Волчёнков Сергей Геннадьевич
Кандидат технических наук, доцент Ярославского государственного университета им. П.Г. Демидова, заместитель председателя жюри Международной олимпиады школьников «Туймаада» по информатике.

Некоторые особенности задач олимпиады «Туймаада» по информатике

В статье рассмотрены избранные задачи по информатике, дававшиеся в разные годы на олимпиаде «Туймаада». В качестве решений приведены идеи алгоритмов, реализовать которые на любом высокоуровневом языке программирования читатель сможет самостоятельно.

Олимпиада «Туймаада» проводится по нескольким предметам, один из которых — информатика. Формат проведения в основном совпадает с форматами Международной и Всероссийских школьных олимпиад по информатике. Соревнования проводятся в два тура, в каждом из которых школьникам предлагается решить 3 задачи. После каждого тура проходит проверка написанных школьниками программ по заранее разработанной системе тестов. Оценка за задачу равна сумме оценок успешно пройденных тестов. Однако регламентация олимпиады «Туймаада» не такая строгая и позволяет в ряде случаев давать задачи нетрадиционные, а значит, в каком-то смысле более интересные.

В данной статье будут приведены три такие задачи, каждая из которых давалась на «Туймааде» в разные годы. Условия задач будут приведены в более вольной, по сравнению с олимпиадой, формулировке. Не будут приводиться некоторые ограничения по тестированию, примеры правильной работы программ, а также технические требования на входные и выходные данные. Первая задача, наряду с элементами вычислительной геометрии, является комбинаторной. Вторая — комбинаторная, имеющая отношение к области поиска информации с неточно задаваемыми условиями поиска. Третья задача, хотя и решается методом стандартного динамического программирования, требует нескольких неожиданных логических ходов.

Задача 1. «Салфетки» (2000). На столе лежат несколько круглых салфеток единичного радиуса. Известно, что границы (окружности) никаких

трёх салфеток не проходят через одну точку. Требуется определить:
1) можно ли все салфетки приколоть к столу одной кнопкой;
2) какое наибольшее количество салфеток можно приколоть одной кнопкой;
3) каким наименьшим количеством кнопок можно приколоть все салфетки.

Опишем решение задачи по каждому пункту.

1. Заметим, что если салфетка всего одна или какие-то салфетки не имеют пересечений с другими салфетками, то соответствующая ситуация легко проверяется и ответ на поставленный вопрос очевиден.

Если существует точка, принадлежащая всем салфеткам, то существует область, ограниченная дугами окружностей и принадлежащая всем салфеткам. В этом случае и граница этой области принадлежит всем салфеткам, то есть для установления такой ситуации достаточно проверить, принадлежит ли всем салфеткам каждая точка пересечения любых двух окружностей. Если хотя бы одна точка принадлежит всем салфеткам, то ответ на поставленный вопрос положительный, иначе — отрицательный.

2. Замечания пункта 1 и в этом случае остаются в силе. Отличие состоит только в том, что при проверке каждой точки на принадлежность кругам требуется считать количество кругов, которым точка принадлежит. Ответ — максимум из полученных результатов.

3. Это самый интересный из случаев. Алгоритма, находящего точный ответ, кроме переборного, не известно. Поэтому для получения ответа необходимо либо осуществить перебор, либо применить какой-либо эвристический подход. В первом случае остаётся надеяться, что перебор уложится в отведённое для тестирования время, во втором — что эвристика выдаст правильный, а не приближённый ответ. Именно такие задачи неохотно дают на официальных олимпиадах.

Какие же соображения необходимо учитывать при решении этой задачи?

Во-первых, если салфетка не пересекается с другими, то на неё необходимо выделить отдельную кнопку. Значит, все такие салфетки можно удалить, а в конце скорректировать полученный ответ.

Во-вторых, если салфетку необходимо прикалывать кнопкой, то будет не хуже, если прикалывать её вместе с какой-то другой салфеткой. В частности, если салфетка пересекается только с одной другой салфеткой, то одну кнопку надо воткнуть в пересечение этих областей, после чего обе салфетки можно удалить из рассмотрения.

Возникает идея выделения областей — общей части нескольких салфеток, в которые надо втыкать кнопки. К сожалению, эти области уже не являются такими простыми, как начальные круги, и поэтому работать с ними лучше без учёта геометрических свойств. Каждая такая область характеризуется множеством салфеток, которые будут приколоты, если кнопка будет воткнута в эту область.

В-третьих, если одно из рассмотренных множеств (K) является подмножеством другого (M), то K можно не рассматривать, так как при необходимости его прикалывания не худшего результата можно добиться при прикалывании множества M.

