2. ВТОРОЙ ЭТАП

Задачи второго этапа

Данный этап включает задачи, для решения которых достаточно школьных знаний и умений программы 10-11 класса, навыков использовать школьные знания для решения новых задач. Включает 5 задач, целью которых является подготовка к финальному командному туру. Данный тур позволяет очертить область предметных знаний необходимую для участия в профиле. Методические рекомендации к данному туру позволяют очертить область знаний и навыков для самостоятельного изучения. Все задачи с уклоном в специфику тура, сочетают в себе математику и информатику. Задачи участники решали на сервере компании разработчиков через веб-интерфейс.

3.1. Задание 1

Задача 3.1.1. Чет-нечет (6 баллов)

Тренируемые навыки: работа с массивами, поиск периодичностей, поиск максимумов, элементы теории вероятности, код Хемминга, логарифмы.

Условие задачи

Начинающий связист написал программу, реализующую алгоритм помехоустойчивого кодирования на основе кода Хемминга, например (7,4) и решил использовать его для любого канала связи, предварительно не исследуя присутствующие в нем шумы и помехи. Однако через какое-то время выяснилось, что выбранный код не обеспечивает помехоустойчивую связь, т.е. в блоке длинной 7 бит появляются двукратные ошибки.

Задача состоит в том, чтобы по частоте возникновения ошибки в зависимости от номера блока определить, как часто встречаются ошибки и какой код Хемминга нужно использовать для полного устранения ошибок.

Необходимо выбрать такой код, который будет обеспечивать наименьшую избыточность и исправлять все ошибки.

Формат входных данных

Файл с распределение вероятности ошибки в зависимости от номера принятого блока, следующего формата: первая строка, содержит N – кол-во блоков, в следующих N строках p_i – вероятность встретить ошибку в блоке i.

Формат выходных данных

Файл, содержащий код Хемминга, который позволяет избежать ошибок.

Пример №1

Стандартный ввод
11 0 0.12345679 0.148148148 0.074074074 0 0.074074074 0.148148148 0.12345679 0 0 0.12345679
Стандартный вывод
(9, 5)

Система оценки

Программа по входному файлу создает выходной файл. Если программа получает верный результат и выполняется не более 1 минуты, команда получает максимальное количество баллов — 6.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018;06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи.

Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками.

Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Комментарии

Вопрос:

Под частотами ошибок, которые подают на ввод, имеются ввиду частоты появления в принципе ошибки хотя бы в одном бите или частоты появления именно двукратной ошибки?

Ответ:

Речь идет о частоте ошибок на выходе декодера, работающего на основе кода Хемминга (7,4). Сбой происходит, если на вход декодера приходит блок с двумя и большим количеством ошибок.

Алгоритм решения задачи

Причина появления ошибок в случае, когда частота ошибок реже, чем ожидается в канале, представлено на рисунке 1. Видно, что если однократная ошибка появляется в 9-битовом блоке, то двукратная ошибка в 7-битном блоке появляется в случае, когда 7-битный блок оказывается на стыке двух 9 битовых блоков. На рисунке также показано как рассчитывается вероятность двукратной ошибки в зависимости от номера 7-битного блока.

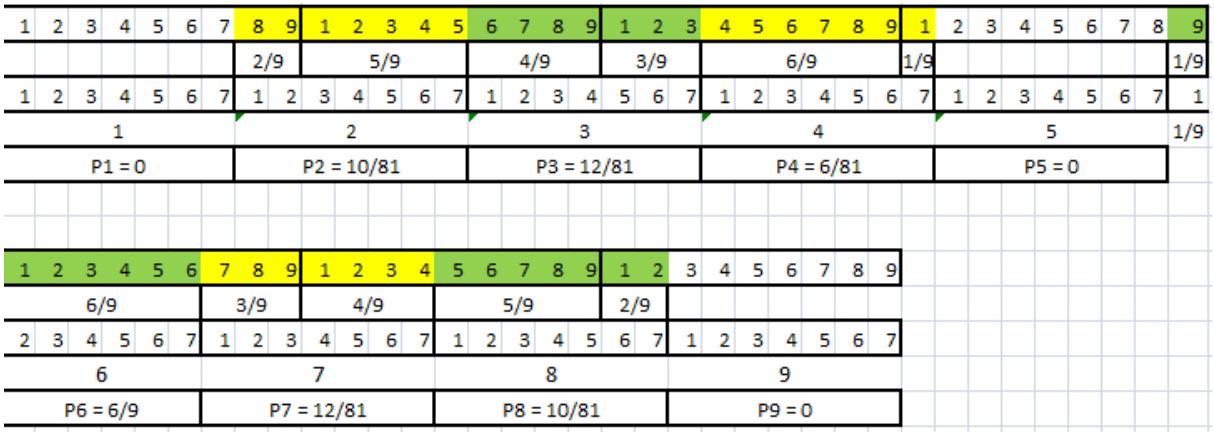


Рисунок 1. Причина появления 2-х кратной ошибки.

На рисунке ниже представлен график вероятности, появления двукратной ошибки в зависимости от номера 7-битного блока. На графике виден характерный период, соответствующий частоте появления ошибки, в данном случае видно, что однократная ошибка встречается в среднем каждые 9 бит. Таким образом, для решения задачи необходимо определить период в представленной зависимости вероятности появления двукратной ошибки в зависимости от номера 7-битного блока.

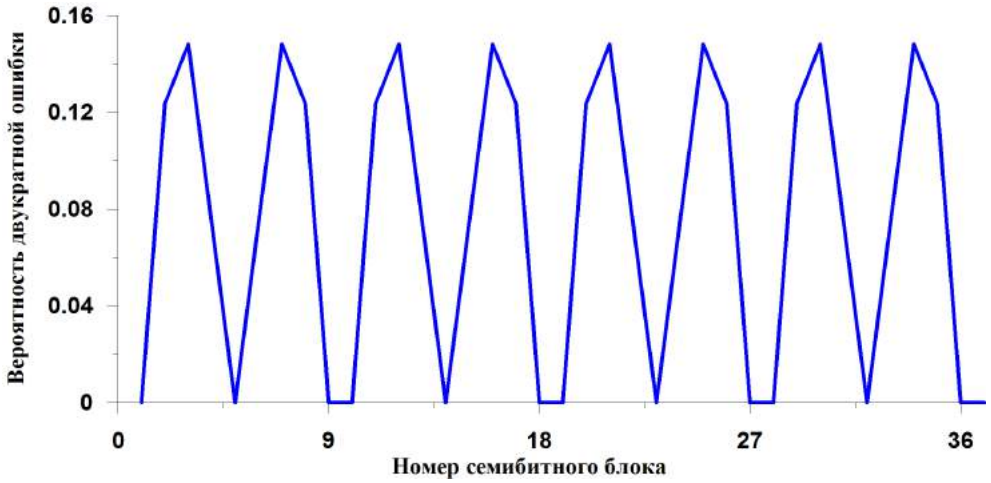


Рисунок 2. Вероятность двукратной ошибки.

