Вычислительная сложность настольных и компьютерных игр

Федорович Арсений

Физтех-школа прикладной математики и информатики
МФТИ

15 апреля 2025 г.

Под руководством Мусатова Д. В.
МФТИ

Содержание

Введение

Вычислительная сложность игр описывает их принадлежность к каким-либо сложностным классам, а также их полноту в этих классах при растущем размере игрового поля. Фиксированный размер поля нас в данном случае не интересует, потому что в таком случае решение вычисляется за константное время. Основной фокус будет на доказательстве полноты в том или ином сложностном классе.

Существующие результаты

В книге [Hearn and Demaine(2009)] авторы привели набор универсальных графовых игр для различных видов настольных или компьютерных игр (с 0, 1, 2 и большим числом игроков) и показали их полноту в соответствующих классах.

Также полнота была показана для планарных универсальных игр. Это позволяет использовать их для моделирования того, что происходит в настольной или компьютерной игре. При сведении формул к играм не всегда видна взаимосвязь между формулами и конструкциями игры.

Во второй части книги авторы привели новые результаты, определив сложностные классы и показав полноту в них для таких игр, как **TipOver**, **Hitori**, **sliding-block puzzles** и др.

Логика ограничений

Определение

Граф ограничений — неориентированный граф, вершины которого имеют порог — значение от 0 до 2, а рёбра имеют вес 1 (красного цвета) или 2 (синего цвета).

Замечание

На самом деле, граф будет иметь направления рёбер, которые могут меняться в зависимости от определённых условий. Смену направлений рёбер будем называть переключениями. Считаем порог равным 2.

Определение

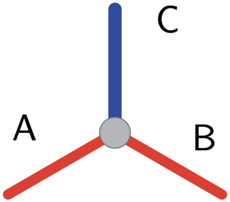
Условие корректности — суммарный вес входящих рёбер больше либо равен порогу вершины.

Условие корректности накладывает определённые ограничения на то, каким граф может быть, поскольку переключать рёбра разрешено только в соответствии с ним.

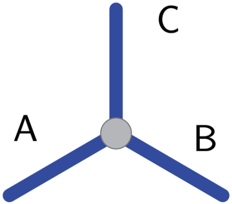
Гаджеты AND и OR

Цель — цепочкой переключений рёбер добиться переключения конкретного целевого ребра. При этом, например, в играх с двумя игроками у каждого игрока могут быть свои целевые ребра, которые они должны переключить, чтобы победить.

Примеры гаджетов:



(a) Гаджет AND: ребро C может быть направлено из вершины тогда и только тогда, когда рёбра A и B направлены в неё.



(b) Гаджет OR: ребро C может быть направлено из вершины тогда и только тогда, когда хотя бы одно из рёбер A или B направлено в неё.

Ограниченные и неограниченные игры

В дальнейшем сфокусируемся лишь на играх с одним игроком. Однако, вне зависимости от числа игроков, все игры подразделяются на два вида — **ограниченные** и **неограниченные**. В ограниченных играх можно сделать полиномиальную оценку на число ходов. Обычно это предполагает наличие некоторого ресурса, который расходуется в процессе игры. В неограниченных, напротив, нет ограничения на количество ходов. Часто в таких играх можно отменять уже сделанные ходы.

Примеры

- Игра **Судоку** является **ограниченной**, так как с каждым ходом игровое поле заполняется, ходы отменять невозможно.
- Игра **TipOver**, в которой игрок должен ронять блоки в правильном порядке, поднять которые уже невозможно, является **ограниченной**.
- Игра **Пятнашки** является **неограниченной**, так как костяшку можно перемещать влево-вправо произвольное количество раз.

Ограниченные и неограниченные игры

В ограниченном случае недетерминированно вычисленное решение можно проверить за полиномиальное время, что указывает на принадлежность таких игр к классу NP. Неограниченные же игры принадлежат NPSPACE, что равно PSPACE по теореме Сэвича.

Авторы также доказали полноту неограниченных и ограниченных игр в этих классах, что позволяет сводить их к настольным и компьютерным играм. Более формально:

Определение

Ограниченная недетерминированная логика ограничений (BNCL) - это задача

$$\{(G, e) \mid \text{ можно последовательным переключением рёбер } G \text{ переключить } e\},$$

при этом переключать рёбра можно не больше одного раза

Ограниченные и неограниченные игры

Определение

Неограниченная недетерминированная логика ограничений (UNCL) - это задача

$$\{(G, e) \mid \text{ можно последовательным переключением рёбер } G \text{ переключить } e\},$$

при этом переключать рёбра можно многократно

Теорема 1

UNCL PSPACE-полна даже для планарных графов, у которых встречаются только вершины вида AND и OR.

