Санкт-Петербургский государственный университет

Отчёт о выполнении pet-проекта

«Решение нечётных олимпиадных задач из темы «Поиск в ширину» с сайта acmp.ru.»

Выполнил:
Студент группы 23.Б12-мм
Швец Владимир Владимирович

Санкт-Петербург
2023

Глава 1. Используемые библиотеки.

1.1 <iostream>

Данная библиотека использовалась в ходе написания кода для его отладки.

1.2 <fstream>

Данная библиотека использовалась для реализации файлового ввода-вывода, требуемого в условиях задач.

1.3 <string>

Данная библиотека использовалась в некоторых задачах, где было нужно работать со строками (например, построчный ввод).

1.4 <queue>

Библиотека для работы с классом std::queue<T> - очередь. Это контейнер, который работает по принципу FIFO (first-in first-out или "первый вошел — первым вышел") — первым всегда извлекается первый добавленный элемент. Класс, необходимый для реализации поиска в ширину.

1.5 <map>

Данная библиотека использовалась единожды для решения задачи. Представляет собой контейнер, где каждое значение ассоциировано с определенным ключом. И по этому ключу можно получить элемент. Причем ключи могут иметь только уникальные значения.

Глава 2. Ход решения задач.

2.1 A. Путь.

Путь

(Время: 1 сек. Память: 16 Мб Сложность: 40%)

В неориентированном графе требуется найти длину кратчайшего пути между двумя вершинами.

Входные данные

Во входном файле INPUT.TXT записано сначала число N - количество вершин в графе ($1 \leq N \leq 100$). Затем записана матрица смежности (0 обозначает отсутствие ребра, 1 - наличие ребра). Затем записаны номера двух вершин - начальной и конечной.

Выходные данные

В выходной файл OUTPUT.TXT выведите длину кратчайшего пути. Если пути не существует, выведите одно число -1.

Для решения этой задачи за основу взят код из домашнего задания по графам, где на ввод также поступала матрица смежности неориентированного графа.

Реализован файловый ввод этой матрицы и в класс добавлена новая функция, которая по алгоритму Флойда – Уоршелла преобразовывала матрицу смежности в матрицу кратчайших путей. Из матрицы кратчайших путей брался и выводился ответ.

2.2 C. Табличка

Табличка

(Время: 0,5 сек. Память: 16 Мб Сложность: 51%)

Вам дана табличка, состоящая из N строк и M столбцов. В каждой клетке таблицы стоит либо 0, либо 1. Расстоянием между клетками (x1,y1) и (x2,y2) называется |x1-x2|+|y1-y2|. Вам нужно построить другую таблицу, в которой в каждой клетке стоит расстояние от данной до ближайшей клетки, содержащей 1 (в начальной таблице). Гарантируется, что хотя бы одна 1 в таблице есть.

Входные данные

В первой строке входного файла INPUT.TXT содержатся два натуральных числа, не превосходящих 100 - N и M. Далее идут N строк по M чисел - элементы таблицы.

Выходные данные

Выходной файл OUTPUT.TXT должен содержать N строк по M чисел - элементы искомой таблицы.

Для решения этой задачи, нам необходимо сначала посчитать количество единиц в исходной таблице и запомнить их координаты. За это отвечают функции *OneCount()* и *edinici()* соответственно. Далее для каждой клетки таблицы мы находим расстояние до каждой единицы и записываем минимальное (*ansfunc()*). Полученный результат записываем в output.txt.

2.3 E. Звезда

Звезда

(Время: 1 сек. Память: 16 Мб Сложность: 50%)

Вам в руки попала карта звездного неба. Карта представляет собой прямоугольную таблицу, из N строк и M столбцов. В каждой ячейке таблицы может быть:

- Точка - означает отсутствие звезды.
- Знак "звездочка" - означает звезду.

Созвездие - это соединенные вместе несколько звезд. Звезды считаются соединенными, если они являются соседями сверху, снизу, справа или слева. Одна изолированная звезда также считается созвездием.