Для поиска периодичности в функции можно воспользоваться автокорреляционным подходом. В данном подходе определяется энергия $f(i)$ произведения исходной функции $p(j)$ и сдвинутой $p(i+j)$ на некоторый интервал i.

$$f(i) = \sum_{j=1}^{N} p(j) \times p(i+j).$$

При сдвиге i кратном периоду функции, сдвинутая функция совпадет с исходной и энергия, при таких сдвигах, даст максимальные значения. На рисунке 3 представлена автокорреляционная функция $f(i)$ рассчитанная для функции, приведенной на рисунке 2.

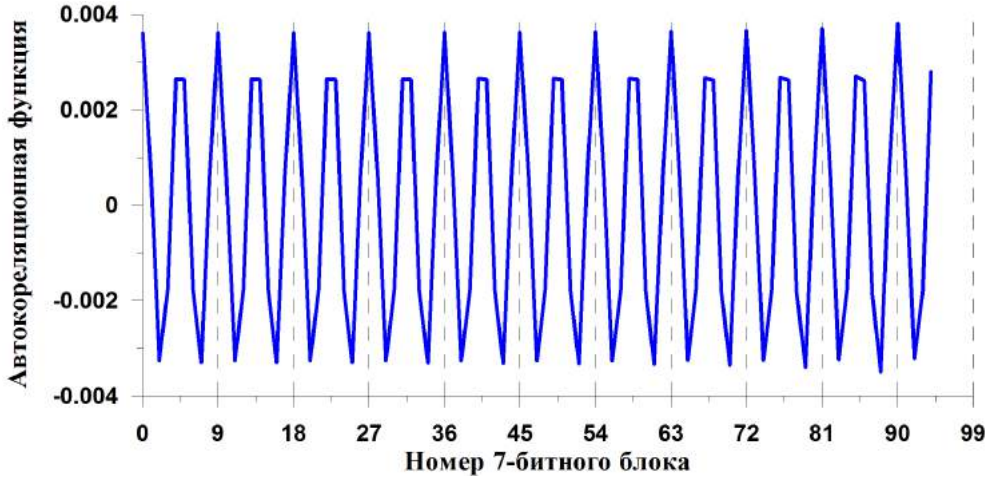


Рисунок 3. Вид автокорреляционной функции для функции, представленной на рисунке 2.

Далее, вычисляя расстояния между главными максимумами, полученной автокорреляционной функции $f(i)$ находим период функции m.

Осталось определить какой код Хэмминга использовать для того чтобы общая длина блока была равна $m = n + k$, где n – количество информационных бит, k – количество контрольных бит. Для этого нужно решить неравенство:

$$2^k \geqslant k + n + 1 = m + 1,$$

тогда

$$k = [log_2(m)] + 1.$$

В выходной фал записываем **(m,m-k)**.

3.2. Задание 2

Задача 3.2.1. Веселый спутник (26 баллов)

Тренируемые навыки: преобразование различных типов координат, работа с массивами, фильтрация случайных выбросов, поиск максимума коэффициента корреляции последовательностей, методы поиска оптимальных значений.

Условие задачи

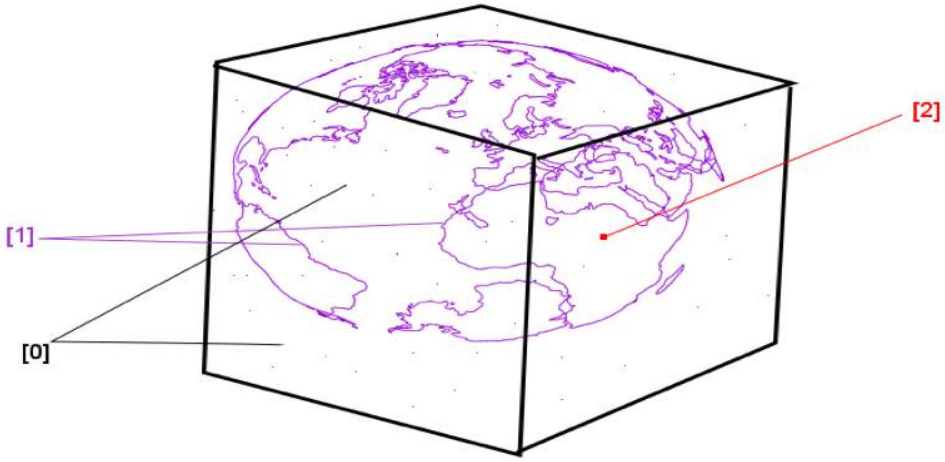
Спутник занимается мониторингом землетрясений на удаленной планете. Программное обеспечение спутника позволяет определять границы континентов и местоположение эпицентра землетрясения. Зная его ориентацию и орбиту программы

