Ворожцов Артём Викторович
тренер сборной команды по программированию МФТИ

О доминошках, дружбе мальчиков и девочек и трубопроводе

Здесь мы расскажем про задачу «Замощение доминошками», которая предлагалась на IV личной олимпиаде МФТИ по программированию 11 апреля 2004. Автор этой задачи — Барский Евгений. Это яркий пример задачи, которая неожиданным образом сводится к классическим задачам «Максимальное паросочетание» и «Максимальный поток».

Задача про замощение фигуры доминошками

Из клеточного листа бумаги размера $N \times N$ вырезали некоторую фигуру. Доминошки — это плитки размера 2×1. Замостите фигуру доминошками, а если её невозможно замостить целиком, то положите максимальное число доминошек (каждая доминошка занимает две соседние клетки).

Формат входных данных

В первой строчке указано число N,

$$2 \leqslant N \leqslant 20,$$

а затем идут N строчек по N символов '#' или '.' Знак '#' означает, что соответствующая клетка листа вырезана.

Формат выходных данных

Выведите одно число — максимальное число доминошек, которое можно положить на лист бумаги.

Примеры:

Вход №1:	Выход №1:
2 ## #.	0

Вход №2:	Выход №2:
2	2

Вход №3:	Выход №3:
3 ... #.# ...	2

Идеи решения

Рассмотрим различные примеры входных данных. На рисунке 1 приведено три листка бумаги, из которых вырезаны некоторые клетки.

Видно, что на первый кусочек можно положить без перекрытия только две доминошки (этот случай соответствует входу №3). Возможное положение доминошек обозначено чёрточками. На вторую фигуру из клеток можно положить пять доминошек, при этом останется две незакрытые доминошками клетки, на третью — одиннадцать. Понятно, что число доминошек меньше либо равно числу клеток в фигуре пополам.

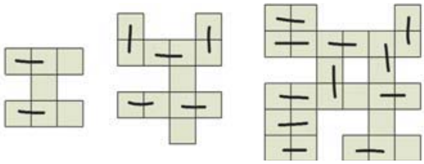

Рис. 1

Один из возможных алгоритмов замощения фигуры доминошками может быть таким: находим пару соседних непокрытых клеток и кладем на них сверху доминошку; затем снова ищем пару соседних непокрытых клеток и кладём на них доминошку и так далее, пока не получим ситуацию, в которой нет соседних непокрытых клеток.

1. Приведите пример, когда такой алгоритм находит неоптимальное решение, то есть располагает на фигуре меньше доминошек, чем в принципе возможно.

Приведенный нами алгоритм принадлежит к классу жадных алгоритмов. Жадные алгоритмы на каждом шаге ищут улучшение ситуации (например, первое попавшееся). Иногда жадным алгоритмам удается найти оптимальное решение, а иногда – нет. Для некоторых задач придуманы жадные алгоритмы, которые всегда находят оптимальное решение. Но для указанного алгоритма это не так.

Идея точного решения этой задачи основана на методе раскраски. Представим себе, что клетки бумаги раскрашены в чёрный и белый цвет, как клетки шахматной доски. Заметим, что доминошка покрывает одну чёрную и одну белую клетку.

Будем говорить, что чёрная и белая клетка спарены, если на них лежит доминошка. Каждая клетка может быть спарена с одной из соседних клеток противоположного цвета. Можем подсчитать число чёрных и белых клеток. Очевидно, что количество доминошек не может быть больше, чем минимум из этих двух чисел. Заметим, что для всех трёх случаев, изображённых на рисунке 1, данная верхняя оценка в точности совпадает с ответом — максимальным числом доминошек.

2. Приведите пример связной фигуры, для которой минимум из числа белых и чёрных клеток не является ответом на поставленную задачу.

Вот главный вопрос, на который попробуем дать ответ:

? Как написать программу, которая всегда находит оптимальное решение?

Задача про максимальное паросочетание

Это задача про девочек и мальчиков. Пусть у нас есть M мальчиков и D девочек. Известно, кто с кем дружит. Каждый мальчик может дружить с несколькими девочками, и каждая девочка может дружить с несколькими мальчиками.

Нужно получить как можно больше пар мальчик–девочка, в которых мальчик и девочка дружат. Каждый ребенок может быть только в одной паре. При этом некоторые дети могут остаться без пары.

