Кубок Псковской области по программированию среди школьников — 2019
Финальный этап, первый дивизион

Разбор задач

A. Остатки сладки

Данная задача на тему модульной арифметики. В заочном туре была похожая задача, там было дано два остатка. Некоторые с этой задачей справились во время тура, был дан разбор решения.

Обозначим остатки как a, b, c, d, соответственно полученные при взятии исходного искомого значения на A, B, C, D.

Возьмём, например, первые два остатка a и b. Решим задачу только относительно этих двух величин. Как решить – указано в разборе к аналогичной задаче в заочном туре.

В результате решения будет найдено число e, которое при делении на A будет давать остаток a, при делении на B остаток b.

Тогда задача сведена к начальной, но с меньшим на один остатком, то есть теперь имеются остатки: e, c, d полученные при делении на E, C, D соответственно, где E равно произведению A на B.

Проделав вышеописанную операцию ещё два раза, задача будет решена. Единственное, надо заметить, что если ответ к подзадаче ищется перебором, то на «медленных» языках имеет смысл начать не с чисел, полученных в результате деления на малые величины, а с числа, полученного делением на 10000000.

B. Комплексный обед

В данной задаче нужно было обратить внимание на входные данные и понять, что i здесь не является переменной (как некоторые считали), а является всего лишь обозначением мнимой части комплексного числа.

Как произвести расчёты в явном виде указано в задаче, ограничения таковы, что все расчёты могут быть выполнены в double, аппаратно поддерживаемой арифметике («без подвоха»).

У некоторых возникли проблемы с выводом чисел с заданной точностью. На это хочется указать, что нужно выводить было с точностью не хуже указанной в задаче, то есть можно было выводить и до, например, 9 знака после запятой, главное, что бы ответ не отличался от истинного отдельно в действительной и мнимой частях более чем на 10^{-6}. Ещё, среди решений участников, встречалась ошибка, когда на промежуточных этапах расчёта были произведены округления до 10^{-6}, ошибки могли накапливаться и приводить к ошибочному результату.

C. Поиск столовой

Данная задача – типичная задача на поиск в ширину. Вообще, если требуется найти кратчайший путь в графе, то универсальным алгоритмом (с известными для этого алгоритма ограничениями) для этой задачи является поиск в ширину.

В данной задаче описание возможных направлений для каждой комнаты по факту определялось состоянием конкретного бита числа, то есть представляло собой массив флагов.

D. Миллиметровки как гуталина

Одним из возможных решений был способ воспользоваться «жадным алгоритмом», для этого, например, можно было перебирать знаки матрицы слева направо сверху вниз и для каждой позиции искать максимальное расширение прямоугольника вправо (влево не надо,

так как там было уже рассмотрено на предыдущем шаге). Далее, ниже, строка за строкой, искать полоски найденной ширины, увеличивая, таким образом, максимальную высоту очередного прямоугольника. Найдя очередной прямоугольник вывести его координаты и цвет. В матрице, все знаки, попавшие в прямоугольник можно было заменить, например, на «.» (точки) или пометить как-то по-другому. По окончанию работы такого алгоритма задача будет решена.

Эту задачу можно было решить с более малым выделением памяти, то есть запоминать набор данных соизмеримо с длиной одной строки и обрабатывать символ за символом, не сохраняя его в дальнейшем в памяти. Рекомендуем в качестве тренировки реализовать этот более эффективный по памяти алгоритм.

E. Непопулярное мнение

Данная задача аналогична задаче на поиск минимального и максимального значения в массиве. Осложнялась она требованием распознать строки разного типа и вычислением длительности между двумя датами. В остальном это поиск минимального и максимального значений.

F. Ключи и гайки

Так как в этой задаче ограничения малы, то рассматривать возможности оптимизаций здесь не будем. Алгоритмически следовало получить список всех размеров гаек и перебирая их искать ключ с таким же размером. Найдя гаечный ключ с нужным размером, так же проверить, не подходит ли он ещё к какой нужной гайке из списка, если да, то и её помечать, что бы повторно для неё не искать инструмент. Подсчитать всё множество найденных ключей.

Другой вариант, пронумеровать все ключи, напротив каждой гайки проставить номер ключа, который для неё подходит. Получив список ключей (их номера), оставить только уникальные. Количество уникальных номеров ключей в полученном списке и будет решением задачи.

Есть и другие варианты реализации.

G. Дорога

Первое, до чего следовало бы догадаться, что факт пересечения дороги и здания (строения) можно свести к проверке пересечения прямой (дорога) и отрезков (список рёбер многоугольников, описывающих каждое строение).

Вариантов определения факта пересечения отрезка и прямой может быть несколько. Один из вариантов, определить точку пересечения двух прямых, одна из которых соответствует дороге, другой принадлежит рассматриваемый отрезок. Найдя точку пересечения, проверить, принадлежит ли она отрезку. Есть вариант, что прямые не пересекаются, тогда они либо совпадают, что так же можно проверить, отрезку-дороге будут принадлежать крайние точки отрезка-грани или не совпадают и они параллельны, дополнительной проверки не потребуется если проверены предыдущие варианты.

Другой вариант – использовать знаковую площадь параллелограммов (ACMB и ADMB) образованных векторами: AC, AB и AD (см. рисунок). Точки A и B определяют прямую дорогу, эти данные даны в задаче. Точки C и D определяют отрезок (грань многоугольника из описания здания).

Тогда, если прямая проходит через одну из точек, с помощью которых задан отрезок, то площадь параллелограмма будет равна нулю и, соответственно, такое здание нужно будет

сносить. Если прямая линия (дорога) не пересекает отрезок, то обе площади будут иметь одинаковый знак (положительные или отрицательные значения площадей). Если пересекает, то площади будут иметь различные знаки.

Таким образом, здание нужно сносить, если при проверке хотя бы одной из граней многоугольника, описывающей это здание, выяснится, что прямая (дорога) имеет хотя бы одну общую точку с этой гранью.