Напишите программу для подсчета количества созвездий на карте.

Входные данные

Первая строка входного файла INPUT.TXT содержит два натуральных числа N и M – количество строк и столбцов на карте звездного неба (N, M ≤ 300). Далее следуют N строк, каждая из которых содержит M символов «.» (точка) или «*» (звездочка).

Выходные данные

В выходной файл OUTPUT.TXT выведите количество созвездий на карте.

Изначально я пытался решить эту задачу поиском в ширину, «маркируя» пройденные вершины просто удаляя их. Таким образом мой алгоритм заключался в следующем:

1) Обойти массив
2) Если мы наткнулись на *, то поиском в ширину мы удаляем её и все соседние звёздочки и добавляем 1 к количеству созвездий.
3) Выводим количество созвездий.

Однако мой код на 8 тесте выдаёт memory limit exceeded. Тогда поискав информацию в интернете, я наткнулся на решение очень похожей задачи, но поиском в глубину. Тогда в своём алгоритме я решил удалять созвездия поиском в ширину. Такое решение прошло все тесты (Task-E_test).

2.4 G. Два коня

два коня

(Время: 1 сек. Память: 16 Мб Сложность: 55%)

На стандартной шахматной доске (8x8) живут 2 шахматных коня: красный и зеленый. Обычно они беззаботно скачут по просторам доски, пощипывая шахматную травку, но сегодня особенный день: у зеленого коня день рождения. зеленый конь решил отпраздновать это событие вместе с красным. Но для осуществления этого прекрасного плана им нужно оказаться на одной клетке. Заметим, что красный и зеленый шахматные кони сильно отличаются от черного с белым: они ходят не по очереди, а одновременно, и если оказываются на одной клетке, никто никого не съедает. Сколько ходов им потребуется, чтобы насладиться праздником?

Входные данные

Во входном файле INPUT.TXT содержатся координаты коней, записанные по стандартным шахматным правилам (т.е. двумя символами - маленькая английская буква (от a до h) и цифра (от 1 до 8), задающие столбец и строку соответственно).

Выходные данные

Выходной файл OUTPUT.TXT должен содержать наименьшее необходимое количество ходов, либо -1, если кони не могут встретиться.

Для решения этой задачи мы создаём две доски (массива 8x8) и заполняем их каким-то очень большими, заведомо недостижимым, (или отрицательным) числом.

Клетки первой доски мы заполняем длинами кратчайшего пути первого коня до этих клеток через поиск в ширину.

Клетки второй доски мы аналогично начинаем заполнять длинами кратчайшего пути второго коня до этих клеток. Однако если у нас в какой-то клетке длина пути на первой доске и на второй совпадут, мы выводим эту длину.

2.5 I. Алхимия

Алхимия

(Время: 1 сек. Память: 16 Мб Сложность: 42%)

Алхимики средневековья владели знаниями о превращении различных химических веществ друг в друга. Это подтверждают и недавние исследования археологов.

В ходе археологических раскопок было обнаружено m глиняных табличек, каждая из которых была покрыта непонятными на первый взгляд символами. В результате расшифровки выяснилось, что каждая из табличек описывает одну алхимическую реакцию, которую умели проводить алхимики.

Результатом алхимической реакции является превращение одного вещества в другое. Задан набор алхимических реакций, описанных на найденных глиняных табличках, исходное вещество и требуемое вещество. Необходимо выяснить: возможно ли преобразовать исходное вещество в требуемое с помощью этого набора реакций, а в случае положительного ответа на этот вопрос — найти минимальное количество реакций, необходимое для осуществления такого преобразования.

Входные данные

Первая строка входного файла INPUT.TXT содержит целое число m ($0 \le m \le 1000$) – количество записей в книге. Каждая из последующих m строк описывает одну алхимическую реакцию и имеет формат вещество1 -> вещество2, где вещество1 – название исходного вещества, вещество2 – название продукта алхимической реакции. m+2-ая строка входного файла содержит название вещества, которое имеется исходно, m+3-ая – название вещества, которое требуется получить.