Обозначим девочек цветочками, а мальчиков – звездочками. Цветочек будем соединять линией со звездочкой, если соответствующие девочка и мальчик дружат. Мы получим то, что в дискретной математике принято называть двудольным графом. Некоторые из этих линий нам нужно сделать жирными. Это будет означать, что соответствующую пару мы выбрали. Из каждой вершины может исходить только одна жирная линия (каждый ребенок может быть только в одной паре). Нам нужно получить как можно больше жирных линий. Будем решать эту задачку так:

1. Найдем какое-нибудь разбиение на пары. Просто будем перебирать мальчиков и

смотреть, есть ли свободная девочка, с которой он дружит. Таким образом, мы получим несколько пар. Если все мальчики оказались задействованы, то мы нашли оптимальное решение. Если нет, то переходим к пункту 2.

2. У нас есть некоторое число незадействованных мальчиков. Попробуем сделать перестановку в парах, которая позволит добавить еще одну пару. Для этого попробуем найти такую цепочку:
(свободный мальчик m_1) $\to$ (девочка d_1) $\Longrightarrow$ (её мальчик m_2) $\to$ (девочка d_2) $\Longrightarrow$ (её мальчик m_3) $\to \ldots \to$ (свободная девочка d_k)

В этой цепочке соседние элементы дружат. Жирная стрелочка от девочки к мальчику обозначает, что на текущий момент эта пара у нас выбрана, а нежирная стрелочка означает, что соответствующая пара дружит, но не выбрана.

Если мы найдем цепочку, которая начинается со свободного мальчика и заканчивается свободной девочкой, то мы сможем сделать следующую перестановку в парах: жирные стрелочки поменять на нежирные, а нежирные сделать жирными. Например, если бы мы нашли цепочку

$$m_1 \to d_1 \Longrightarrow m_2 \to d_2 \Longrightarrow m_3 \to d_3,$$

то мы могли бы ее преобразовать в

$$m_1 \Longrightarrow d_1 \to m_2 \Longrightarrow d_2 \to m_3 \Longrightarrow d_3.$$

Заметьте, что число жирных стрелочек (число выбранных пар) стало больше на одну — было две, а стало три.

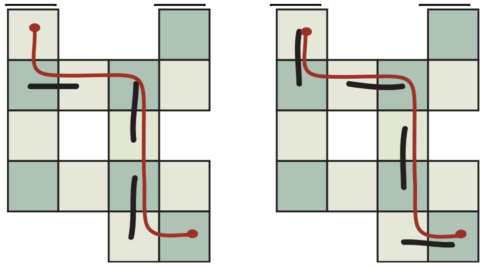

Рис. 2

Таким образом, пока такие цепочки есть, мы можем увеличивать число пар. Но в какой-то момент мы обнаруживаем, что таких цепочек больше нет, и мы не можем увеличить число пар. Можно показать, что полученное число пар максимально.

Но при чём здесь доминошки? Вы уже наверное догадались сами. Пусть чёрные клетки будут мальчиками, а белые — девочками, а соседство клеток обозначает дружбу, тогда задача про доминошки превращается в задачу про максимальное паросочетание.

Поиск указанной выше цепочки из мальчиков и девочек соответствует последовательности соседних клеток фигуры. В этой последовательности первая и последняя клетка свободны, все внутренние клетки покрыты доминошками, причём каждый чётный шаг в этой цепочке делается вдоль доминошки. На рисунке 2 показано, как работает шаг оптимизации цепочки, приводящий к увеличению числа доминошек.

Как находить цепочки?

Нам нужно найти цепочку от одного из свободных мальчиков к одной из свободных девочек. Для этого имеет смысл использовать поиск в ширину в графе. В этом графе из мальчиков исходят ребра (линии), соответствующие дружбе, а из занятых девочек — только одно ребро (линия), ведущее к мальчику, который с ней в паре (на текущий момент).

Поиск в ширину школьники часто называют «волновым алгоритмом». В начале работы алгоритма «фронт волны» состоит из свободных мальчиков. На первом шаге мы находим всех девочек, которые дружат с одним из свободных мальчиков. Если среди этих девушек есть свободные, то нужная цепочка найдена.

На следующем шаге мы берём мальчиков, которые находятся в парах (на текущий момент) с найденными девочками. Эти мальчики представляют собой следующий фронт волны.

Тех мальчиков и девочек, через которых прошёл фронт волны, нужно помечать, чтобы второй раз их не рассматривать.

В конце концов, мы либо доберемся до свободной девочки, либо на некотором шаге не найдем ничего нового, и фронт волны окажется пустым.