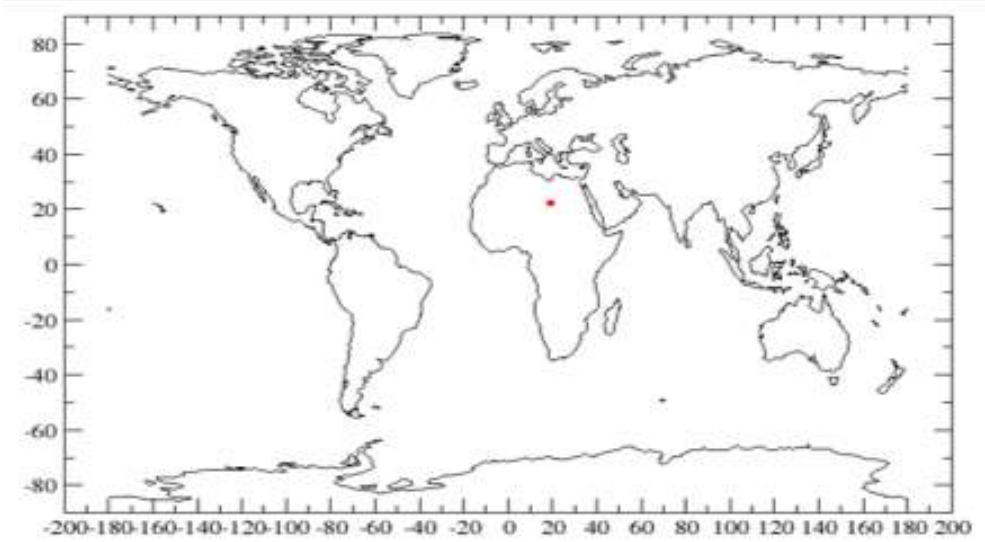
спутника позволяют определять географические координаты землетрясения и передавать их на Землю.

В процессе функционирования блок, отвечающий за ориентацию спутника вышел из строя, и спутник начал вращаться по очень сложной орбите, которую нельзя скорректировать и блок расчета координат землетрясения перестал функционировать. Съемка границ континентов и привязка к ним эпицентра землетрясения продолжает работать и это информация предаётся на Землю в виде трехмерного образа планеты (а именно в виде последовательности нулей, единиц и двоек). Единицам соответствуют границы континентов, двойкам - местоположение эпицентра землетрясения, нулям - все остальные точки. Контуры континентов расположены на поверхности воображаемой сферы с неизвестным радиусом, центром и ориентацией (они стали неизвестны из-за сбоя орбиты спутника). Кроме того, в данных может присутствовать случайный шум из единиц, возникший из-за передачи большого объема данных.

Задача: Сделать программу, которая по полученному трехмерному образу планеты (заданному файлом в виде последовательности 0, 1 и 2) и двумерной карте материков (заданному файлом в виде последовательности координат каждой границы - широта и долгота) определяет географические координаты землетрясения с точностью не хуже удвоенного размера дискрета в первом файле.

Пояснения





Красная точка [2] на верхнем рисунке – эпицентр землетрясения, который надо определить.

Синие точки – границы континентов и случайный шум, остальные точки – нули.

Снизу – карта неизвестной планеты.

Описание формата входных и выходных данных для задачи:

Порядок следования данных: $x[0][0][0]$, $x[0][0][1]$, $x[0][0][2]$, ..., $x[0][0][N]$, $x[0][1][0]$, $x[0][1][1]$, $x[0][1][2]$, ...$x[0][N][N]$, $x[1][0][0]$, $x[1][0][1]$, ..., $x[N][N][N]$. Трехмерный массив $201 \times 201 \times 201$.

Соответствующие им результаты работы программы приведены в файлах out1.txt, out2.txt и out3.txt

Оцифрованная карта материков приведена в файле world_110m.txt:

долгота1 широта1 долгота2 широта 2 ... долготаN широта N

В файле world_110m.txt могут встречаться пустые строки

Языки программирования - Python,C,C++,Java

Программа должна читать исходный файл со стандартного потока stdin и передавать решение на стандартный поток stdout

Требования к программе:

Программа должна читать исходные данные со стандартного потока ввода (stdin) и выдавать его на стандартный поток вывода (stdout).

В процессе работы она может читать файл двумерной карты материков, имеющий название world_110m.txt

Количество начисляемых баллов:

Для проверки решения в различных начальных условиях используется несколько тестовых примеров (в этой задаче - 10 тестовых примеров). По каждому примеру определялось количество баллов, заработанное командой в этом примере. Результирующая оценка команды за задачу вычисляется, как среднее значение по проведенным тестам:

$$\text{Количество баллов за задачу} = \frac{1}{10}\sum_{i=1}^{10} Q_i$$

Баллы за каждый тест рассчитываются следующим образом.

Если программа успешно выполнила тест за 5 минут, то команда получает за этот тест половину максимального количества баллов $Q_i = 13$.

Если программа успешно выполнила тест за 2.5 или менее минут, команда получает за этот тест максимальное количество баллов $Q_i = 26$.

Если программа успешно выполнила тест за время T от 2.5 до 5 минут команда получает за этот тест

$$Q_i = (1 - (T - 2.5)/5) \cdot 26 \text{ баллов}$$

Если программа выполняет тест более 5 минут или выполнила его неуспешно, команда получает за этот тест 0 баллов.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018; 06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи.

Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками. Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Алгоритм решения задачи

1. Удалить шум. Поскольку он имеет вид случайных выбросов, можно удалить все точки со значением равным 1, у которых нет соседей со значением равным 1 в трех измерениях (т.е. в ближайших 26 ячейках). Все остальные точки будут находиться на поверхности некой сферы или не дальше одного дискрета сетки от нее. Удаление шума можно сделать, например предварительно занеся исходные данные в трехмерный массив координат (x, y, z) и проверив число соседей со значением равным 1 у каждой клетки.
2. Определить центр и радиус сферы. Один из способов - поиск оптимального значения функции невязки прямым перебором координат.

   Условием оптимального значения является такое положение центра (x_c, y_c, z_c) при котором минимален разброс расстояния между центром и точками, со значением, равным 1 в смысле:

   $$\sum_{i=1}^{N}(r_i - R)^2 = min \qquad (1)$$

   где

   $$r_i = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2 + (z_c - z_i)^2} \qquad (2)$$

   - расстояние от центра (x_c, y_c, z_c) с координатами до i-ой точки со значением 1 с координатами (x_i, y_i, z_i)

   $$R = \frac{1}{N}\sum_{i=1}^{N} r_i \qquad (3)$$

   - среднее расстояние от центра до точек со значением 1.

   при этом R будет радиусом этой сферы, а (x_c, y_c, z_c), на которых этот минимум (1) достигается, являются координатами центра.

