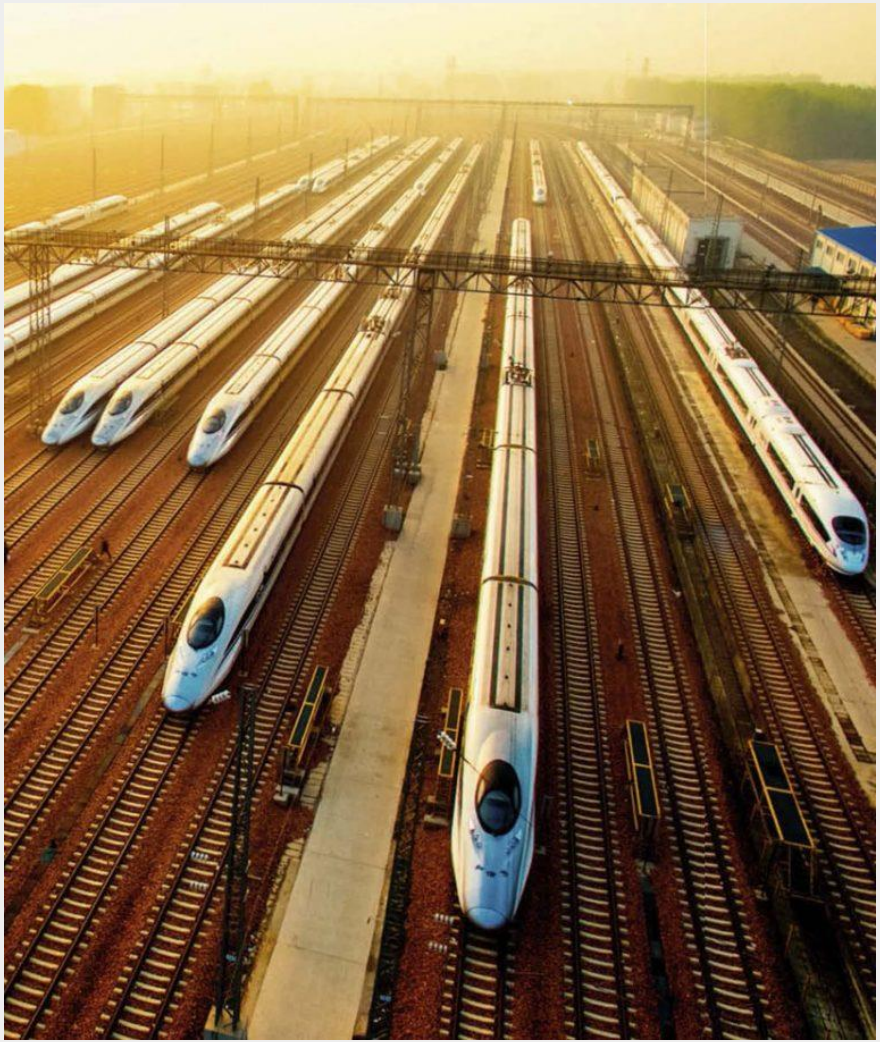
- Если сумма весов всех предметов не превосходит W, то придется взять все предметы. В этом случае ответ – это сумма стоимостей всех предметов
- В противном случае существует предмет, который не будет присутствовать в искомом множестве
- Расположим предметы по убыванию весов: $w_1 \geq w_2 \geq \ldots w_n$
- Пусть k – индекс последнего (самого легкого) предмета, который не выбран
- Тогда все предметы с большими индексами нужно взять в множество
- Их вес $s_k = w_{k+1} + w_{k+2} + \ldots + w_n$, а стоимость $c_{k+1} + c_{k+2} + \ldots + c_n$

- Из оставшихся предметов (с индексами меньше *k*) нужно выбрать множество с суммарным весом не меньше $W - s_k - w_k + 1$ и не больше $W - s_k$
- Нетрудно видеть, что множество с суммарным весом из данного диапазона в объединении с предметами $k+1, \ldots, n$ удовлетворяет требованиям задачи
- Нужно минимизировать стоимость
- “Задача о рюкзаке”

- Стандартная задача на динамическое программирование
- `dp[i][j]` – минимальная стоимость некоторых предметов среди первых `i` с суммарным весом ровно j
- $dp[i][j] = \min(dp[i-1][j], dp[i-1][j-w_i]+c_i)$
- Можно найти наименьшую стоимость множества предметов, где k – индекс последнего невзятого
- Посчитаем ответ для каждого k и выберем минимальный. Асимптотика: $O(nW)$

