Алгоритмы и лабиринты

А.Я.Белов-Канель, И.А.Гажур, И.А.Иванов-Погодаев, А.С.Малистов

Цикл А. Это вводный цикл. Его задачи являются вспомогательными и будут сняты после промежуточного финиша.

В первом (вводном) цикле задач изучается стандартный конечный автомат: робот, перемещающийся по клеточной плоскости. Робот — это механизм, производящий заранее определенный набор действий по определенной программе. Робот может делать следующее:

1. Перемещаться вперед (в соседнюю клетку);
2. Поворачиваться на месте на 90 градусов;
3. Ставить флажок в клетку, где находится (если флажок у него есть);
4. Проверять наличие флажка в своей клетке.
5. Брать флажок из своей клетки.

Наличие у робота флажков (и их количество) оговаривается заранее. Программа состоит из пронумерованного упорядоченного набора инструкций для робота, причем некоторые инструкции могут заключаться в переходе к другой инструкции. Мы будем ставить для робота задачи обхода, состоящие в том, чтобы обойти, то есть побывать в каждой клетке некоторой области. Сначала выясним некоторые возможности робота.

Пример. Докажем, что робот с двумя флажками может обойти ленту — бесконечную полосу с шириной в 1 клетку. Пусть в начале робот находится в клетке с номером 0 с двумя флажками. Построим программу робота.

▷1. Поставить флажок.
▷2. Перейти в клетку 1.
▷3. Поставить флажок.

Теперь нужно организовать «челночное» движение робота между флажками, с попутным увеличением расстояния между ними. То есть, середина отрезка между флажками остается на месте, а флажки робот «расталкивает» в разные стороны. Этого добиваемся с помощью следующей программы:

▷4. Повернуться на месте 2 раза. (разворачиваемся на 180 градусов)
▷5. Пройти вперед.
▷6. Проверить наличие флажка.
▷7. Если флажка нет – перейти к 5.
▷8. Взять флажок и перейти на одну клетку вперед.
▷9. Положить флажок и перейти к 4.

Легко видеть, что робот побывает в каждой клетке ленты.

Интересен вопрос, может ли робот без флажков обойти ленту? Ответ – нет. Чтобы доказать это, необходимо формализовать конструкцию робота. Ясно, что действие, которое робот выполнит в следующий момент, полностью зависит от номера инструкции, которую должен выполнить в данный момент робот и от наличия-отсутствия в клетке флажка. Назовем эти факторы *внутренним состоянием робота*. Ясно, что у робота конечное число возможных внутренних состояний. Выполнив какое-либо действие, робот, вообще говоря, его меняет. По принципу Дирихле, через какое-то время внутреннее состояние робота повторится. Пусть, прошло t секунд и, по сравнению с первым моментом, когда у него было такое состояние, робот сдвинулся вправо на расстояние a. Отметим, что прошло конечное время, и поэтому существует клетка K, левее начальной, где робот еще не был. Поскольку состояние робота такое же, как t секунд назад, еще через t секунд робот сдвинется еще на a вправо, опять не посетив эту клетку. Легко видеть, что далее робот будет действовать, периодически сдвигаясь вправо, и клетку K он не посетит никогда.

А0. *Докажите, что робот с одним флажком ленту обойти не сможет.*

Указание. Рассмотрите отдельно случаи: 1) когда робот не удаляется от флажка на расстояние большее некоторого `N`; и 2) когда робот отходит от флажка сколь угодно далеко.

А1. *Докажите, что робот с* 4 *флажками может обойти плоскость.*

А2. *Докажите, что робот с* 3 *флажками может обойти плоскость. Может ли робот с* 3 *флажками обойти трехмерное пространство?*

А3. *Пусть некоторые границы клеток непроходимы для робота. Робот может видеть барьеры между клетками (проверять на их наличие любую из сторон клетки, где он находится). Пусть непроходимая для робота линия делит плоскость на полуплоскости. Докажите, что робот с* 1 *флажком может обойти полуплоскость.*

А4. *Докажите, что робот с* 1 *флажком не может обойти плоскость с четырьмя разрезами (рисунок 1), а с двумя флажками — может.*

А5. Пусть робот ходит по вершинам графа (возможно, бесконечного).