Во входном файле упоминается не более 100 различных веществ. Название каждого из веществ состоит из строчных и заглавных английских букв и имеет длину не более 20 символов. Строчные и заглавные буквы различаются.

Выходные данные

В выходной файл OUTPUT.TXT выведите минимальное количество алхимических реакций, которое требуется для получения требуемого вещества из исходного, или -1, если требуемое вещество невозможно получить.

Для решения этой задачи, я создал двумерный массив N x 2, где в первом столбце записаны названия исходных веществ, а во втором получаемые. Строки массива соответствуют реакциям.

Теперь мы начинаем поиск в ширину, изначально добавив в очередь данное исходное вещество. На каждом шаге сначала мы проверяем не получили ли мы конечное вещество. Если получили, завершаем поиск, выводим длину пути. Если нет, то мы ищем наше вещество в первом столбце массива реакций и добавляем соответствующее ему вещество из второго столбца в очередь, а наше вещество перезаписываем как «passed».

2.6 K. Игра Jammed

Игра Jammed

(Время: 1 сек. Память: 16 Мб Сложность: 58%)

Всем известна игра «Пятнашки», где надо выстроить изначально неупорядоченную последовательность чисел, перемещая фишки с нанесёнными числами от 1 до 15 в квадрате 4×4. На основе данной игры была разработана другая – поле в ней лишь 4×2 клетки, на поле 7 фишек, но на фишках изображены буквы английского алфавита и арабские цифры (на каждой фишке – один символ, но на разных фишках могут быть одинаковые символы). Цель игры прежняя – упорядочить в соответствии с образцом стартовую расстановку фишек за минимальное количество ходов.

Свободная клетка обозначается специальным символом «#» и используется для перемещения фишек по полю. Перемещать фишки на свободную клетку разрешается из соседних клеток, имеющих общую грань со свободной. Например, на рисунке более правый символ «0» можно переместить вниз на свободную клетку, тогда «0» будет в нижней клетке, а пустой станет верхняя клетка, либо в свободную клетку переместить букву «C» или цифру «2».

M	0	0	A
8	C	#	2

Входные данные

Входной файл INPUT.TXT содержит четыре строки: две первые строки содержат стартовую комбинацию символов, следующие две - образец. Каждая строка содержит 4 символа (английский алфавит и арабские цифры), пустая клетка обозначается символом «#» (решетка).

Выходные данные

В выходной файл OUTPUT.TXT выведите минимальное количество перемещений, необходимых для получения искомой комбинации. Если нужную комбинацию получить нельзя, выведите число -1.

Для решения этой задачи я написал класс Jammed. Элементы этого класса хранят в себе текущее положение доски. Также для этого класса я реализовал функции хода по кажому из четырёх направлений, функцию чтения с файла и операторы == и <. Далее я реализовал функцию поиска в глубину, которая принимает начальное и конечное состояние доски и выводит число операций необходимое совершить чтобы из начального состояния добраться до конечного. Для реализации этой функции мне понадобилось воспользоваться библиотекой <map> для хранения пути с Jammed.

2.7 M. Кладоискатель

Кладоискатель

(Время: 1 сек. Память: 16 Мб Сложность: 49%)

Кладоискателю Васе попалась карта древнего подземелья. Подземелье представляет собой лабиринт размера N×M. Каждая клетка лабиринта либо пуста и по ней можно пройти, либо содержит стену. Из клетки можно переходить только в смежную по стене клетку (так, у каждой клетки может быть не более 4 смежных клеток).

В одной из клеток находится клад, который и хочет достать Вася. В лабиринте есть K входов, из которых Вася может начать свой путь.

Требуется определить, с какого входа Васе нужно начать свой путь, чтобы пройденное расстояние до клада было наименьшим. Если таких входов несколько, нужно вывести вход с наименьшим номером.