Задача D
Железные дороги Берляндии

Автор задачи:
Николай Будин
Разработка задачи:
Иван Сафонов



Постановка задачи

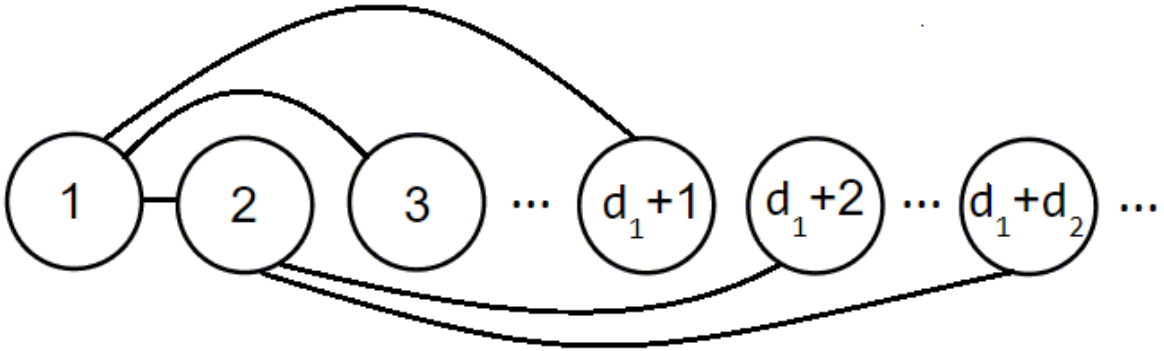
- Нужно построить дерево из n вершин, у которого степень i-й вершины равна d_i
- Среди всех таких деревьев найти дерево с минимальным диаметром

Решение задачи

- Упорядочим все степени d_i в обратном порядке, то есть будем считать, что $d_1 \geq d_2 \geq \ldots \geq d_n$
- Алгоритм построения дерева:
  - Перебираем вершины в порядке увеличения их номера
  - Рассматриваем вершину `i`. Проводим ребра из нее в такие вершины справа, в которые еще не проводили ребер до этого, пока степень вершины `i` не станет равной d_i

Первые шаги работы алгоритма

- Рассматриваем вершину 1. Из нее мы проведем ребра в вершины 2, 3, …, d_1+1
- Рассматриваем вершину 2. Из нее мы проведем ребра в вершины $d_1+2, d_1+3, \ldots, d_1+d_2$ + одно ребро уже провели до этого из вершины 1
- Рассматриваем вершину 3. Из нее мы проведем ребра в вершины $d_1+d_2+1, \ldots, d_1+d_2+d_3-1$ + одно ребро уже провели до этого
- …



- Алгоритм строит дерево, потому что $d_1+d_2+\ldots+d_n = 2n - 2$
- Можно доказать, что диаметр такого дерева минимален
- Доказательство основывается на том, что выгодно ставить вершины с наибольшими степенями как можно ближе к вершине 1, делая дерево более широким и уменьшая диаметр
- Полное доказательство остается в качестве упражнения, его можно будет прочитать в текстовом разборе
- Асимптотика: $O(n\log(n))$ или $O(n)$ (если сортировать подсчетом)

Задача E
Сложные задачи

Автор задачи:
Александра Олемская
Разработка задачи:
Александра Олемская

Постановка задачи

- Есть n задач
- Дана строка, в ней есть сколько-то латинских заглавных букв A
- На каждую задачу в строку было выписано различное ненулевое число букв A
- Между буквами A могло быть выписано сколько угодно строчных букв
- Требуется найти максимальное возможное число задач

Решение задачи

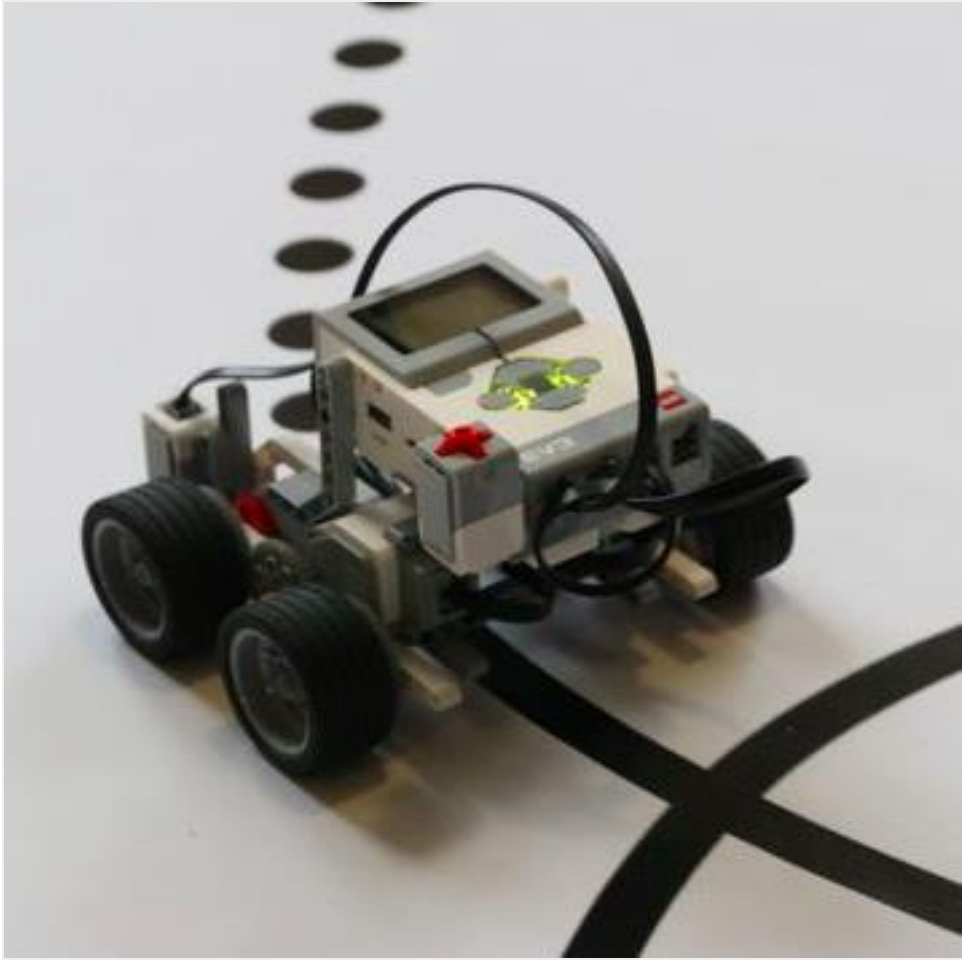
- Посчитаем число d - количество букв A в строке
- Поскольку все остальные буквы могли быть записаны в любом порядке, между любыми буквами A, то их наличие и положение в строке не играет никакой роли (для любого набора решений и букв A любое расположение решений между буквами A является возможной выписанной Тошей строкой)
- Научимся находить максимальное такое n, что могло быть выписано d букв A

