Разбор задач 2-го тура окружного (муниципального) этапа
Всероссийской предметной олимпиады школьников по информатике

Задача A. Игроки

Переберем порядок, в котором расположены тесты. После этого посчитаем ответ для заданного порядка тестов. Найдем наилучший из них. Сложность такого алгоритма O(N!)
Домашнее задание. Придумать решение, которое работает для данной задачи за время O(N). Придумать решение, которое работает для произвольного количества участников M <= 10.

Задача B. Сборщики мусора

Сначала определим, кто из волонтеров может быстрее всех добраться до последнего ряда клеток. Для этого построим кратчайшие пути от клеток нижнего (последнего) ряда до клеток верхнего ряда (это делается «классическим» динамическим программированием). Найдем среди них самый короткий. Волонтер, изначально стоящий в клетке, в которую приводит самый короткий путь от нижнего ряда, может пройти этим путем, и никакой другой волонтер не сможет «перейти» ему «дорогу». Восстановим путь, устанавливая в пройденных им клетках значение G+1 (в том числе и в качестве флажка). Теперь, с учетом измененной карты вновь найдем кратчайший путь от нижнего ряда до верхнего. Разумеется, клетку, которая была выбрана в качестве стартовой на предыдущем этапе, следует исключить из рассмотрения.

Описанная процедура повторяется до тех пор, пока не будет определен кратчайший путь для последнего волонтера. Рекомендуется детально воспроизвести рассуждения для всех трех примеров, приведенных в условии задачи.

Так, в частности, в первом примере быстрее всех добраться до нижнего ряда может волонтер, исходно стоящий в 3-ем столбце (напомним, что когда несколько волонтеров желают одновременно занять одну и ту же клетку, преимущество имеет волонтер с меньшим номером). Первый и второй волонтеры не успевают опередить 3-его в выборе наименее загрязненного пути. Поэтому для каждого из них оптимальным выбором будет движение по своему столбцу. Что касается волонтера #4, то он, избегая «штрафных» клеток, вынужден будет свернуть на третий столбец. Пятому волонтеру также придется «уходить» от «штрафа», поскольку по его столбцу прошел третий волонтер, однако он окажется в еще менее выгодном положении, нежели четвертый.

Во втором примере всем волонтерам также удается миновать «штрафные» клетки. А вот в третьем примере волонтерам не избежать прохода по «штрафным» клеткам, в результате чего часть мусора оказывается неубранной.

Задача C. Заплыв

Будем действовать по следующему алгоритму. Начем с «заплыва» от 1го до 1го буйка. Далее, если заплыв не дольше времени T, то будем заканчивать заплыв на один буёк дальше, а иначе будем начинать заплыв на один буёк позже. Ответом будет самый длинный встретившийся нам заплыв, длительность которого не превосходит максимально допустимую.

Продемонстрируем это на втором примере из условия задачи.

Сначала выясним, какой буек Витя минует последним, если войдет в воду строго напротив первого буйка. Легко видеть, что по прошествии 25 единиц времени он окажется возле 6-го буйка. Чтобы добавить в рассмотрение 7-ой буек, необходимо, чтобы у Вити в «запасе» оставалось не менее 6 единиц времени, когда он окажется возле 6-го буйка. Если исключить из рассмотрения первый буек, и предполагать, что Витя войдет в воду напротив второго буйка, то

возле 6-го буйка он окажется через 23 (= 25 — 2) единицы времени. Этого недостаточно, и мы исключим из рассмотрения второй буек (путь от 3-го до 6-го займет 20 (= 23 — 3) единицы времени). Однако и после этого Витя не успеет доплыть до 7-го буйка. Поэтому приходится убрать из рассмотрения и третий буек. Путь от 4-го до 6-го буйка преодолевается за 16 (= 20 — 4) единиц времени. Теперь к этому времени можно прибавить 6 единиц, позволяющих Вите добраться до 7-го буйка. Однако оставшихся в «запасе» 3 единиц (= 25 — (16 +6)) недостаточно, чтобы достичь 8-го буйка, и придется исключать из рассмотрения очередной буек из начала рассматриваемого диапазона. Надо также обратить внимание на правильную обработку ситуации, когда расстояние между буйками превышает допустимое время плавания. Так, в рассматриваемом примере входных данных время, за которое можно проплыть между 18-ым и 19-ым буйками составляет 28, а значит, 19-ый буек может быть только стартовым.

Выполняя включение новых буйков в диапазон, нужно проверять, не удалось ли превзойти ранее найденный максимум. При этом неравенство должно быть строгим, так как буйки рассматриваются по возрастанию номеров, а при прочих равных необходимо вывести вариант, в котором стартовый буек имеет наименьший номер.

Задача D. Грузите вентиляторы бочками

Задача носит технический характер. На рисунке показано, как искать ответ для клетки (0,2)

21	22	23	24	25
20	7	8	9	10
19	6	1	2	11
18	5	4	3	12
17	16	15	14	13

Для того чтобы найти, какое число находится в клетке (i,j), сделаем следующее:

1. Найдем d=max(|i|,|j|), это будет "номер" квадрата в котором находится клетка.
2. Добавим сразу в ответ число (2*d — 1)^2, это будет "внутренняя" область квадрата.
3. Выясним, лежит ли точка на правой (d = i), левой (—d = i), нижней (—d=j) или верхней (d=j) границе квадрата.
4. Проверим отдельно 4 группы точек:

а) лежащие на правой, но не на верхней границе;
б) лежащие на нижней, но не на правой границе;
в) лежащие на левой, но не на нижней границе;
г) лежащие на верхней, но не на левой границе.

5. Для каждой из этих групп несложно получить выражение для номера точки в группе, и по нему — ответ задачи.

Задача E. Дорога и облака

Чтобы отыскать наиболее загрязненные отрезки на прямой, применим следующую технику. Спроецируем на прямую все отрезки-облака, обозначая начало очередного отрезка открывающей скобкой, а конец — закрывающей. Тогда при каждом прохождении через открывающую скобку загрязнение будет увеличиваться, а при прохождении через закрывающую скобку — уменьшаться. На рисунке показана скобочная последовательность, приблизительно соответствующая рисунку в тексте задачи. Голубым цветом показаны скобки, расположенные в одной точке (первая пара — открывающая и закрывающая скобки, вторая пара — обе скобки

открывающие, третья пара — обе скобки закрывающие).

Опишем структуру данных, позволяющую хранить координату точки и признак того, является ли она открывающей или закрывающей (удобно приписать «открытию» значение +1, а «закрытию» значение —1). Это может быть массив записей (в Pascal / Delphi), массив структур в C++, список объектов в Java и т.п.— реализации могут заметно различаться; можно также использовать два целочисленных массива (один для хранения координат, другой для хранения признаков).

Отсортируем точки по неубыванию координат, при этом «открывающая» точка при равных значениях координат должна оказаться левее «закрывающей». Обозначим как Z текущее загрязнение дороги. До первой «открывающей» точки Z = 0. Теперь достаточно пройти по отсортированной последовательности точек, прибавляя к Z единицу, если точка «открывающая», и вычитая из Z единицу, если точка «закрывающая», и определить, какое максимальное значение может принять Z при учете всех отрезков-облаков. Это максимальное значение Z и будет искомым A. Определение отрезков, на которых загрязнение максимально, осуществляется вторым проходом по описанной последовательности точек. Заметим, что отрезок может вырождаться в точку (о чем свидетельствует второй пример в условии задачи).