С учётом всех замечаний приведём один из возможных алгоритмов решения задачи.

Перебираем всевозможные точки пересечения двух окружностей и образуем множество номеров салфеток, которым эта точка принадлежит. Включаем это множество в список множеств, если в списке ещё нет большего множества. Под бóльшим множеством понимается такое, которое включает данное в качестве подмножества. Для организации быстрой проверки и включения в список рекомендуется список хранить в лексикографическом порядке.

После составления полного списка задача оказывается эквивалентной выбору из списка наименьшего количества множеств, объединение элементов которых содержит все номера салфеток. Вообще-то, эта задача является переборной, но несколько замечаний делает применение перебора достаточным для положительного результата на всех тестах. Во-первых, количество множеств в списке не так уж велико, так как каждая область является одной из областей, на которые исходные окружности делят плоскость, и таких областей не очень много. Во-вторых, если при организации полного перебора методом *backtracking* начинать с множеств, имеющих наибольшее количество элементов, то перебор проходит достаточно быстро.

Задача 2. «Отгадай число» (2002). Ваша программа должна отгадать натуральное число, задуманное жюри. Об этом числе заранее известно только то, что оно не превышает некоторого заданного N. Вы задаёте вопросы, в которых указываете некоторый числовой отрезок, и если загаданное число входит в этот отрезок (включая концы), жюри отвечает «Y», а если не входит, то — «N». После нескольких таких вопросов программа должна сообщить загаданное число. Коварство жюри заключается в том, что на один из вопросов (неизвестный нам) жюри может дать неправильный ответ. Количество вопросов должно быть по возможности меньшим.

Отметим, что поставленная таким образом задача не может быть полностью протестирована. Действительно, программа случайно может сразу указать загаданное число и выиграть данный тест с большим преимуществом. Более или менее объективную картину может дать только большое количество тестов. По этой причине подобная задача вряд ли появится на большинстве олимпиад, несмотря на ряд интересных идей, которые могут быть реализованы при её решении.

Итак, что можно предложить для эффективного отгадывания числа? Если бы неверных ответов не было,

то оптимальным алгоритмом поиска был бы бинарный поиск, при котором каждый вопрос сужает в два раза зону поиска. К сожалению, неверный ответ на любом шаге сбивает с верного пути, что не приводит к получению правильного ответа. Сохранить идею бинарного поиска может следующая тактика.

Тактика 1. Задаём каждый вопрос два раза. В случае одинаковых ответов делаем вывод, что этот ответ правильный, иначе задаём вопрос третий раз. Легко видеть, что такая тактика может увеличить количество вопросов в два раза.

Тактика 2. Разобьём отрезок A, на котором находится задуманное число, на четыре равных: $A1$, $A2$, $A3$, $A4$. В первом вопросе указываем объединение $A1$ и $A2$. Во втором — объединение $A2$ и $A3$. Если бы ответы на оба вопроса были правдивыми, то нам был бы точно указан один из четырёх интервалов. Назовём его I. Третьим вопросом указываем интервал I. В случае положительного ответа мы убеждаемся, что задуманное число действительно принадлежит интервалу I. Если бы это было не так, то неверных ответов было бы два. В этом случае интервал поиска сузился в 4 раза за 3 вопроса, что лучше тактики 1. Отрицательный ответ на третий вопрос означает, что среди первых двух ответов один был ложным, а третий правдивый, то есть последующие ответы будут правдивыми, и можно применять бинарный поиск. Это даст быстрые преимущества, несмотря на то, что за первые 3 вопроса интервал поиска уменьшился незначительно.

Тактика 3. Применяем метод бинарного поиска, не обращая внимания, что среди ответов может быть неверный. Возможны следующие случаи.

1. Все ответы действительно были правдивыми. В этом случае поиск идёт по оптимальному пути.

2. Среди ответов один был ложным. Нетрудно заметить, что все последующие (правдивые) ответы будут отрицательными, так как все наши вопросы будут касаться интервалов, на которых заведомо нет задуманного числа. Будем бить тревогу, если на K подряд поставленных вопросов последовали отрицательные ответы. В этот момент требуется проверить, идём ли мы по правильному пути. Ложными ответами могут оказаться последнее «да» или один из последующих ответов. Эти вопросы можно повторить. В этой тактике главным является правильный выбор числа K.

Итак, мы описали тактики, которые можно применять при решении поставленной задачи. Какую из них вы-

брать? Доказать теоретически преимущество какой-то из них перед остальными трудно. В таком случае рекомендуется воспользоваться возможностью, которую предоставляет компьютер — во время тура провести эксперимент и сравнить их между собой. Заметим, что и подбор K в тактике 3 тоже должен осуществиться экспериментально.