- Заметим, что если выбрано число n, то выписано хотя бы 1 + 2 + … + n букв A, потому что каждая задача имеет уникальную сложность, поэтому если их упорядочить по возрастанию сложности, то за первую будет выписано хотя бы одна буква, за вторую - хотя бы две, и так далее
- Суммарно для n задач тогда будет выписано хотя бы n(n + 1)/2 букв A
- А еще заметим, что если могло быть n задач, то могло быть и n - 1, потому что можно взять сложности для n задач a_1, a_2 … a_n, и взять первые n - 2 с такими же сложностями, а последнюю - со сложностью, равной $a_{n-1} + a_n$

- Таким образом, нам достаточно найти максимальное число n такое, что $n(n + 1)/2$ не превосходит d
- Тогда будет существовать такой набор сложностей a_1, a_2 … a_n:
  - $a_i = i$ при $i < n$
  - $a_n = d - (n - 1)n/2$
- Ищем подходящее n двоичным поиском или извлекая корень из 2d и проверяя его и числа в небольшой окрестности

Задача F
Гонка роботов

Автор задачи:
Михаил Анопренко
Разработка задачи:
Михаил Анопренко

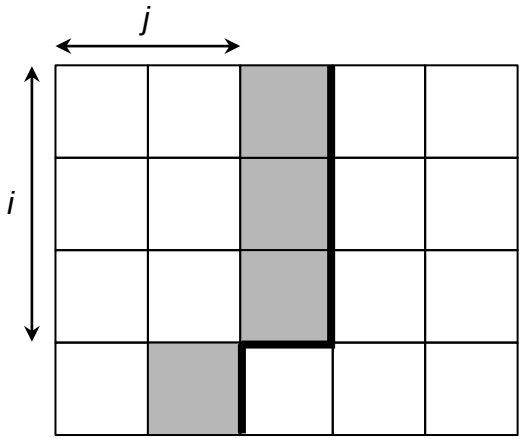


Постановка задачи

- Дано клетчатое поле. Можно перемещаться вправо и вниз. Стоимость перемещения между соседними клетками либо 0, либо 1
- Известны некоторые стоимости
- Посчитать количество способов назначить стоимости, чтобы расстояния от всех клеток левого столбца до правой нижней клетки были одинаковыми

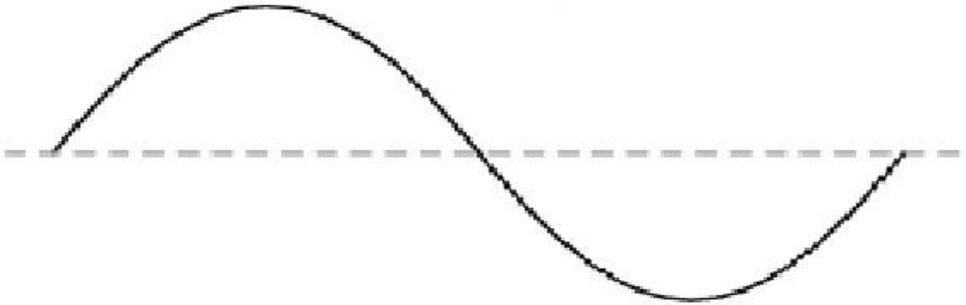
Решение задачи

- Перевернем поле. Теперь нужно сделать так, чтобы расстояния от левой верхней клетки до всех клеток правого ряда были одинаковыми
- Воспользуемся динамикой по профилю, просматривая столбцы слева направо, внутри столбцы просматривая сверху вниз. Будем считать, что все расстояния между соседними просмотренными клетками уже определены
- `dp[j][i][prof]` – просмотрели `j` столбцов, в `j`-м столбце просмотрели `i` клеток. `prof` – массив расстояний до крайних клеток в каждой строке
- Для хранения можно использовать `map<vector<int>, int>`
- При добавлении новой клетки пересчитываем расстояние до нее