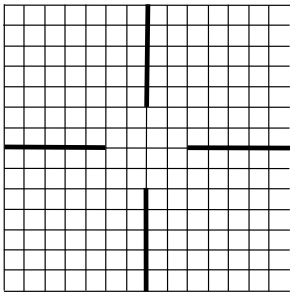

Рис. 1: Четыре разреза на бесконечной плоскости

a) *Докажите, что робот с* 1 *флажком не может обойти бесконечное дерево (граф без циклов), каждая вершина которого имеет степень* 3.

b) *Докажите, что робот с* 2 *флажками не может обойти дерево (граф без циклов), каждая вершина которого имеет степень* 3.

A6. *Докажите, что робот с* 2 *флажками не может обойти плоскость.*

Указание. Используйте идеи, применяющиеся при доказательстве в A0, A4.

Пусть робот без флажков движется в первой четверти плоскости. Границы (оси координат — непроходимые стенки и робот может их обнаруживать.

A7. *Пусть в начале робот находится в клетке с координатами* $(2^n, 0)$. *Составьте универсальную по* n *(независящую от* n*) программу, переводящую робота в клетку* $(3^n, 0)$ *с последующей остановкой в ней.*

Указанный в этой задаче переход будем называть *переходом от* 2^n *к* 3^n.

A8. *По аналогии с предыдущей задачей, осуществите следующие переходы:*

a) от 2^n *к* 6^n;

b) от 2^n *к* 2^{2^n};

c) от $2^n \cdot 3^m$ *к* $2^n \cdot 3^m \cdot 5^{mn}$;

d) от $2^n \cdot 3^m$ *к* $2^n \cdot 3^m \cdot 5^{m+n}$;

e) от 2^n *к* $2^{\lfloor\sqrt{n}\rfloor}$;

f) от 2^n *к* 2^{n^2};

g) от 2^n *к* 2^{k_n}, где k_n есть n-ая цифра десятичного разложения $\sqrt{2}$;

h) от 2^n *к* 2^{k_n}, где k_n есть n-ая цифра десятичного разложения π?

A9. *Докажите, что робот с* 3 *флажками может обойти* n*-мерное клеточное пространство.*

A10. *Докажите, что робот с* 3 *флажками может обойти бесконечное дерево, каждая вершина которого имеет степень* 3.

Алгоритмы крестьянина.

Есть *мир* – ограниченное подмножество клеточной плоскости, неизвестных заранее размеров. Каждая клетка плоскости относится к одному из следующих типов:

1. Скала. Клетка, недоступная для посещения.

2. Земля. Клетка доступна для посещения.

3. Озеро. Клетки недоступны для посещения. В таких клетках можно ловить рыбу, находясь в соседней по стороне клетке. С каждой клеткой-озером связаны параметры `nibble` — клев (количество рыбы, которое вылавливается из озера за 1 раз, при применении операции лова рыбы) и `fish` — количество оставшейся в озере рыбы.

4. Дом. Уникальная клетка в мире, доступна для посещения, обычно крестьянин находится в начале в ней. Часто требуется отнести пойманную рыбу в дом.

Крестьянин перемещается по миру, переходя из клетки в клетку. Находясь в клетке, он смотрит в одну из четырех сторон (направление движения).

Областью действия крестьянина будем называть множество из двух элементов: клетку, где крестьянин находится, и клетку перед ним (соседнюю по направлению движения).

Крестьянину доступна информация о клетках из области действия. Информация о других клетках мира ему недоступна. У крестьянина есть ограниченное условиями задачи количество флажков. Он может оставлять флажки в клетках и таким образом исследовать мир. Кроме того, находясь в соседней с озером клетке, крестьянин может ловить рыбу. У него может быть при себе некоторое количество рыбы, не превышающее 10 кг.

В клетках мира могут находится разные объекты (флаги, рыба). С объектами может быть связано несколько переменных, информация о которых доступна крестьянину. Переменные, связанные с флагами, крестьянин может менять, в том числе можно создавать новые переменные и изменять их значения.

Крестьянин обладает ограниченной памятью. Он может создавать собственные переменные и изменять их значения. Значения собственных переменных доступны крестьянину независимо от клетки, где он находится.