   На первом этапе можно перебирать значения (x_c, y_c, z_c) с шагом в 1 градус в пределах $-50 \leqslant x_c \leqslant 50; -50 \leqslant y_c \leqslant 50; -50 \leqslant z_c \leqslant 50$, на втором этапе уточнить найденное значение перебором с шагом в 0.2 градуса вблизи найденного на первом этапе значения.
3. Перевести получившиеся точки со значениями 1 или 2 в сферические координаты (широта,долгота) на эквидистантную (прямоугольную) сетку, с шагом в 1 градус по широте и долготе.

4. Перевести карту планеты, заданную в файле world_110m.txt в сферические координаты на идентичную пункту 4 сетку.
5. Одним из методов поиска, например перебором с шагом в 1 градус найти сдвиг сферических координат (широта, долгота) при котором положения точек со значением 1 на сетках, полученных на этапах 4 и 5, дают наибольшее количество совпадений.
6. Вычислить координаты эпицентра (точки со значением 2) исходя из найденного оптимального положения.

3.3. Задание 3

Задача 3.3.1. Космический вальс (30 баллов)

Тренируемые навыки: работа с массивами, рекурсивные алгоритмы перебора, школьный курс геометрии: расстояние между точками, параллельный перенос, зеркальное отражение, пересечение прямых, элементарная тригонометрия, работа с массивами и структурами данных.

Условие задачи

Контрабандисты на корабле A неожиданно подлетели к неподвижному патрульному кораблю C по неизвестной сложной траектории. Патруль корабля C попросил команду корабля A предоставить запись маршрута движения с целью проверки посещения запретных зон. Однако, на корабле A система записи маршрута движения не работала, и команда утверждала, что в запретные зоны они не заходили. Корабли могут двигаться только между узлами пятиугольной плитки, параметры которой и способ укладки приведены на рисунках 1 и 2. Командир C обратил внимание на то, что команда A, как и команда C слушает и записывает музыку радиостанции B. Сравнив мелодии, записанные на A и C, командиру корабля C удалось восстановить траекторию движения корабля A и проверить контрабандистов.

Вам предлагается решить эту же задачу — определить траекторию движения корабля , сравнив записи мелодий на кораблях и C. Радиостанция находится в начале системы координат, т.е. $x_B = y_B = 0.0$. Данные передаются и принимаются пакетами длительностью в ♪= 1/8. Каждый пакет представляют собой отдельную строку в текстовом файле, хранящем запись мелодии. Каждый пакет передается и принимается, во время движения корабля от вершины к вершине. Пример мелодии и соответствующий ей файл приведены ниже в приложении А.

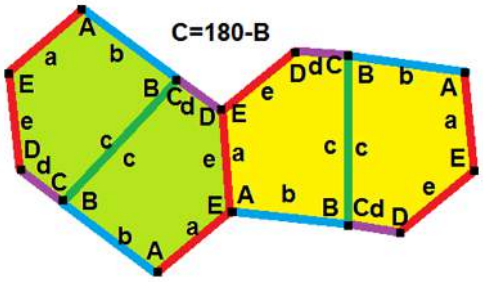


Рисунок 1. $a = e, B + C = 180°, A + D + E = 360°$

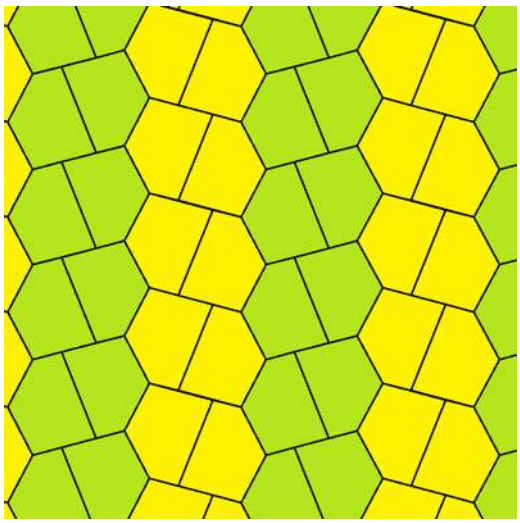


Рисунок 2. Способ замощения пространства.

Постоянные параметры задачи:

- Скорость распространения сигнала = 100000.0 (сигнал распространяется по прямой, не по вершинам);
- Скорость корабля $V = 250.0$;
- Несущая частота $f_0 = 40000.0$.

Формат входных данных

Input.dat

- Координата корабля $\{x_A, y_A\}$,
- координата корабля $\{x_C, y_C\}$,
- координаты вершин трех плиток, окружающих точку ,

Real_song.dat — переданный файл с корабля B, Record_song.dat — Принятый файл на корабле A.

Формат выходных данных

Output.dat — файл с маршрутом движения (последовательность координат узлов, посещенные кораблe A до встречи с патрульным кораблем B).

Формат Output.dat:

КоординатаX КоординатаY Доплеровское смещение частоты

Система оценки

Задача считается решенной, если суммарное отклонение найденной траектории от действительной траектории A не превышает 10% от общей длины маршрута.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018;06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи.

Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками.

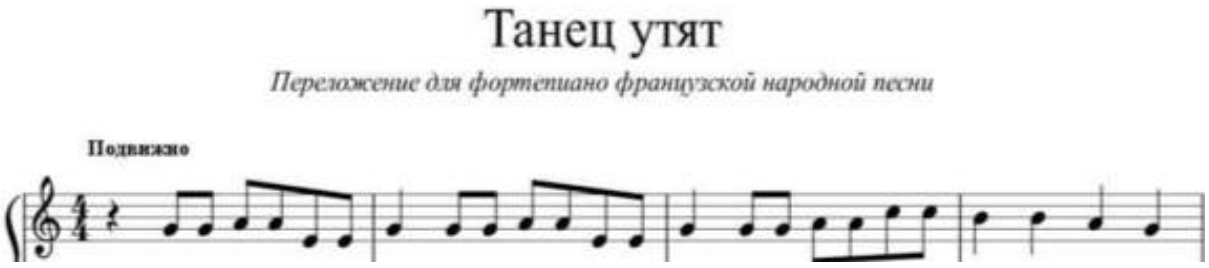
Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Приложение А

Таблица нот и соответствующих им частот в Гц.