План работы

Планируется показать, что можно свести **UNCL** к выбранной игре с одним игроком. Необходимо показать принадлежность игры к классу PSPACE, а затем построить гаджеты AND и OR, используя игровые конструкции. Тем самым будет доказана полнота в классе PSPACE.

Описание выбранной игры

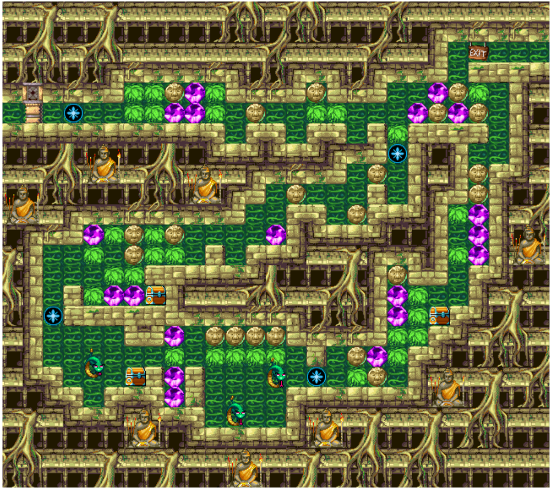
Diamond Rush — 2D-платформер, в котором игроку предстоит перемещаться по пещерам, полным различных опасностей. Нас будут интересовать такие элементы игры, как круглые камни, нажимные плиты и двери, которые открываются при активации связанных с ними плит. Игрок может толкать камни до тех пор, пока те не упрутся в стену. В игре действует гравитация, поэтому их можно столкнуть вниз. Нажимная плита активируется, только когда на ней находится камень. В активном состоянии она удерживает связанную с ней дверь открытой. Как только камень смещается с плиты, дверь немедленно закрывается.

Цель игры — добраться до финиша.

Будем рассматривать игровое поле как квадрат $n \times n$, состоящий из клеток, каждая из которых представляет собой определенный игровой объект.

Описание выбранной игры

Игровое поле выглядит так:



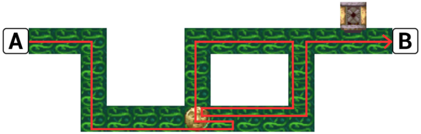
Гаджет One-Way

Будем показывать PSPACE-полноту. Для этого необходимо показать сводимость **UNCL**$\leq_p$**Diamond Rush**. По **UNCL**-графу нужно построить **Diamond Rush**-игру, которая решается тогда, когда можно обратить ребро в **UNCL**-графе. Согласно теореме 1, будем считать, что он планарный и состоит из гаджетов AND и OR. Поэтому нам нужно показать, как построить данные гаджеты в игре, а также как их соединять вместе. Однако для начала нам понадобится вспомогательный гаджет One-Way, который может быть пройден лишь в одном направлении. Мы будем подсоединять данный гаджет к выходным вершинам гаджетов AND и OR.

Рис. 2: Гаджет One-Way

Гаджет One-Way

Гаджет One-Way работает следующим образом: мы считаем, что камень стоит на нажимной плите и удерживает дверь, расположенную рядом с B, открытой. Квадрат, расположенный над проходом у B, обозначает панель, из которой вниз выдвигается дверь. Сейчас плита зажата, поэтому дверь находится в этой панели и проход открыт. Чтобы попасть из A в B, нужно подвинуть камень вправо, чтобы пройти наверх. После этого необходимо вернуть камень в прежнее состояние, чтобы открыть дверь. Таким образом, после прохода из A в B гаджет восстанавливается до прежнего состояния. Из B в A попасть мы не можем, так как при проталкивании камня влево он упрется в стену. Схема обхода гаджета изображена ниже:



Гаджет OR

Гаджет OR работает следующим образом: добраться до C можно, если попасть либо в A, либо в B.

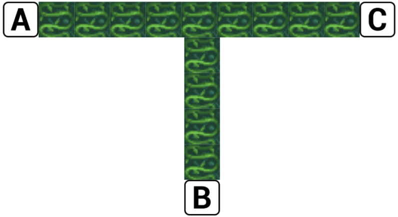


Рис. 3: Гаджет OR

Гаджет AND

Гаджет AND выглядит следующим образом. Нижняя пара камней зажимает нажимные плиты (не отображённые на рисунке 4), которые активируют верхние двери. Видимые плиты, находящиеся рядом с нижними камнями, отвечают за нижние двери. Изначально они закрыты, так как соответствующие плиты не зажаты.

