Лабораторная работа № 7 по курсу дискретного анализа: Динамическое программирование

Выполнил студент группы 08-207 МАИ *Днепров Иван.*

Условие

1. Вариант 2. Пустои прямоугольник.

2. Задан прямоугольник с высотои n и ширинои m, состоящии из нулеи и единиц. Наидите в нем прямоугольник наибольшои площади, состоящии из одних нулеи.

Метод решения

После получение параметров поля, объявления всех переменных и структур, я начинаю считывание, оно работает за $O(h \cdot l)$, где h – высота, а l – ширина поля. Для каждой строки в поле я зачитываю строку и посимвольно копирую её в двумерный вектор field.

За тем начинается обработка поля. Идея алгоритма, который я применил для решения этой задачи заключается в том, что при помощи дополнительного вектора размера ширины поля, заполняется этот вектор так: обнуляем элементы вектора, соответствующие элементам строки равным еденице, остальные увеличиваем на еденицу. После каждой такой итерации вычесляем максимальную площадь, участка, касающегося текщей строки и ограниченного ей.

Для этого я завожу вспомогательный стек индексов, который будет наполняться и опустошаться по мере работы алгоритма. В цикле пока мы не превысили длину вектора, увеличивая значение k от нуля, увеличиваем ячейку countingArray, соответствующую k или не опустошили стек, мы находим площадь на основе вектора. Если стек пуст или занчение из countingArray, соответствующее верхнему индексу в стеке, не превышает значение соответствующее текущему индексу, добавляем в стек текущий индекс k и увеличиваем его на единицу. В противном случае извлекаем элемент из стека и записываем его в currentIndex, если элементы в стеке закончились, то текущая площадь равна произведению элемента, соответсвующего currentIndex в countingArray и k. В противном случае площадь равна произведению элемента, соответсвующего currentIndex в countingArray и k - 1 - верхний элемент стека.

После чего проверяем привышает ли эта площадь максимальную, если – да, то перезаписываем значение максимальной площади.

По сути для каждого элемента, значение которого больше значения следующего элемента countingArray, мы вычисляем площадь прямоугольника из нулей, касающегося этого элемента и отсекаемого им. А из этих значений мы выбираем максимальное.

Описание программы

Все действи я происходят в main, така прогармма оказалась максимально компактной, я решил, что разбиение кода на несколько файлов и разделение его на функции только сделает его объемнее и усложнит читаемость.

Выводы

Перегон поля из поток ввода в двумерный вектор работает за $O(h \cdot l)$. На ыычисление countingArray для всех строк суммарно уходит тоже $O(h \cdot l)$ итераций, а вычисление площади работает быстрее за счёт того, что площадь приходится считать не для всех ячеек, а только для тех, значения которых больше значений следующих. Итоговая сложность получается всё равно квадратичной. Считтывание можно ускорить до линии и хранить поле не как двумерный вектор, а как вектор строк, но так как значения вспомогательного вектора всё равно должны быть вычисленны от каждого элемента, сложность останется $O(h \cdot l)$. Я не могу представить себе алгоритм, который бы работал быстрее чем за $O(h \cdot l)$, потому что этот алгоритм не будет рассматривать все элементы поля, по этому не понятно, как он будет находить максимальную площадь.

Хоть код этой программы занимает всего 60 строк, я потратил на него приличное количество времени, потому что долго не мог придумать алгоритм. В итоге я всё-таки нашёл подобный алгоритм в интернете и подстроил его под эту задачу.