Задача G
Гибкие отрезки

Автор задачи:
Дмитрий Якутов
Разработка задачи:
Дмитрий Якутов

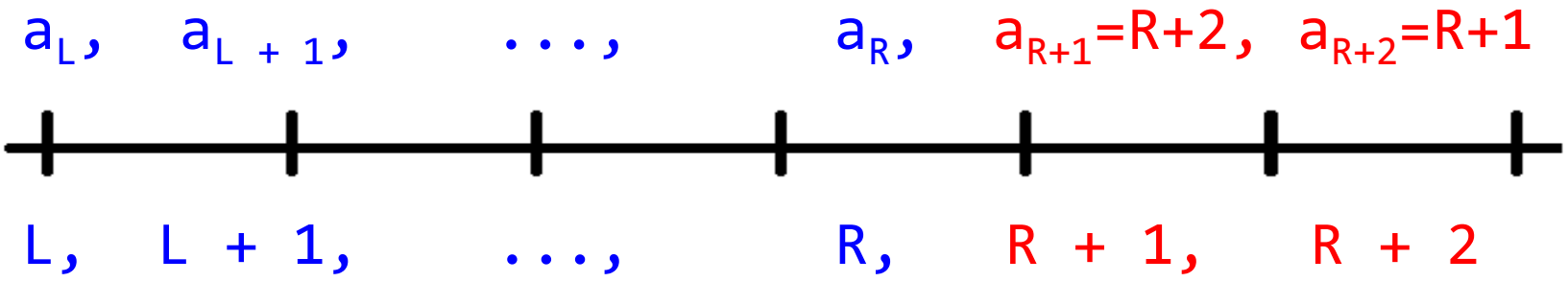


Постановка задачи

- Дано число n
- Нужно выбрать отрезок из n последовательных натуральных чисел
- Каждое число изменить ровно на 1 (вычесть 1 или прибавить 1)
- Так, чтобы произведение элементов исходной последовательности равнялось произведению элементов получившейся последовательности
- Либо сообщить, что это невозможно сделать

Решение задачи

- Если отрезок [L; R] гибкий, то [L; R + 2] – тоже



- Можно добавить к отрезку два элемента

- $n = 2$
  - Отрезок $[1; 2]$ гибкий: $a_1 = 2, a_2 = 1$
  - Произведение $1 \cdot 2 = 2 \cdot 1 = 2$
- $n = 3$
  - Отрезок $[3; 5]$ гибкий: $a_3 = 2, a_4 = 5, a_5 = 6$
  - Произведение $3 \cdot 4 \cdot 5 = 2 \cdot 5 \cdot 6 = 60$

Задача H
Секретный шифр

Автор задачи:
Антон Гардер
Разработка задачи:
Роман Коробков

Постановка задачи

- Даны некоторые цифры
- Требуется составить из них число, чтобы каждая его подстрока длины три делилась на три.

Решение задачи

- Переберем первые две цифры числа
- Если в числе уже есть хотя бы две цифры, то можно однозначно определить остаток следующей добавленной цифры по модулю 3.
- 123...ab c
- Если мы добавим c, то $(a + b + c) = 0 \pmod 3$
- Среди цифр с нужным остатком, возьмем максимальную
- Время работы – $O(digits^2 \cdot \Sigma c_i)$

Пример

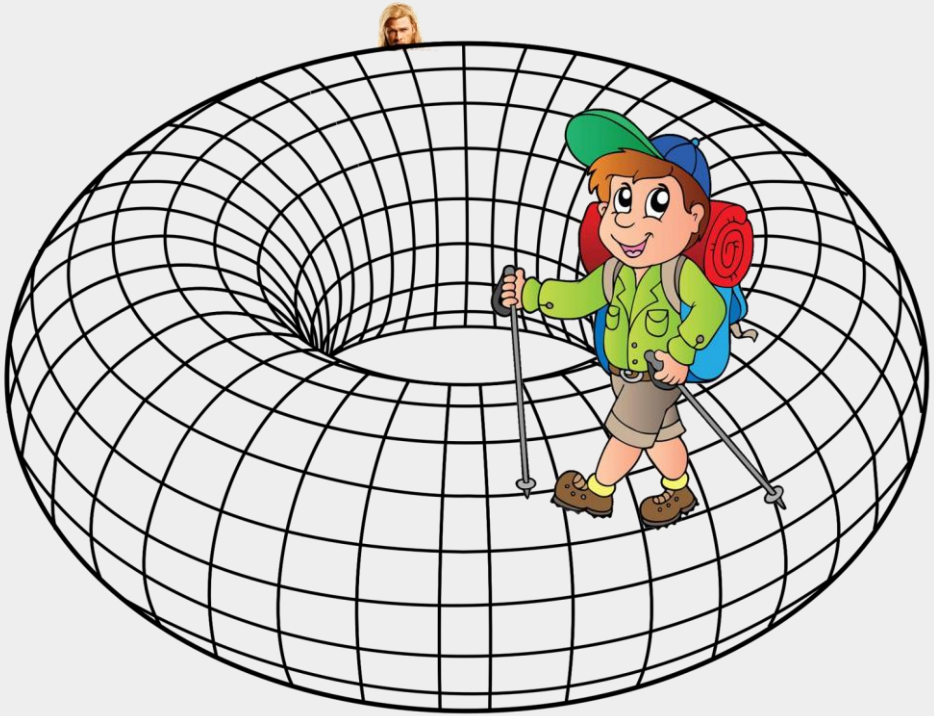
- Например, набранное число 123...34
- Сумма остатков последних двух цифр 1, следовательно, у следующей цифры остаток 2
- Есть неиспользованные 1, 3, 5 и 8
- Подходящими цифрами являются 5 и 8
- Добавим из них максимальное, то есть 8