Входные данные

Первая строка входного файла INPUT.TXT содержит 2 числа N и M, задающие размеры лабиринта ($1 \le N, M \le 100$, $N \times M \le 100$). Далее следует описание лабиринта: N строк по M символов в каждой. 0 означает, что клетка свободна; 1, что в клетке находится стена. Символ «*» обозначает клетку с сокровищем (такая клетка в лабиринте ровно одна).

В (N+2)-й строке находится число K ($1 \le K \le N \times M$) – количество входов в лабиринт. Далее в K строках содержатся координаты входов. Так, в i-й строке содержатся числа y_i и x_i, означающие, что i-й вход расположен в y_i-й строке и в x_i-м столбце ($1 \le y_i \le N$, $1 \le x_i \le M$). Гарантируется, что координаты входов попарно различны, и то, что все входы расположены в пустых клетках. Ни один из входов не находится в клетке с сокровищем.

Выходные данные

В выходной файл OUTPUT.TXT выведите одно число – искомый номер входа (нумерация начинается с 1). Если до сокровища невозможно добраться, выведите -1.

Для решения этой задачи мы сначала считаем массив – «карту подземелья», и немного преобразуем эту карту, возведя стены из единичек по краям (readMatrix()). Далее мы считаем в массив входов координаты входов (readEnters()).

Теперь для каждого входа мы посчитаем его длину кратчайшего пути до клада через поиск в ширину(MyBfs()). Из полученных длин находим минимальную и выводим номер соответсвующего выхода.

2.8 O. Lines - 2

Lines - 2

(Время: 1 сек. Память: 16 Мб Сложность: 48%)

В таблице из N строк и N столбцов некоторые клетки заняты шариками, другие свободны. Выбран шарик, который нужно переместить, и место, куда его нужно переместить. Выбранный шарик за один шаг перемещается в соседнюю по горизонтали или вертикали свободную клетку. Требуется выяснить, возможно ли переместить шарик из начальной клетки в заданную, и, если возможно, то найти путь из наименьшего количества шагов.

Входные данные

В первой строке входного файла INPUT.TXT находится число N, в следующих N строках - по N символов. Символом точки обозначена свободная клетка, английской заглавной O - шарик, @ - исходное положение шарика, который должен двигаться, английской заглавной X - конечное положение шарика. ($2 \le N \le 250$)

Выходные данные

В выходной файл OUTPUT.TXT выведите в первой строке «Y», если движение возможно, или «N», если нет. Если движение возможно, то далее следует вывести N строк по N символов - как и на вводе, но буква X, а также все точки по пути следует заменить плюсами. Если решений несколько, выведите любое.

Сперва создадим два двумерных массива NxN – matrix и disM. В первый введём нашу таблицу символов из input.txt, а второй исходно заполним -1 (или заведомо большим чем длина пути числом)– он будет хранить длины путей до клетки i j.

Далее проходим поиск в ширину до искомой клетки, попутно заполняя disM. Если клетка не достижима – конец, выводим ‘N’. В ином случае выводим ‘Y’ и теперь переписываем matrix по такому алгоритму:

1) в конечную клетку записываем +.
2) Ищем соседнюю клетку длина пути до которой отличается от длины пути до нашей клетки на 1. Теперь это наша конечная клетка.
3) Повторяем первые два пункта пока не дойдём до начальной клетки.

Теперь выводим нашу matrix.

2.9 Q. Игрушечный лабиринт

Игрушечный лабиринт

(Время: 1 сек. Память: 16 Мб Сложность: 51%)

Игрушечный лабиринт представляет собой прозрачную плоскую прямоугольную коробку, внутри которой есть препятствия и перемещается шарик. Лабиринт можно наклонять влево, вправо, к себе или от себя, после каждого наклона шарик перемещается в заданном направлении до ближайшего препятствия или до стенки лабиринта, после чего останавливается. Целью игры является загнать шарик в одно из специальных отверстий – выходов. Шарик проваливается в отверстие, если оно встречается на его пути (шарик не обязан останавливаться в отверстии).