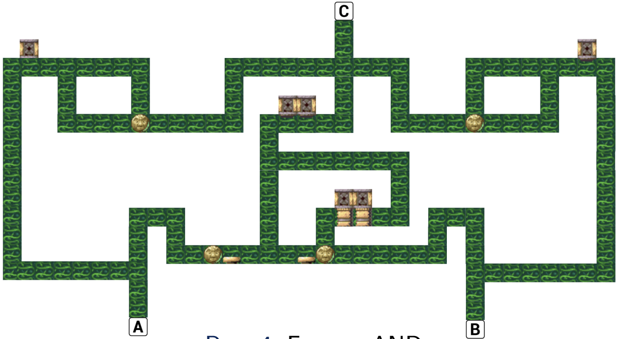
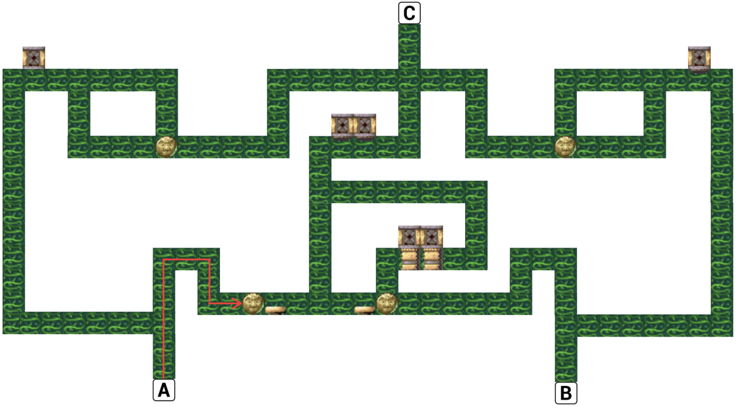


Рис. 4: Гаджет AND

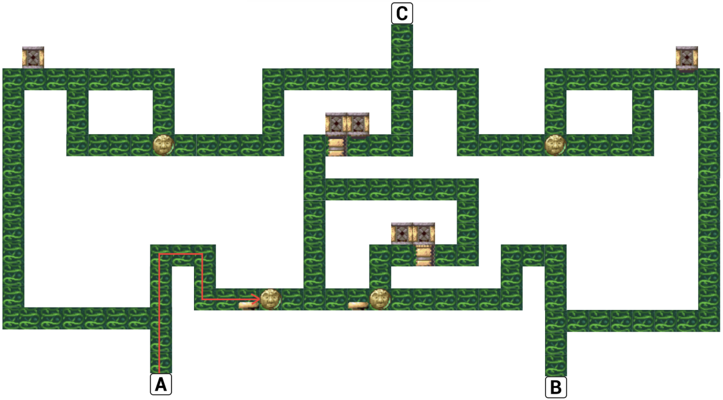
Гаджет AND

Покажем, как он работает. Сначала сдвигаем левый нижний камень вправо на один шаг. Изначально он стоит на нажимной плите, активирующей левую дверь сверху. После сдвига вправо он активирует другую (видимую) плиту, открывающую левую дверь снизу.



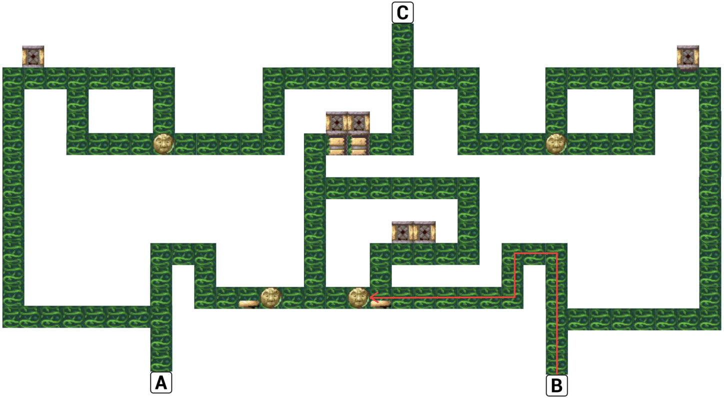
Гаджет AND

После сдвига одна из дверей сверху открылась, а одна из дверей снизу — закрылась:



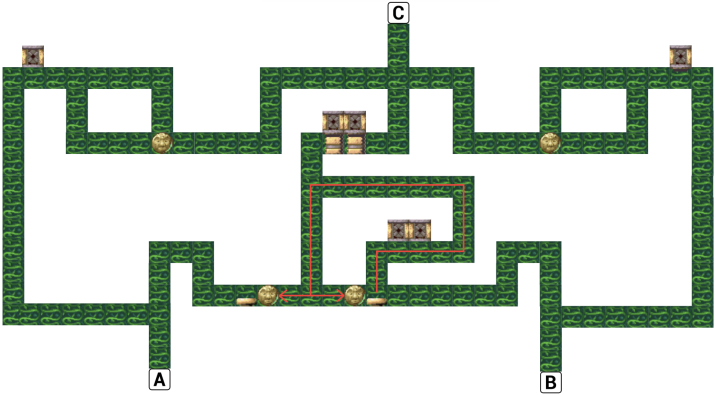
Гаджет AND

Затем сдвигаем правый нижний камень влево на один шаг. Изначально он стоит на нажимной плите, активирующей правую дверь сверху. После сдвига влево он активирует другую (видимую) плиту, открывающую правую дверь снизу. Обе нижние двери теперь открыты.



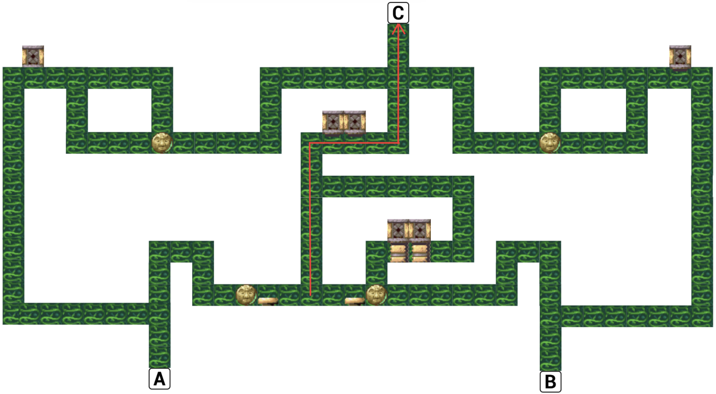
Гаджет AND

Однако верхние двери закрыты, и попасть в C можно только после возврата камней на свои прежние места.



Гаджет AND

Наконец, верхние двери открываются и можно попасть в C.



Вспомогательные леммы

Лемма

Конструкция, изображенная на рисунке 3 удовлетворяет тем же ограничениям, что и гаджет OR, где A, B - входящие ребра, а C - исходящее ребро.

Доказательство.

Мы можем попасть в C тогда и только тогда, когда можем попасть либо в A, либо в B. One-Way гаджет, подсоединенный к вершине C, исключает возможность попадания в неё иными способами. □

Вспомогательные леммы

Лемма

Конструкция, изображенная на рисунке 4 удовлетворяет тем же ограничениям, что и гаджет AND, где A, B - входящие ребра, а C - исходящее ребро.

Доказательство.

Покажем, что можем попасть в C тогда и только тогда, когда можем попасть и в A, и в B. Описанным выше способом мы можем достичь C. При этом, если мы пойдем только в A или только в B, то заблокируем проходы и не сможем достичь C. Левое и правое ответвления гаджета с соответствующими верхними камнями являются One-Way гаджетами из C в A и из C в B. Это позволяет, если мы попали в C из вершин A и B, затем вернуться из C обратно в обе эти вершины, ведь в **UNCL** разрешены множественные переключения ребер. При этом данные ответвления не позволяют добраться в C из какой-то одной вершины. One-Way гаджет, подсоединенный к вершине C, исключает возможность попадания в неё иными способами. □

PSPACE-полнота выбранной игры

Теорема 2

Diamond Rush PSPACE-полна.

Доказательство.

Сводим **UNCL**$\leq_p$**Diamond Rush**: по планарному **UNCL**-графу, состоящему из гаджетов OR и AND, строим игру. Гаджеты соединяются последовательностью клеток свободного пространства, по которым игрок может перемещаться. Далее используем теорему 1. **Diamond Rush** принадлежит PSPACE, так как потенциальное решение хоть может и не быть проверено за полиномиальное время (если, например, перемещаем камень влево-вправо больше, чем полиномиальное количество раз), но проверяется с использованием полиномиальной памяти от размера поля n. □

Список литературы

Robert A. Hearn and Erik D. Demaine.
Games, Puzzles, and Computation.
A K Peters, 2009.
ISBN 978-1-56881-322-6.
URL `https://doi.org/10.1201/b10581`.