Доказательство

- Заметим, что в числе, у которого любая подстрока длины три делится на три, можно поменять две цифры с одинаковым остатком по модулю 3, и это свойство сохранится
- Если $x < y$ и $x = y \pmod 3$, то мы можем поменять x, y местами и увеличить число
- Значит в оптимальном ответе числа одного остатка отсортированы по убыванию

Задача I
Путешествие по тору

Автор задачи:
Даниил Орешников
Разработка задачи:
Даниил Орешников



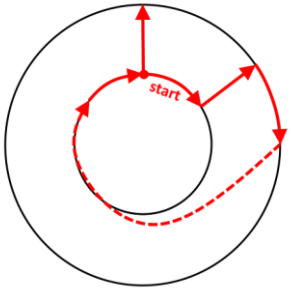
Постановка задачи

- На торе есть четыре большие окружности и n малых, по которым можно перемещаться
- Найти минимальный путь, проходящий хотя бы по четверти каждой малой окружности (хотя бы по одной дуге между пересечениями каждой малой со всеми большими окружностями)
- Малых окружностей может быть много, до 10^9

- На оптимальном пути Сеня никогда не попадает на внешнюю дорогу, так как он мог симметрично пойти на внутреннюю, и уменьшить путь
- Аналогично, из северной и южной дорог рассмотрим только южную, потому что пути по ним симметричны и не отличаются расстоянием
- На всех следующих (плоских) рисунках внешняя окружность будет обозначать южную, внутренняя внутреннюю

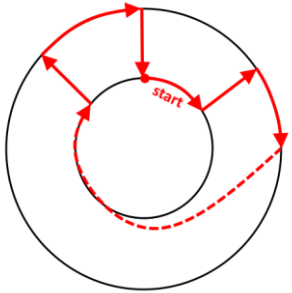
- Рассмотрим только пути, в которых Сеня движется только по часовой стрелке по большим дорогам
- В таких путях ровно n - 1 перемещение из страны в соседнюю, если Сеня начинает с перемещения в соседний город, и ровно n, если нет, так как надо вернуться в исходную страну, чтобы проехать по ней

- На путях, где Сеня двигается только в одном направлении по большим дорогам:
- Если по большим дорогам n перемещений:



- вернулись в исходный
затем пошли в соседний

+ оптимальнее было бы удалить ребро *start* и пойти по маршруту, начинающемуся во второй стране



- попали в соседний
затем вернулись в начало

+ можно снова удалить ребро *start*, а весь путь развернуть в обратном направлении

- Таким образом, все минимальные пути содержат ровно n - 1 перемещение между странами
- Выделим в каждой стране первый раз, когда мы проезжаем между ее городами - это будет сразу после попадания в каждую страну
- Все части пути между такими перемещениями назовем *заграничными*. Заграничные пути бывают четырех видов (среди тех, в которых нет двух перемещений между городами туда-обратно)

- 4 вида заграничных путей:
  - i_k -> i_{k+1}
  - s_k -> i_k -> i_{k+1}
  - s_k -> s_{k+1}
  - i_k -> s_k -> s_{k+1}
- Где i_k - город на внутренней окружности, s_k - на южной
- Все заграничные пути 4 вида можно заменить на какие-то пути вида i_k -> i_{k+1} -> s_{k+1}, что делает суммарное расстояние короче, а поэтому в минимальном пути 4 вида нет

- Тогда заграничные пути бывают трех видов:
  - i_k -> i_{k+1}
  - s_k -> i_k -> i_{k+1}
  - s_k -> s_{k+1}
- Между любыми двумя заграничными путями есть ровно одно перемещение внутри страны, после которого положение меняется с i на s или наоборот
- Поэтому после 1 вида могут быть 2 и 3, после 2 - тоже 2 и 3, а после 3 - только 1

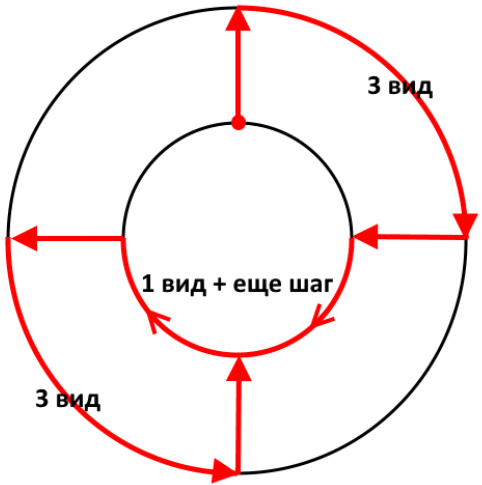
- Если то, какой заграничный путь может следовать после какого, выглядит как:
  - `1 -> 2,3`
  - `2 -> 2,3`
  - `3 -> 1`
- Тогда путь можно представить как чередование 2..2 и 31..31\313..13
  - например в 312313 это чередование (31), (2) и (313)
- При этом путь начинается с заграничного номер 3 или 2, потому что Сеня стартует из города i_1, и едет в s_1 сразу же