Задача 3. «Пираты» (2005). Пираты захватили богатую добычу — несколько мешков с одинаковыми золотыми монетами. Всего K монет. После того как их радость улеглась, они приступили к дележу. Для этого все N пиратов пересчитались и выбрали для дележа человека с наименьшим номером. Тот разложил захваченные монеты на N кучек и сообщил, какую кучку должен взять себе каждый из пиратов. Если пират оставался доволен доставшейся ему долей, он высказывал своё одобрение, если нет, то высказывался против. Если против предложенного раздела высказались половина или больше всех пиратов (включая производившего делёж), то производившего делёж убивали и доверяли делить добычу следующему по номеру.

Перед тем как высказать своё одобрение или неодобрение, каждый пират тщательно продумывал своё решение и высказывал одобрение только в том случае, если опасался, что после казни делящего он тоже будет убит или получит меньше денег. Очевидно, что делящий добычу тоже прежде всего заинтересован остаться в живых и при этом получить максимальную долю.

Требуется определить, останется ли в живых первый пират и найти количество монет, которое ему достанется.

Задача, на первый взгляд, кажется знакомой. Действительно, по крайней мере две её вариации широко обсуждались в узких кругах и даже были представлены в интернете. По этой причине предлагать такую задачу на официальных олимпиадах не принято. Однако кажущееся незначительным изменение условия в правилах голосования пиратов по сравнению с известными вариантами делает её другой.

Начнём рассмотрение с нескольких простейших случаев.

Так, если пиратов всего двое, то, как бы первый ни делил монеты, второй проголосует против такого дележа и заберёт себе всё.

Если же пиратов трое, то первый разделит добычу так: себе — всё, второму и третьему — ничего. Второй будет голосовать за такой раздел, потому что в случае казни делящего его тоже неминуемо казнят.

При большем количестве пиратов у делящего добычу появляются дополнительные соображения, как привлечь на свою сторону голоса пиратов. Для этого в варианте дележа он некоторым пиратам должен предложить такое количество монет, при котором те получат больше денег, чем в случае отказа от предложенного варианта (подкупить некоторых).

Казалось бы, что подобные рассуждения помогут решить задачу и для больших N. В частности, так было во

всех ранее обсуждавшихся вариантах этой задачи. В данном же случае в игру вступает краевой эффект. Что будет, если денег для подкупа не хватит? Можно ли в этом случае остаться в живых? Оказывается, что иногда можно.

Разберём ещё несколько случаев.

При $N = 4$ делящему добычу надо получить два голоса кроме своего, чтобы остаться в живых. Для этого ему достаточно «подкупить» двух последних пиратов, дав им по одной монете. Второго подкупать бессмысленно (см. случай $N = 3$). Делящему при этом достанется всё, кроме двух монет. Ясно, что если двух монет у него нет, то нет и шансов остаться в живых.

При $N = 5$ делящему добычу надо тоже получить два голоса, кроме своего, чтобы остаться в живых. Как он может их получить, затратив минимум денег? Ясно, что один голос можно получить, отдав одну монету третьему с конца, который в случае убийства делящего не получит ничего. Второй же голос он может получить, предложив две монеты вместо одной одному из последних двух пиратов. Другому при этом давать ничего не надо. Итак, первый получит все монеты, кроме трёх. Заметим, что один из последних двух пиратов получит две монеты, а другой — ни одной. Гарантий получить что-либо у них нет, поэтому в случае $N = 6$ они, получив хотя бы по монете, будут голосовать «за».

Таким образом, чтобы получить большинство голосов в случае $N = 6$, делящий подкупает четвёртого с конца и двух последних пиратов, предложив им по одной монете.

Мы видим, что для разбора каждого нового случая нам надо рассматривать результат предыдущего, то есть решать задачу динамически. При этом кажется, что при нехватке монет, необходимых для подкупа, делящего ничего не может спасти. Однако в случае, если его казнят, в аналогичном положении может оказаться второй пират! Того такая перспектива, очевидно, не устраивает. И он, чтобы её исключить, проголосует «за», даже не получив ни одной монеты. Таким образом, в исследование надо включать и эту возможность. Все эти соображения делают задачу достаточно увлекательной и сложной — ведь все ситуации надо предусмотреть в программе!

Кроме указанного динамического способа решения этой задачи можно было бы, потратив больше времени, продолжить разбор случаев и заметить повторяющуюся закономерность результата, которую и запрограммировать. Читателям мы советуем провести эти исследования и получить удовольствие.

Если у вас есть какие-либо другие идеи решения этих задач, то высылайте их на info@potential.org.ru. Наиболее интересные из них будут опубликованы в журнале.