Первоначально шарик находится в левом верхнем углу лабиринта. Гарантируется, что решение существует, и левый верхний угол не занят препятствием или отверстием.

Входные данные

В первой строке входного файла INPUT.TXT записаны целые числа N и M – размеры лабиринта ($1 \le N, M \le 100$). Затем идет N строк по M чисел в каждой – описание лабиринта. Число 0 в описании означает свободное место, число 1 – препятствие, число 2 – отверстие.

Выходные данные

В выходной файл OUTPUT.TXT выведите единственное число – минимальное количество наклонов, которые необходимо сделать, чтобы шарик покинул лабиринт через одно из отверстий.

Для решения этой задачи также создаём два двумерных массива NxM – matrix и disM, и заполним их аналогично предыдущей задаче. Теперь реализуем поиск в ширину, с учётом специфик данной задачи.

Попав в 2, прерываем поиск и выводим ответ. Он хранится в массиве disM под соответствующими нашей 2 индексами.

2.10 S. Мосты

Мосты

(Время: 1 сек. Память: 16 Мб Сложность: 56%)

Вася живет в городе С.-П. Не нужно говорить (это итак все знают), что в городе С.-П. очень много мостов. И, как любому человеку, Васе часто нужно добираться из пункта А в пункт Б.

У Васи есть детальная карта города С.-П. Карта представляет собой квадратную решетку, размером N×M, где каждая ячейка – либо свободное пространство, по которому можно двигаться, либо препятствие (дома, водоемы, и т. п.), либо мост. Север, как обычно, находится вверху карты. Все мосты расположены с запада на восток (или с востока на запад). Это значит, что если Вася войдет в клетку с мостом с запада или востока, то он окажется на мосту, и выйти с этой клетки он может только на запад или на восток. Если же Вася войдет в клетку с мостом с юга или севера, то он окажется под мостом, и выйти сможет только на юг или на север.

Помогите Васе – посчитайте, сколько клеток карты ему нужно пройти, чтобы попасть из пункта А в пункт Б (умножить ваш ответ на масштаб карты Вася может и сам).

Входные данные

Первая строка входного файла INPUT.TXT содержит два натуральных числа N и M – размеры карты (N, M ≤ 100). Следующие N строк содержат саму карту (по M символов в каждой строке). Символ «.» соответствует пустому пространству, символ «#» – препятствию, «B» – мосту, «S» – стартовой точке маршрута, «E» – конечной точке маршрута. Стартовая и конечная точки находятся на пустом пространстве.

Выходные данные

В выходной файл OUTPUT.TXT выведите длину кратчайшего маршрута между начальной и конечной точкой или -1, если такого маршрута не существует.

Решение этой задачи очень похоже на два предыдущих, но для нее мы теперь создаём два массива disMx и disMy. В первый мы записываем длину пути до моста, если мы зашли на него по вертикали, а во второй - если по горизонтали. Достигнув «.» или E, мы записываем длину пути до неё в оба массива.

2.11 U. Только направо

Только направо

(Время: 1 сек. Память: 16 Мб Сложность: 60%)

Змей Горыныч оказался в лабиринте и хочет выбраться из него как можно скорее. К сожалению, после вчерашнего употребления кефира, левая голова Змея соображает плохо. Поэтому Змей Горыныч может поворачивать направо и идти прямо, но не может поворачивать налево и разворачиваться на месте. Помогите Змею Горынычу определить длину кратчайшего пути до выхода из лабиринта.

Входные данные

В первой строке входного файла INPUT.TXT через пробел записаны числа N и M ($4 \le N, M \le 20$) – количество строк и столбцов в карте лабиринта. В каждой из следующих N строк записано по M символов, задающих эту карту. Символ «S» обозначает положение Змея Горыныча, символ «F» – точку выхода из лабиринта, символ «X» – стенку. Пробелами обозначены пустые клетки. Гарантируется, что лабиринт окружен стенами, и Змей Горыныч всегда сможет выйти из лабиринта. Перед началом движения Змей Горыныч может сориентироваться по любому из 4 направлений (вверх, вниз, влево или направо).