- Поскольку минимальный путь можно представить как чередование 2..2 и 31..1\313..3,
- Заметим, что поскольку 1 вид - самое короткое перемещение, то когда длина 3 <= длины 2, 2 в минимальный путь вообще входить не будет, и ответ будет выглядеть как 31313... с перемещениями между соседними городами i_k и s_k между каждыми 1,3 и 3,1

- Рассмотрим, когда эта формула не оптимальна
- Это случается когда длина 2 < длины 3, тогда получаем неравенство
  - $2\pi(R - r)/n + 2\pi r/4 < 2\pi R/n$
  - $r/2 < 2r/n$
  - $n < 4$
- Тогда при $n < 4$ рассмотрим каждое n отдельно. При $n = 1$ и $n = 2$ первая полученная формула дает наилучший результат. При $n = 2$ выгоднее взять путь вида 2

- Осталось доказать неоптимальность путешествий не только по или против часовой стрелки, а еще и со сменой направления
- Аналогично предыдущим случаям, выделим в пути первый раз когда Сеня проехал по какой-то стране, и все отрезки путей между ними
- Каждый из отрезков пути между проездами двух стран будет одного из рассмотренных ранее 4 видов, только дополненный лишними перемещениями между странами, по которым Сеня не проехал

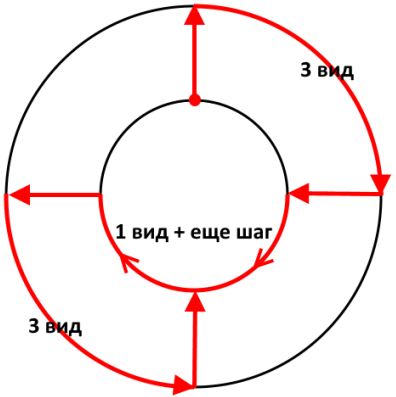
- Например, на рисунке ниже видно, как может быть устроен путь, в котором Сеня меняет свое направление движения по большим окружностям



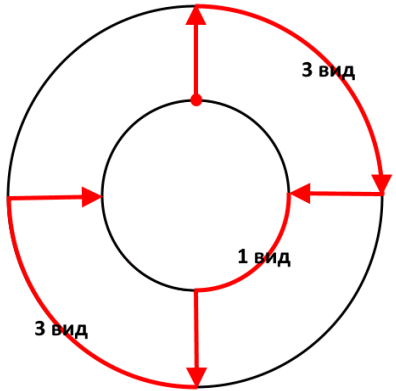
В данном примере происходит чередование 3 и 1 видов, но первый вид удлинен еще одним перемещением

Оставим ту же последовательность видов, но пройдемся по странам подряд - получим путь короче, значит этот был не минимальный

- Сделаем замену - сохраним последовательность видов заграничных путей, но будем совершать их между соседними странами в одном направлении
- Получаем путь, который короче исходного

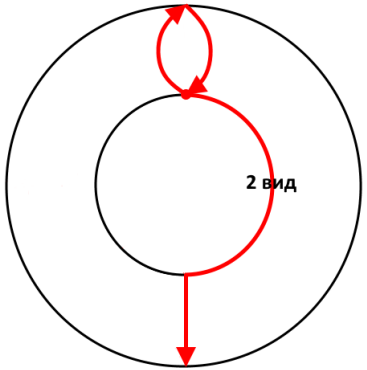


заменяем на



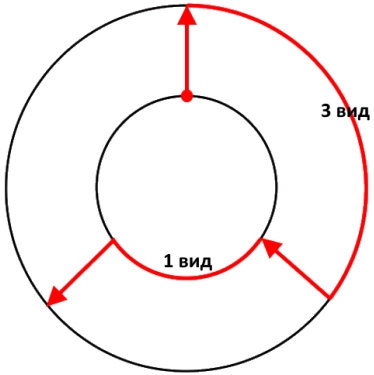
и получаем путь, который короче за счет отсутствия “лишнего” перемещения между странами

- Таким образом, наш ответ будет достигаться так:



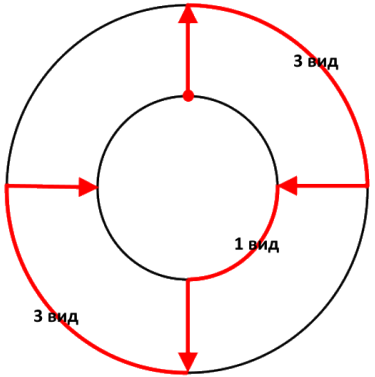
при n = 2

$\pi(R - r) + 3\pi r/2$



при n mod 2 = 1
k = (n - 1) / 2

$2k\pi R/n + 2k\pi(R - r)/n + n\pi r/2$



при n mod 2 = 0, n > 2
k = n / 2

$2k\pi R/n + 2(k - 1)\pi R/n + n\pi r/2$

Задача J
Новая прогулка Амальтеи

Автор задачи:
Александр Фёдоров
Разработка задачи:
Александра Дроздова

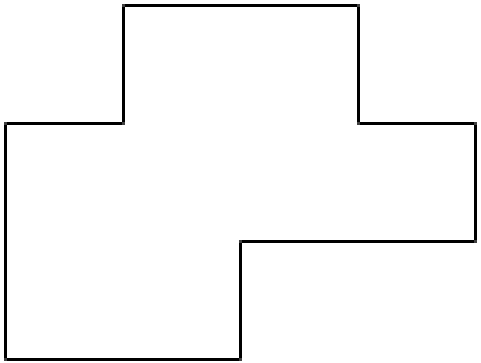
Постановка задачи

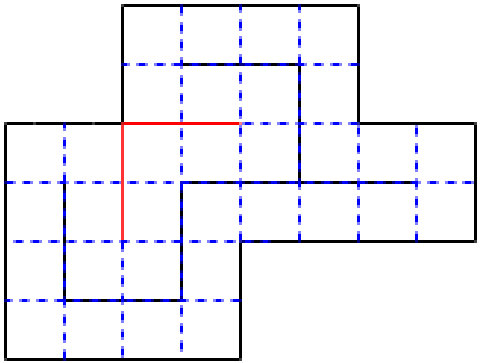
- Дана связная клетчатая фигура
- Разделим каждую клетку на четыре равных квадрата
- Требуется обойти получившиеся маленькие квадраты по циклу, побывав в каждом ровно по одному разу

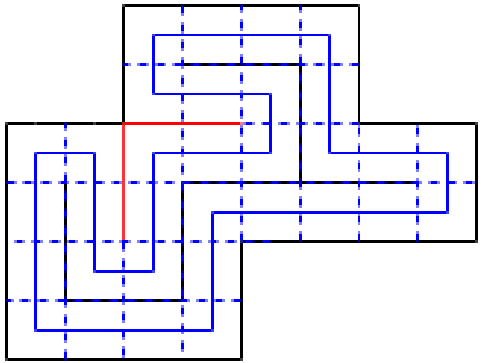
Решение задачи

- Обойдем заданную фигуру обходом в глубину
- Проведём рёбра дерева dfs как отрезки между центрами клеток
- Если две соседние клетки не соединены ребром dfs, проведем между ними отрезок-границу
- Теперь все маленькие клетки организованы ровно в один цикл

Иллюстрация





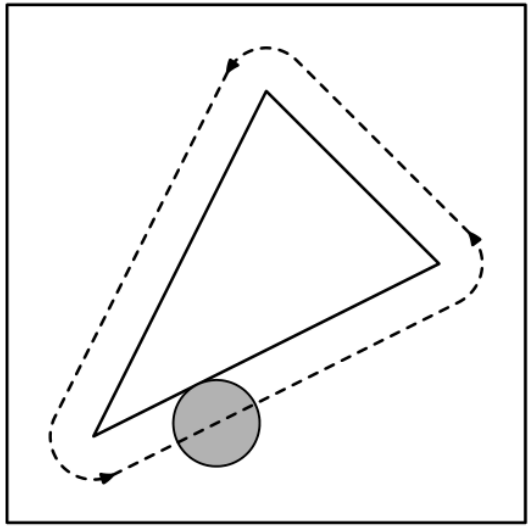


Задача K
Формула 42

Автор задачи:
Андрей Станкевич
Разработка задачи:
Николай Будин

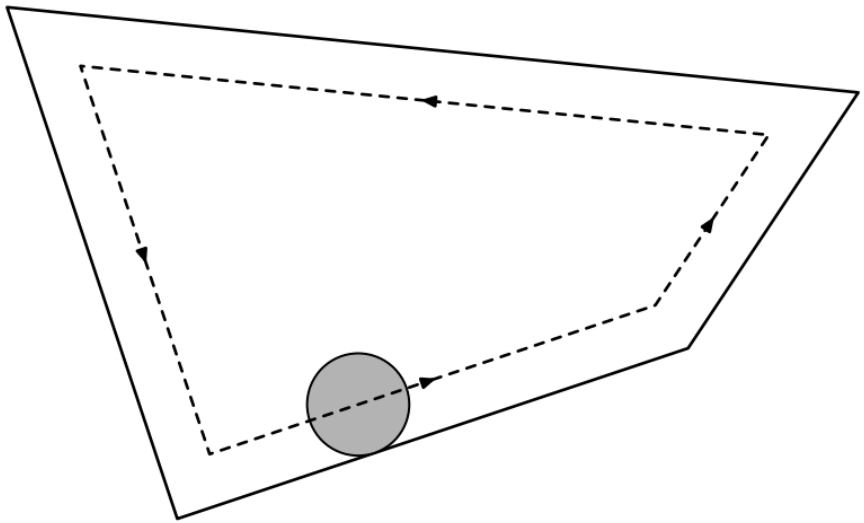
Постановка задачи

- Даны два многоугольника
- Требуется разместить один в другом, чтобы между ними мог “проехать” круг максимального радиуса