Формально, крестьянин может совершать следующие действия:

1. [Go] Переместиться вперед.
2. [Rotate] Повернуться на месте в любую из четырех сторон.
3. [Read] Получить информацию о переменных, связанных с объектами в клетках области действия.
4. [SetFlag] Поставить флажок в клетку области действия или убрать флажок из нее. Также можно создавать переменные, связанные с флагом, и изменять их значения.
5. [CheckFlag] Проверить наличие флажка в области действия.
6. [Fishing] Ловить рыбу в клетке перед ним.
7. [GetPutFish] Оставлять и забирать рыбу в области действия.
8. [Write] Создавать собственную переменную и изменять ее.
9. [Math] Проводить арифметические и логические операции над переменными. В частности, можно проверить равенство некоторых переменных.

Течение времени. Время течет дискретно, по тактам. Если не указано иного, выполнение Go, Fishing и GetPutFish прерывают текущий такт времени (происходит переход на следующий такт). Остальные операции Rotate, Read, SetFlag, CheckFlag, Write, Math не занимают времени, но если в текущем такте выполняется 500 таких операций, такт прерывается. Кроме того, выполнение SetFlag второй раз за такт прерывает текущий такт.

В задачах ниже требуется предъявить алгоритм крестьянина для их решения. Алгоритмы (в виде блок-схем) можно составлять и запускать в специальной программе «VillagerLife», доступ к которой вы получите.

В каждой задаче существует предельное допустимое время работы алгоритма. Размер карты заранее неизвестен. Если не сказано иное, крестьянин умеет определять край карты (если находится на краю). Под обходом карты мы понимаем посещение каждой ее клетки.

Цикл B. В этом цикле запрещается использовать собственные переменные.

B1. *С помощью четырех флагов обойти конечное прямоугольное поле без скал и озер, при условии, что крестьянин не умеет определять, что находится на краю.*

B2. *На карте без скал и озер стоят два флага: один в клетке с координатами $(x, 0)$, второй в клетке с координатами $(0, y)$. Третий флаг находится у крестьянина, который стартует в начале координат. Требуется установить любой флаг в клетку с координатами $(x + y, 0)$. x, y – положительные числа.*

B3. *Пусть теперь могут быть скалы и озера. Обойти карту с произвольным числом флажков (поставить флаг в каждую доступную клетку).*

B4. *Решить задачу B3 за $2n$ тактов, где n — число доступных клеток.*

Цикл C. В этом цикле разрешено использовать собственные переменные.

C1. *Найти число доступных клеток и записать его в переменную* `SPACE`. *Потратить на это не более $2n$ тактов, где n- число доступных клеток.*

C2. *Найти число клеток-озер, доступных со стартовой позиции и записать его в переменную* `WATER`.

C3. *Найти кратчайшее расстояние из начального положения до флага, в котором переменная* `TARGET=THIS`. *Записать ответ в переменную* `DIST`.

C4. *Найти клетку-озеро с наибольшим клевом. Записать координаты клетки в переменные* `NIBBLE_X` *и* `NIBBLE_Y`, *а само значение клева в переменную* `NIBBLE`.

C5. *Для каждой клетки-озера определим параметр полезности равный максимальному удельному количеству рыбы, которое можно принести домой (отношение максимального количества рыбы, которое можно взять в озере, к минимальному количеству тактов, требующихся для ловли и доставки домой). Найти клетку-озеро с наибольшей полезностью. Записать координаты клетки в переменные* `USEFULNESS_X` *и* `USEFULNESS_Y`, *а само значение полезности в переменную* `USEFULNESS`.

C6. *Накопить* `n` *кг рыбы в доме. Значение* `n` *записано в переменной* `FishReq`, *связанной с домом.*

C7. *На карте есть дом с другим крестьянином, который тоже умеет ловить рыбу (но не умеет воровать ее из нашего дома). Требуется накопить* `n` *кг рыбы в доме. Значение* `n` *записано в переменной* `FishReq`, *связанной с домом.*

C8. *На карте есть дом с другим крестьянином, который тоже умеет ловить и воровать рыбу. Требуется накопить* `n` *кг рыбы в доме. Значение* `n` *записано в переменной* `FishReq`, *связанной с домом.*