Выходные данные

В выходной файл OUTPUT.TXT выведите единственное число – расстояние, которое придется пройти Змею Горынычу.

В этой задаче для поиска длины пути, нам необходимо по отдельности обрабатывать каждый случай, в зависимости от того с какого направления мы ступили на клетку. Поэтому создаём четыре массива disM и для каждого случая отдельно обрабатываем новый шаг и записываем расстояние в соответствующий массив.

2.12 W. Лабиринт минотавра

Лабиринт минотавра

(Время: 2 сек. Память: 32 Мб Сложность: 55%)

Вы слышали про игру «Лабиринт минотавра»? В этой игре бесстрашный герой Тесей пытается выбраться из запутанного лабиринта. Лабиринт имеет прямоугольную форму – N×M клеток. За один ход Тесей может перейти на соседнюю свободную клетку по вертикали или по горизонтали, либо, если он находится на краю лабиринта, покинуть его. Клетка может быть либо свободной, либо занятой стеной или дверью. Сквозь стены проход невозможен. Сквозь двери можно проходить только при наличии ключа, который подходит ко всем дверям и хранится где-то в лабиринте!

Для поиска выхода из лабиринта Тесей применяет специальную тактику:

- Сначала он пытается найти выход по свободным клеткам.
- Если Тесею это не удается – он пытается отыскать ключ, чтобы открывать им двери и искать путь на свободу.

По карте лабиринта требуется определить, сможет ли Тесей выбраться из лабиринта, или же ему придется остаться в этом ужасном месте.

Входные данные

Первая строка входного файла INPUT.TXT содержит два натуральных числа N и M – длина и ширина лабиринта ($2 \le N, M \le 1000$). Далее идет N строк по M символов – описание лабиринта. Свободные клетки обозначены символом «.», стены – символом «#», двери – символом «-», «T» – положение Тесея в начальный момент времени, «K» – свободная клетка с ключом. Гарантируется, что в лабиринте только один Тесей и только один ключ.

Выходные данные

В выходной файл OUTPUT.TXT выведите:

- Если Тесей может выйти за пределы лабиринта без ключа – минимальное количество ходов, которое для этого понадобится.
- Если не может – напечатать «No way» (без кавычек) и дополнительно через пробел вывести:
  - минимальное количество ходов, которое понадобится, чтобы добраться из начальной позиции Тесея до ключа;
  - «No key» (без кавычек) - если ключ недостижим.
  - Если ключ достижим, то следует дополнительно через пробел вывести:
    - минимальное количество ходов после взятия ключа, которое понадобится для того, чтобы покинуть лабиринт;
    - «No way» (без кавычек) – если даже с ключом нельзя выбраться из лабиринта.

Для этой задачи я написал функцию для нахождения кратчайшего пути через поиск в ширину, которая на вход получает lS – символ из которого надо прийти, и lF – символ, к которому надо прийти. Если lS = ‘T’, то функция считает двери за препятствия, если lS = ‘K’, то нет.

Также я организовал считывание лабиринта так, что он стал обрамлён символами ‘E’ – выходами.

Далее алгоритм такой:

1) Есть путь из T в E?
    1.1) Да – выводим My_BFS(T, E);
    1.2) Нет –выводим “No way”, переходим к пункту 2;
2) Есть путь из T в K?
    2.1) Да – выводим My_BFS(T, K), переходим к пункту 3;
    2.2) Нет – выводим “No key”;
3) Есть путь из K в E?
    2.1) Да – выводим My_BFS(K, E);
    2.2) Нет – выводим “No way”;