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
Малая октава	130.81	138.59	146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94
Первая октава	261.63	277.18	293.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88
Вторая октава	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	880.00	932.33	987.77

Мелодию будем представлять в текстовом формате, следующего вида. Например, на нужно закодировать первую строку в мелодии «Танец утят»:



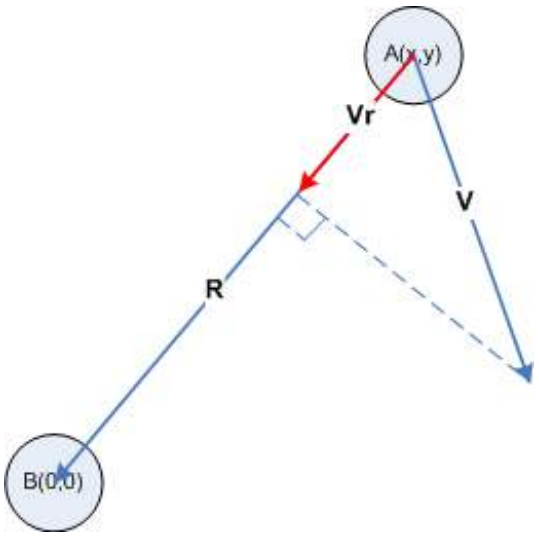
В строках будем отмечать ноты длительностью 1/8, если нота прерывается то ставим 2, если нота продолжает звучать ставим 1, например:

Малая октава												Первая октава												Вторая октава												
C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Первый такт
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Второй такт
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Третий такт
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	Четвертый такт
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Мелодии, записанные на кораблях A и C отличаются, так как корабль C остается неподвижным, а корабль A двигается. В этом случае на A наблюдается эффект Доплера – изменение частоты излучения, воспринимаемое наблюдателем (приёмником), вследствие движения источника излучения и/или движения наблюдателя (приёмника). Смещение частоты df можно найти по следующей формуле:

$$df = \frac{f_0 V_r}{C}.$$

Например, угол между радиус-вектором R и вектором скорости V составляет 60 град, тогда радиальная скорость $V_r = 250 \cdot cos(60) = 125$ и $df = 50$ Гц. Если в этот момент передавалась нота Ля первой октавы (частотой 440 Гц), тогда на корабле A ее услышали на частоте $440 + 50 = 490$ Гц, так как приемник записывает только «чистые» ноты, то была записана наиболее близкая к 490 Гц нота Си (первой октавы, частота 493.88 Гц.)



Для вычисления доплеровского смещения частоты необходимо найти радиальную скорость V_r – проекцию скорости движения V на радиус вектор R, направленный от A к B.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018; 06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи.

Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками.

Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Алгоритм решения задачи

1. Так как известны координаты вершин трех базовых плиток решетки вокруг начала координат, то построить решетку можно с помощью параллельных переносов и зеркальных отражений.
2. На полученной решетке построение требуемой траектории можно с помощью небольшой модификации стандартного переборного алгоритма (`http://acm.mipt.ru/twiki/bin/view/Algorithms/GenerateCPP`). Модификация заключается в том, что выход из рекурсии происходит на «запрещенных» вершинах, т.е. таких вершинах переход в которые запрещен доплеровским сдвигом частоты (формула приведена в условии).

3.4. Задание 4

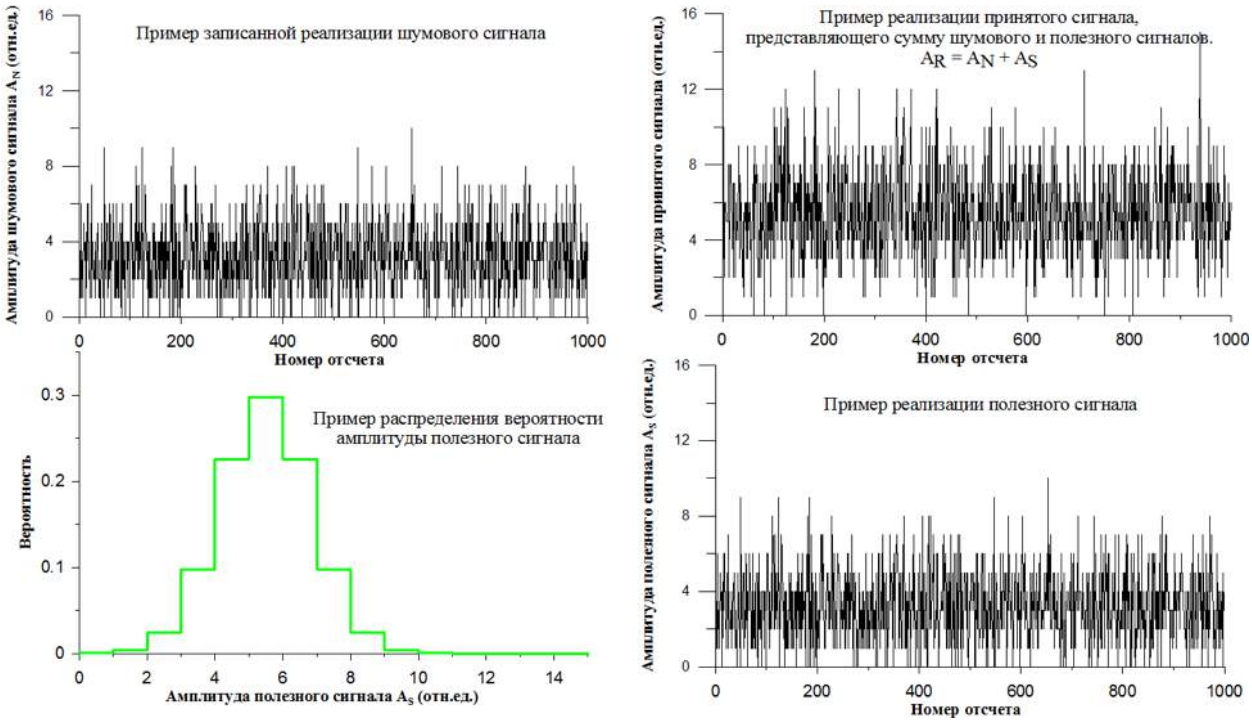
Задача 3.4.1. Передача информации по нескольким каналам связи (22 балла)

Тренируемые навыки: теория вероятности, условная вероятность, работа с массивами.