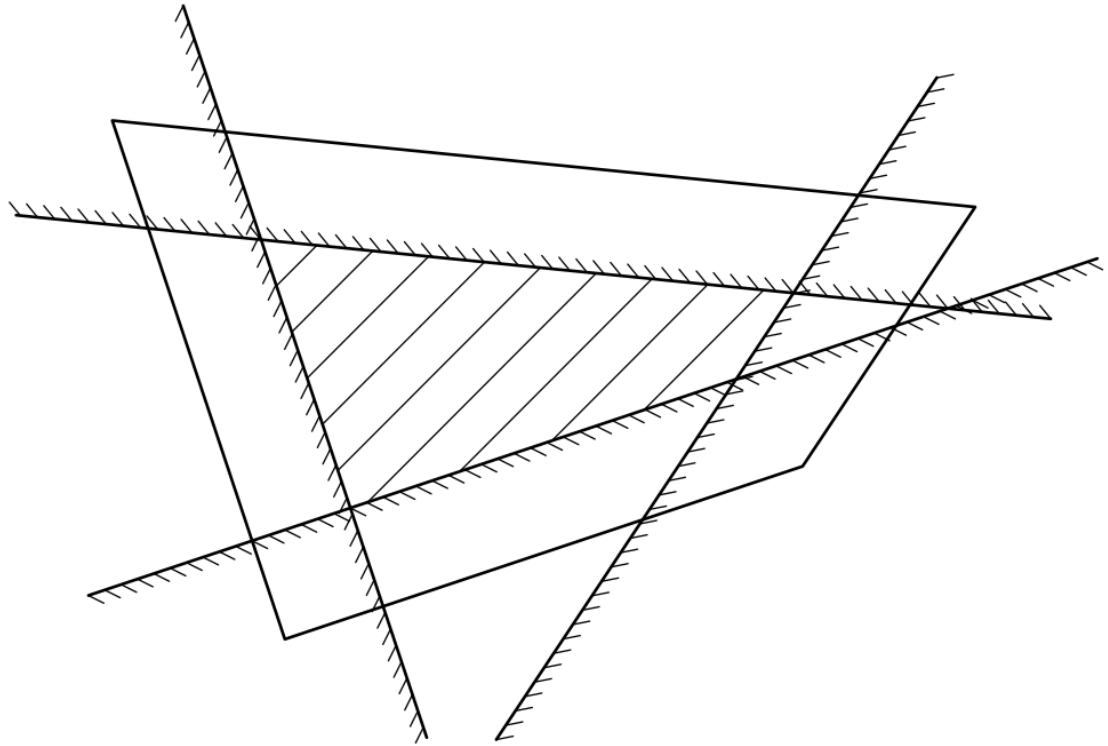


Решение задачи

- Двоичный поиск по ответу
- Нужно проверять, можно ли сделать трассу для радиуса r
- Будем располагать трассу вдоль внешнего барьера



- Множество точек, которые лежат внутри внешнего барьера, и никогда не оказываются внутри круга, образуют выпуклый многоугольник
- А конкретнее, многоугольник, получающийся смещением каждой стороны внешнего барьера на $2r$, перпендикулярно стороне, внутрь многоугольника



- Нужно проверить, можно ли разместить внутренний барьер внутри получившегося многоугольника
- Научимся проверять для выпуклых многоугольников A и B, можно ли параллельным переносом сместить B, чтобы он оказался внутри A
- B лежит внутри A, если каждая вершина B лежит внутри A
- $V_1 = \{v : b_1 + v \in A\}$, где b_1 — первая вершина B
- Заметим, что $V_1 = A - b_1$
- V_1 получается из A параллельным переносом на вектор $-b_1$
- Аналогично определим $V_2, \ldots V_m$
- $V = \bigcap V_i$

- Чтобы проверить на пустоту множество V, нужно проверить на пустоту пересечение *nm* полуплоскостей
- Поддерживаем выпуклый многоугольник, являющийся ответом.
- Начинаем, например, с большого прямоугольника.
- Пересекаем его с полуплоскостями.

Задача L
Сколько тестов

Автор задачи:
Андрей Станкевич
Разработка задачи:
Андрей Станкевич

Задача

Alias	L
Задача	Сколько тестов
Идентификатор	school.spb.2018.main.unerase-tests

Модель оценивания: icpc

Скрипт тестирования: null

Лимит времени	PT1S
Лимит памяти	536MB
Лимит вывода	
Проверяющая программа	http://neerc.ifmo.ru/develop/

№	Ввод	Ответ	Комментарий
01	tests/01	tests/01.a	
02	tests/02	tests/02.a	
03	tests/03	tests/03.a	
04	tests/04	tests/04.a	
05	tests/05	tests/05.a	
06	tests/06	tests/06.a	
07	tests/07	tests/07.a	
08	tests/08	tests/08.a	
09	tests/09	tests/09.a	
10	tests/10	tests/10.a	
11	tests/11	tests/11.a	
12	tests/12	tests/12.a	
13	tests/13	tests/13.a	
14	tests/14	tests/14.a	
15	tests/15	tests/15.a	

Постановка задачи

- Тесты к задаче пронумерованы от 1 до n и дополнены ведущими нулями до минимально возможной одинаковой длины
- Даны некоторые номера тестов
- Определить минимальное и максимальное возможное общее количество тестов

Решение задачи

- Пусть все строки имеют длину l
- Максимальное возможное количество тестов – $10^l - 1$
- Посмотрим на максимальный номер теста среди данных
  - Если в нем есть ведущие нули, то минимальное возможное количество тестов – 10^{l-1}
  - Если ведущих нулей нет, то минимальное возможное количество тестов совпадает с максимальным номером теста среди данных

```
printf("%s\n", "Спасибо за внимание")
```

Материалы олимпиады

http://neerc.ifmo.ru/school