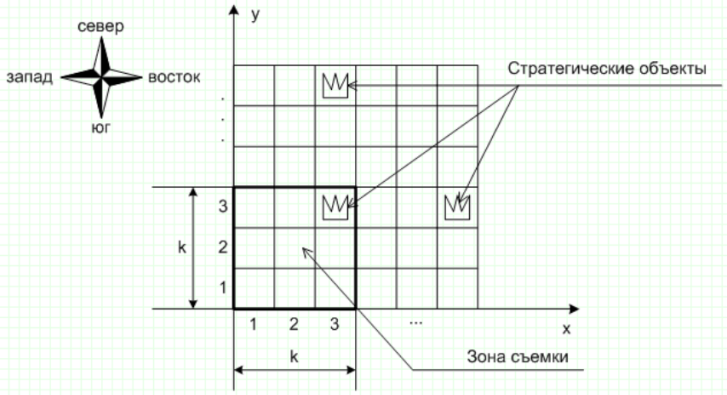
2.13 Y. Космические исследования

Космические исследования

(Время: 1 сек. Память: 16 Мб Сложность: 82%)

Отделу космических исследований поступило задание сфотографировать из космоса n объектов в заданной области. Область имеет форму квадрата размером 50×50 километров. Если разделить ее на квадраты размером 1×1 километр, то интересующие отдел объекты окажутся в центрах некоторых единичных квадратов.

Введем систему координат, направив ось OX с запада на восток и ось OY с юга на север. Тогда каждому единичному квадрату будут сопоставлены координаты в диапазоне от 1 до 50, как показано на рисунке ниже.



Для космической съемки используется специальный фотоаппарат высокого разрешения, установленный на космическом спутнике. Фотоаппарат может делать снимки квадратных участков земной поверхности размером k × k километров. Исходно аппарат наведен на юго-западный угол заданной области, то есть, если сделать снимок, на нем будут видны единичные квадраты с координатами x и y от 1 до k километров.

С помощью специальных двигателей можно изменять орбиту спутника, что приводит к изменению участка съемки. За один день орбиту спутника можно изменить таким образом, что участок съемки сместится либо на один километр на запад, либо на один километр на восток, либо на один километр на север. Переместить участок съемки на юг невозможно. Непосредственно между перемещениями спутника можно сделать снимок, временем съемки можно пренебречь.

Руководство отдела заинтересовалось вопросом: за какое минимальное количество дней можно сделать снимки всех объектов заданной области.

Требуется написать программу, которая по заданному расположению объектов и размеру снимка k определит минимальное время, за которое можно сделать снимки всех объектов заданной области.

Входные данные

Первая строка входного файла INPUT.TXT содержит два целых числа: n и k ($1 \le n \le 1000$, $1 \le k \le 5$). Следующие n строк содержат по два целых числа: x_i и y_i — координаты объектов в заданной области ($1 \le x_i, y_i \le 50$).

Выходные данные

В выходном файле OUTPUT.TXT должно содержаться одно целое число: минимальное количество дней, которое требуется для получения снимков всех объектов в заданной области.

Для решения этой задачи нам необходимо создать трёхмерный массив $disM_{ijm}$ длин пути размерности [52-k]x[52-k]x[2^k], где k – ширина кадра.

Далее обходим нашу область поиском в ширину заполняя массив длин пути, где i = x – посещённой клетки, j = y посещённой клетки, m = специальный код vis, который высчитывается через функцию calcVis(). Условием для выхода из поиска в ширину выступает то, что у нас не осталось объектов выше текущего положения кадра и то что текущий код vis = 2^k – 1, что означает что мы прошли все объекты ниже или на высоте кадра.

Заключение.

В ходе выполнения этого проекта я значительно улучшил свои навыки программирования на языке C++, глубже изучил некоторые встроенные библиотеки и классы. Была проделана большая работа, в ходе которой я развил свой кругозор и гибкость мышления.

Использованные источники

1. Блог «Программирование дилетанта» https://plusplusmustlive.blogspot.com/
2. Youtube-канал https://www.youtube.com/@35zvn