Условие задачи

Необходимо принять с некоторого удаленного объекта сообщение длиной M единиц. Для этого важно понять хватит ли для этого наших возможностей. В нашем распоряжении 2 канала связи в каждом присутствует свой стационарный шум. Для анализа свойств шума в каждом канале была записана реализация шумового сигнала в $N = 1000$ отсчетов. Также известно, что вероятность передачи данных (полезного сигнала) с удаленного объекта составляет 50%, т.е. 50% времени объект передает данные и 50% времени «молчит». Амплитуда полезного сигнала не постоянна, но для каждого канала известно распределение вероятности его амплитуды. В качестве иллюстрации на рисунке приведены: верхний левый – реализация шумового сигнала, нижний левый – распределение вероятности амплитуды полезного сигнала, нижний правый -– пример реализации сигнала, соответствующий распределению на левом нижнем рисунке, правый верхний — пример принятого сигнала A_R, представляющего сумму шума A_N и полезного сигнала A_S:

$$A_R = A_N + A_S.$$



Верхний левый рисунок — пример шумового сигнала, нижний левый рисунок — пример распределения вероятности амплитуды полезного сигнала, пример реализации сигнала, соответствующий распределению на левом нижнем рисунке, правый

верхний — пример принятого сигнала, представляющего сумму шума и полезного сигнала.

Из рисунка видно, что сигналы в каналах очень слабые, таким образом, рассматривается задача обнаружения сигнала (есть сигнал/нет сигнала). Решение о том, что в принятом сигнале присутствует полезный сигнал, принимается на основе сравнения амплитуды принятого сигнала с пороговым уровнем, если амплитуда больше или равна порогу, считаем, что полезный сигнал присутствует и отсутствует в противном случае. Пороговый уровень должен быть таким, чтобы сумма вероятности пропуска сигнала и вероятности ложных тревог (вероятность превышения шумом порога) была минимальной.

Если использование только одного из представленных каналов не позволяет решить задачу, тогда нужно задействовать несколько каналов и принимать решение о наличии сигнала голосованием – при условии, что, хотя бы в двух каналах сигнал выше порогового уровня, то принимаем решение — полезный сигнал присутствует. Для каждого канала известна потребляемая мощность. При использовании нескольких каналов суммируется и их энергопотребление.

Таким образом, необходимо проанализировать представленные шумовые сигналы и гистограммы распределения полезного сигнала, определить пороговый уровень и принять решение о том, какие каналы нужно задействовать, для приема сообщения длинной M=50 с наименьшими энергетическими затратами, так чтобы с вероятностью 99% в сообщении было не более 2 ошибок.

Языки программирования — Python, C, C++, Java

Примеры работы программы, решающей предлагаемую задачу присоединены.

Формат входных данных

2 — кол-во каналов;

50 — длина передаваемого сообщения;

2 — допустимое количество ошибок;

0.6 – вероятность передачи данных (полезного сигнала);

4.0296 5.0000 — 2 числа – затраты энергии на передачу сигнала в каждом канале;

15 — количество уровней сигнала;

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 — значения уровней сигнала;

Далее 2 строки – Распределения уровня сигнала в каждом каналах (вероятность встретить сигнал с данным уровнем в данном канале, строка 1 — распределение вероятности в 1 канале, 2-я строка — распределение во втором канале)

1000 – Длина шумовой реализации

Далее 2 строки – Реализация шумового сигнала для каждого канала (строка 1 – реализация, полученная в канале 1, строка 2 — реализация, полученная в канале 2).

Формат выходных данных

1 — необходимое кол-во каналов;

1 — номера используемых каналов;

4 — пороговый уровень для каждого из выбранных каналов;

0.996471 — вероятность верного обнаружения единичного сигнала, при использовании всех выбранных каналов;

0.999239 – Вероятность того что сообщение длины 50 будет получено с не более чем 2 ошибками;

5.000 – Необходимая энергия для передачи единицы сообщения (суммарная энергия всех задействованных каналов)

Система оценки

Максимальное число баллов — за полностью верный ответ, включающий:

- Правильно определены номера каналов, использование которых позволит решить задачу с наименьшими энергетическими затратами; (2 балла)
- Правильно определены пороговые уровни выбранных каналов, перечисленных в пункте 1; (6 баллов)
- Правильно определена вероятность верного обнаружения единичного сигнала, при использовании всех выбранных каналов, перечисленных в пункте 1 (данную вероятность необходимо определить с точность до 4-го знака); (10 баллов)
- Правильно определена вероятность того, что с использованием выбранных каналов (пункт 1), сообщение длины M будет передано с не более чем 2 ошибками (данную вероятность необходимо определить с точность до 4-го знака); (3 балла)
- Правильно рассчитана энергия, необходимая для передачи единицы сообщения (суммарная энергия всех задействованных каналов). (1 балл)

Штрафы:

- В пункте (3):
  - за ошибку в четвертом знаке снимается 3 балла.
  - за ошибку в третьем знаке снимается 10 баллов.
- В пункте (4):
  - за ошибку в четвертом знаке снимается 1 балл.
  - за ошибку в третьем знаке снимается 3 балла.

Скорость выполнения программы — не более 10 секунд

Программа должна читать исходный файл со стандартного потока *stdin* и передавать решение на стандартный поток *stdout*.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018; 06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи.

Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками.

Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Алгоритм решения задачи

1. Для каждого канала строим гистограмму распределения шума. Интеграл, полученной гистограммы нормируем на единицу. Полученное распределение обозначим $P_{Ni}(u)$, I – номер канала $i \ni [1, 2]$.
2. Для каждого канала на основе распределений сигнала и шума строим распределение "сигнал+шум" $P_{SNi}(u) = \int_0^u P_{Si}(v)P_{Ni}(u-v)dv$. В дальнейшем пороговый уровень будет определяться на основе $P_{Ni}(u)$ и $P_{SNi}(u)$.
3. Используем т. Байесса для того чтобы для каждого канала найти пороговый уровень u_{bi}, при котором минимизируется вероятность ложных тревог и вероятность пропуска.
$$q_i + n_i \to min,$$
$q_i = \int_0^{u_{bi}} P_{SNi}(u)du$ – вероятность пропуска сигнала,
$p_i = 1 - q_i$ – вероятность верного определения сигнала в канале при данном пороговом уровне u_{bi}.
$n_i = \int_{u_b i}^{\infty} P_{Ni}(u)du$ – вероятность ложной тревоги.
4. Находим вероятность правильного принятия решения о наличии сигнала, на основе, найденного порогового уровня.
Вероятность принятия правильного решения о наличии сигнала в канале: $P_i = \frac{p \cdot p_i}{q \cdot n_i + p \cdot p_i}$, где p – вероятность присутствия сигнала, $q = 1 - p$.
5. Для каждого канала рассчитываем вероятность принятия сообщения из M отсчетов с не более чем двумя ошибками:
$$P_{Mi} = P_i^M + C_M^1 P_i^{M-1}(1 - P_i) + C_M^2 P_i^{M-2}(1 - P_i)^2$$
6. Если для одного канала вероятность принятия сообщения из M отсчетов с не более чем двумя ошибками удовлетворяет требованиям задачи, тогда решение задачи – использование данного канала. В случае, если оба канала удовлетворяют требованиям задачи, выбирается канал с наименьшими затратами энергии.
7. Если ни один из каналов не удовлетворяет требованиям задачи, тогда возвращаем сообщение о том, что невозможно принять сообщение длинной M с не более чем двумя ошибками.

3.5. Задание 5

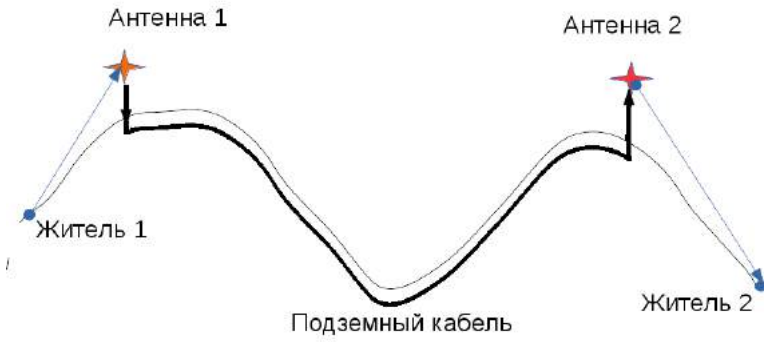
Задача 3.5.1. Сложный рельеф (16 баллов)

Тренируемые навыки: освоение работы с дискретными величинами, растровыми алгоритмами, расчет сторон и углов треугольника, работа с массивами, рациональная организация циклов и условий.

Условие задачи

Оператор сотовой связи принял решение обеспечить связью край Хоббитания. Хоббитания обладает холмистым рельефом, но не содержит крутых гор и скал. Оператор так планирует расположить вышки так, чтобы обеспечить связью всю территорию и при этом использовать наименьшее количество антенн – ретрансляторов.

Антенна устанавливается на высоте 1 от поверхности рельефа. Хобитания будет считаться обеспеченной связью, если любой ее житель находится в прямой видимости хотя бы одной антенны. Сами антенны будут связаны оператором кабелем, находящемся в земле. (см.рисунок). Желательно расположить антенны как можно ближе к друг другу (чтобы уменьшить затраты оператора на монтаж и обслуживание антенн и кабеля).



Задача состоит в том, чтобы написать программу, которая для любого рельефа (удовлетворяющего условиям задачи) определяет положения двух антенн (ретрансляторов), которые обеспечивают устойчивую связь между любыми двумя жителями Хоббитании.

Формат входных данных

Входной файл содержит форму рельефа Хобитании.

Первая строка заголовок.

Каждая последующая строка содержит запись, представляющую собой 3 числа, разделенных табуляцией: X(координату),Y(координату) и Z(высоту) точки рельефа.

Всего записей в файле $101 \cdot 101$.

Формат выходных данных

Первая строка – координаты первой антенны (X1, Y1);

Вторая строка – координаты второй антенны (X2, Y2);

Программа должна читать входной файл со стандартного потокв ввода, и выдавать выходной файл на стандратный поток вывода.

Система оценки

Если задача решена успешно, и антенны обеспечивают устойчивую связь в Хобитании, то баллы определяется по формуле:

$$\text{Баллы} = 16 - 8 \cdot (R - R_{min})/(R_{max} - R_{min})$$

R_{max} — максимально возможное расстояние между антеннами;

R_{min} — минимально возможное расстояние между антеннами;

R — расстояние между антеннами, полученное вашей программой;

Таким образом, максимальное количество баллов заработает тот, кто поставит антенны, обеспечивающие связь в Хобитании, ближе друг к другу;

Программа должна работать не более 10 минут и использовать не более 100МБ памяти.

Комментарии

Вопрос:

Для определения видимости вышки с произвольных точек рельефа нужно, чтобы рельеф был представлен непрерывной поверхностью. Но рельеф в задаче представлен набором дискретных. Как из них выразить непрерывную поверхность?

Ответ:

Рельеф представляет собой набор вокселей с единичной стороной (растр в трёхмерном пространстве или сплайн 0-й степени).

Вопрос:

Как считается расстояние при начисления баллов: не учитывая z точек или учитывая ее?

Ответ:

Расстояние рассчитывается с учетом z координат точек.

Проверка проводилась в несколько этапов(попыток), проводимых в заданную дату (20.12.2018; 29.12.2018; 06.01.2019; 11.01.2019; 14.01.2019). В каждой попытке участники могли загрузить и проверить одно свое решение этой задачи. Результирующая оценка за задачу выбиралась, как максимальная из попыток.

Загрузка решений проводилась на сайте `http://dep1.iszf.irk.ru/wlcomm` на котором было установлено специальное программное обеспечение для регистрации, разделения доступа и возможности загрузки решений участниками.

Тестирование решений проводилось на виртуальной машине с установленной ОС Kubuntu, проверку и начисление баллов за каждый тест осуществляли автоматические программы.

Алгоритм решения задачи

Для решения задачи нужно найти две ближайшие точки P_1 и P_2, из которых остальные точки Хобитании будут находиться в прямой видимости.