Заметим, что при реализации этого алгоритма удобно использовать структуру данных «очередь». В конец этой очереди мы будем класть новых детей, до которых мы смогли добраться по цепочкам. А из начала очереди мы будем последовательно доставать ребенка, чтобы исследовать, куда из него можно сделать следующий шаг в цепочке. Новеньких найденных (тех, которые еще не были в очереди) будем класть в конец очереди, не забыв при этом пометить их как «положенные в очередь».

Из каждой занятой девочки исходит только одно ребро — к её текущему мальчике-паре, поэтому вместо девочки в очередь можно сразу класть её мальчика.

Но у нас остался неразобранным важный вопрос:

> ? Верно ли, что последовательная оптимизация цепочек всегда приведёт к максимально возможному количеству пар?

Задача про трубопровод

Пусть у нас есть система соединённых труб друг с другом. Её можно представлять как множество узлов, соединённых линиями. Линии – трубы, узлы — это места соединения труб. Для каждой линии указана её пропускная способность, то есть сколько единиц жидкости эта труба может пропускать в единицу времени.

Среди узлов один помечен как исток, а другой как сток. Задача: найти максимальное количество единиц жидкости, которую можно пропускать через эту систему труб от истока к стоку.

Задача про максимальный поток в сети — одна из самых сложных алгоритмических задач. Впрочем, один из методов её решения можно пояснить в нескольких предложениях.

Метод Форда-Фалкерсона. Пусть у нас есть некоторый поток, который течёт от истока к стоку. Попробуем увеличить его. Для этого найдём путь от истока к стоку по трубам, которые еще не до конца заполнены жидкостью то есть пропускные способности кусочков труб в этом пути больше, чем поток жидкости, который по ним течёт.

По найденному пути мы пустим дополнительный поток жидкости, максимально возможный. В результате этого одна из труб заполнится жидкостью до конца и через неё в следующий раз мы уже не сможем пропустить жидкость в этом направлении. Зато в обратном направлении её пропускная способность как бы удвоилась.

Удобно ввести обобщённые пропускные способности труб. Обобщённая пропускная способность трубы — это два числа, которые показывают проводимость трубы в двух направлениях.

Если труба в обе стороны способна проводить не более 2 единиц жидкости в единицу времени, то мы будем писать, что её обобщенная пропускная способность равна отрезку $[-2{,}2]$. У каждой трубы одно направление условно обозначим как положительное, а другое — как отрицательное. Если в отрицательном направлении труба проводит 5 единиц, а в положительном — 10 единиц жидкости, то её пропускная способность есть $[-5{,}10]$.

Пусть по трубе $[-10{,}10]$ пустили поток жидкости размера 5 в положительном направлении. Это можно отобразить как изменение её обобщенной проводимости: теперь в положительном направлении можно пустить еще максимум 5 единиц, зато потенциальная пропускная способность в другую сторону увеличилась. Она стала равна 15: поток в 5 единиц мы можем убрать и еще 10 пустить. Итого, можно считать, что труба $[-10{,}10]$ с потоком размера 5 в положительном направлении, превратилась в трубу $[-15{,}5]$. Из обеих границ обобщённой пропускной способности вычитается поток, который через неё идёт.

Таким образом, при построении алгоритма мы можем работать с обобщёнными пропускными способностями. Каждый раз, когда мы пускаем жидкость по найденному пути, мы будем просто менять обобщённые пропускные способности труб и получать новую сеть труб (с новыми параметрами), по которым (как будто бы) ничего не течёт.

Итак, мы ищем путь от истока к стоку, двигаясь по трубам в направлении, в котором они могут пропустить какое-то количество жидкости (их текущая пропускная способность в этом направлении положительна). Пускаем по этому пути максимально возможный поток жидкости, после чего одна из труб в этом пути оказывается заполненной до конца (самое узкое звено). Меняем обобщенные пропускные способности и снова повторяем шаг поиска пути. Так делаем, пока такой путь существует. В какой-то момент мы не сможем добраться по пропускающим трубам от истока к стоку.

Путь от истока к стоку можно искать методом поиска в ширину (это уже упомянутый нами волновой алгоритм; вспомните также, как ищется кратчайший путь в лабиринте). Этот метод будет находить наиболее короткий путь, что хорошо, так как только в этом случае гарантируется, что наш алгоритм будет работать не слишком большое число шагов.

3. Предложите алгоритм поиска самого толстого пути по трубам от истока к стоку (толщина пути равна минимуму из пропускных способностей кусочков труб, по которым он идет).