1. **Проверки видимости точки.** Точка p_{ij} видна из точки P_k если ее угол места $\theta_i j$ больше угла места θ любой другой точки на прямой соединяющей точки P_k и p_{ij}, см. рисунок 1 и 2. Угол места может быть рассчитан так:

$$sin(\theta_{ij}) = \frac{z_{ij} - z_k}{((x_{ij} - x_k)^2 + (y_{ij} - y_k)^2))}$$

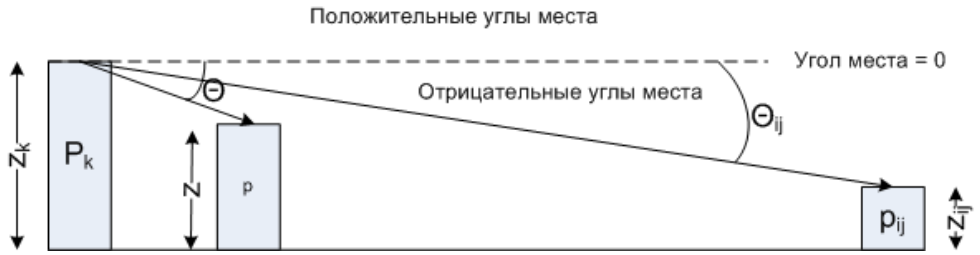


Рисунок 1. Вариант с отрицательыми углами места

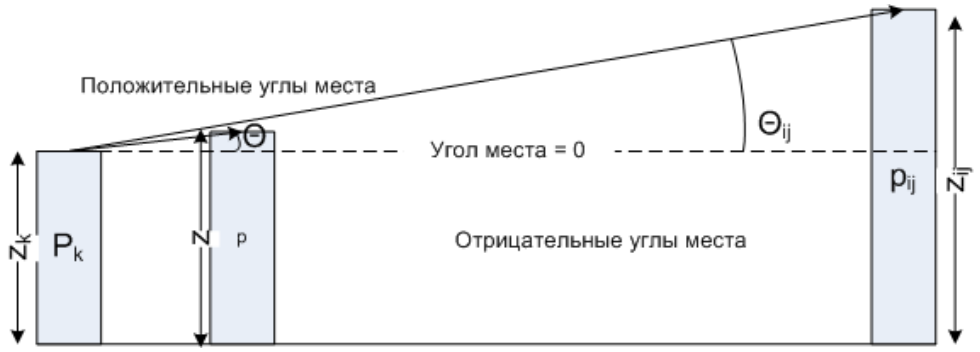


Рисунок 2. Вариант с положительными углами места.

2. **Построение прямой на растре.** Так как рельеф задан дискретно на матрице, для определения точек лежащих на прямой можно воспользоваться, например, растровым алгоритмом Брезенхема. Алгоритм основан на формуле между двумя точками. Для каждого x в заданном диапазоне, значение y получается округлением к целому следующего выражения:

$$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$$

Рассмотрим вариант расположения точек, представленный на рисунке 3. На каждом шаге приращения x на 1 y изменяется на $\frac{y_1 - y_0}{x_1 - x_0}$, а так как $|x_1 - x_0| > |y_1 - y_0|$, то y сохраняет свое значение, либо уменьшается на 1. Данный алгоритм можно распространить на вычисление координат клетки, принадлежащей прямой, во всех направлениях. Это достигается за счет: зеркальных отражений, то есть заменой знака (шаг в 1 заменяется на -1), обменом переменных x и y, обменом координат начала отрезка с координатами конца.

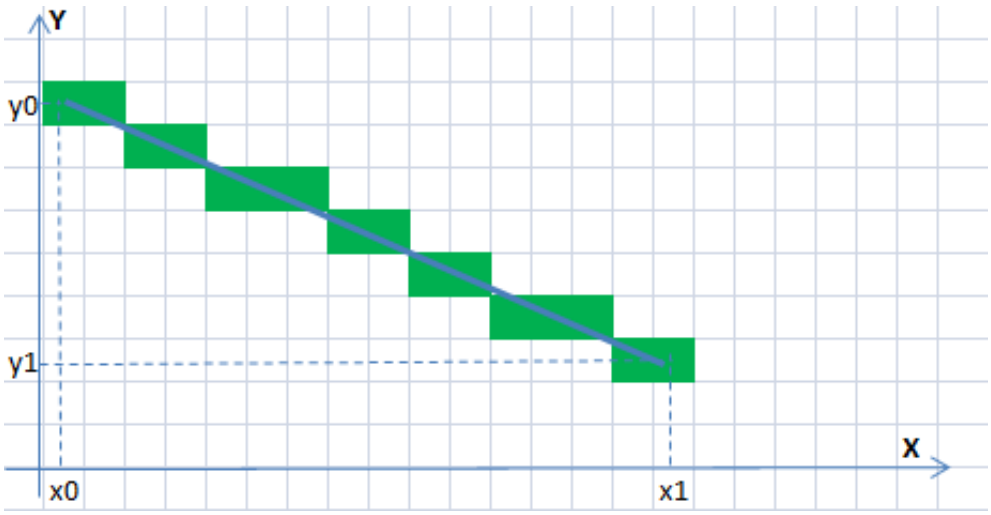


Рисунок 3. Растровый алгоритм построения прямой.

3. **Поиск оптимального расположения антенн.**
   - 3.1 Первую антенну располагаем последовательно во всех точках.
   - 3.2 Для выбранного положения первой антенны, определяем массив невидимых относительно нее точек.
   - 3.3 Располагаем вторую антенну во всех точках, в которой еще не было первой антенны.
   - 3.4 Для каждого выбранного положения второй антенны проверяем массив точек невидимых относительно первой антенны. Если в массиве встречается точка, которая невидна также из текущего положения второй антенны, то выходим из цикла перебора положения второй антенны и ставим первую антенну на новую позицию (переходим к 3.1).
   - 3.5 Если в 3.4 все невидимые точки относительно антенны 1 просматриваются из антенны 2, тогда рассчитываем расстояние между ними. Если найденное расстояние меньше найденного ранее, тогда запоминаем новые положения антенн и расстояние между ними, после переходим к пункту 3.1.