? Почему метод Форда-Фалкерсона найдёт максимально возможный поток? Другими словами, почему, действуя указанными локальными изменениями имеющегося потока, можно получить глобальный максимум для потока от истока к стоку?

Это можно пояснить довольно просто. Итак, мы ищем путь от истока к стоку по трубам, двигаясь по трубам в том направлении, в котором они имеют положительную проводимость. На последнем шаге алгоритма мы обнаруживаем, что такого пути нет. Рассмотрим множество вершин A, до которых мы можем добраться от истока. Обозначим множество остальных вершин как B. Исток находится в множестве A, а сток — в множестве B.

Рассмотрим все трубы, которые начинаются в одной из вершин A, а заканчиваются в одной из вершин B. Через каждую из этих труб течёт максимально возможный поток, причём жидкость течёт в направлении от A к B, иначе бы мы смогли попасть из некоторой вершины из A в некоторую вершину из B по не заполненной до конца трубе.

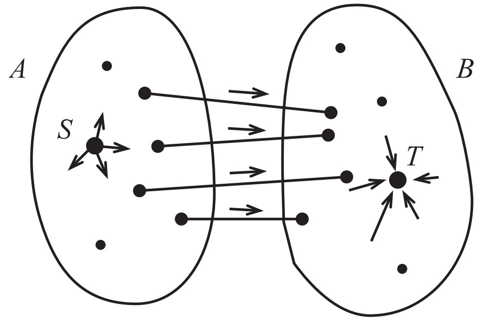

Рис. 3

Суммарный поток, идущий по этим трубам, в точности равен их суммарной проводимости и текущему потоку из истока к стоку. Вся жидкость, которая появляется в истоке S, растекается по вершинам множества A, затем без потерь перетекает по указанным трубам в множество B и в конечном итоге попадает в сток T.

О дружбе и трубах

Покажем, что задача про максимальное паросочетание является частным случаем задачи про максимальный поток.

Рис. 4

Действительно, возьмём наш двудольный граф, вершины которого есть мальчики и девочки, а рёбра обозначают дружбу. Пусть эти рёбра есть трубы, через которые может течь ровно одна единица жидкости от мальчика к девочке, либо не течь ничего. Соединим каждого мальчика трубой с некоторой специальной вершиной S, которая будет истоком, а девочек со стоком T. Пропускная способность этих труб такая же: поток равен либо 0, либо 1.

Тогда максимальный поток от S к T в точности равен максимальному числу пар. По трубе между мальчиком и девочкой течёт жидкость тогда, когда мы выбрали их как пару. В девочку не может втекать больше одной единицы жидкости, так как от неё к стоку идет труба, чья пропускная способность меньше, либо равна 1.

Задача про коробки

На сайте El Judge[1] есть описанная нами задача про замощение клеточной фигуры доминошками (задача №068). Вы можете попробовать написать программу, которая решает поставленную задачу, и послать код на проверку.

Кроме того, на El Judge есть ещё одна интересная задача, которая нетривиальным образом сводится к задаче о максимальном паросочетании.

Коробки. У Пети накопилось много пустых картонных коробок. Они ему ещё сослужат службу в будущем, но сейчас он хотел бы от них избавиться – сдать в камеру хранения. В камере хранения за каждое место берут деньги. Помогите Пете уложить коробки друг в друга так, чтобы плата за их хранения была минимальна. Известно, что никакие две коробки не поместятся ни в какую другую, если их поставить рядом. Коробка со сторонами x_1, x_2, x_3, $x_1 \geqslant x_2 \geqslant x_3$, помещается в коробку со сторонами y_1, y_2, y_3, $y_1 \geqslant y_2 \geqslant y_3$, если

$$x_1 < y_1, x_2 < y_2, x_3 < y_3.$$

Входные данные содержат информацию о числе коробок и их размерах. Длины сторон коробок — три натуральных числа, меньших 2000000. На выходе нужно получить минимальное число коробок, в которые все остальные можно поместить.

[1] El Judge (`http://acm.mipt.ru/judge`) — постоянно действующее соревнование по программированию среди школьников и студентов.

Однажды на лекции

– Ну вы методичку-то читали?
– Да, но в interlaced режиме.

...он смотрел на меня своим дружественным интерфейсом...

Продаем жидкокристаллические мониторы. Опт, мелкий опт. Фасовка — 3, 5, 10 литров.

Однажды в студеную зимнюю пору я вышел из Windows. Был NORTON и DOS...

Если windows есть, значит это кому-то нужно...

Запуская windows, вы рискуете быть повешенным...