Санкт-Петербургский государственный университет

Саакян Артур Суренович

Выпускная квалификационная работа

Алгоритмическая сложность terge gate на 2D-карте

Уровень образования: бакалавриат
Направление *02.03.01 "Математика и компьютерные науки"*
Основная образовательная программа *СВ.5152.2019 "Математика, алгоритмы и анализ данных"*

Научный руководитель:
доцент, Факультет математики
и компьютерных наук СПбГУ,
Авдюшенко Александр Юрьевич

Рецензент:
Главный специалист,
Общество с ограниченной ответственностью
«Газпромнефть
Информационно-Технологический оператор»,
Евменов Владимир Андреевич

Санкт-Петербург
2023

Содержание

1. Введение

В двадцать первом веке появилась игровая индустрия, ориентированная на массовую аудиторию “казуальных” пользователей [KANP07]. С развитием технологий и распространением игровых устройств стало возможным создавать более простые и доступные игры, которые могут быть быстро запущены и которые будут занимать пользователя на короткое время. Казуальные игры также обычно имеют более низкий порог входа и могут привлечь широкую аудиторию, поскольку основаны на очень простых правилах.

Большое количество мобильных игр основаны на общей merge механике, среди них так называемые Match-3 (три в ряд), например, Bejeweled, AniPang, Aurora Feint, Beghouled & Beghouled Twist, Candy Crush Saga, Diamond Twister, Gweled, Goober’s Lab, Jewel Quest, King Boo’s Puzzle Attack, Magic Duel, “Puzzle Quest: Challenge of the Warlords”, Sutek’s Tomb, Switch, The Treasures of Montezuma, Zoo Keeper и многие другие. На рисунке 1 (взято из [GLN14]) изображена хронология появления некоторых merge-игр.

Несмотря на простые правила, достаточно часто сами игры оказываются NP-полными, если рассматривать их обобщения на большие карты с сохранением игровой механики. В частности, одни из самых популярных казуальных игр современности — 2048 и Candy Crush Saga являются NP-полными, если сделать размеры игровых карт параметром.

Связанные работы. Связанные работы в данной области включают исследования сложности различных игровых механик и алгоритмов для их решения. В работе [GLN14] авторы исследуют сложность Match-3 игр, анализируя количество возможных комбинаций и сравнивая их с другими известными задачами, такими как задача о рюкзаке. Результаты показали, что большинство Match-3 игр являются NP-полными.

В статье [Hay16] авторы сводят обобщенную версию игры Судоку к поиску гамильтонова цикла в графе. Также было проведено исследование игры 2048, которая стала популярной после ее появления в 2014 году. В статье [AAD15, AAD16] авторы доказали NP-полноту для классической версии игры и предложили эффективный алгоритм для нахождения решения. В статье [KPS08] можно найти обширный обзор множества разнообразных NP-полных пазлов, включая карточную игру Solitaire и популярную игру из 80-х Sokoban.

Обзор результатов. Данная работа представляет собой захватывающее путешествие в мир алгоритмической сложности, основанное на увлекательной компьютерной игре Merge War. Мы предлагаем доказательство того, что упрощенная версия этой игры является NP-трудной задачей, открывающей перед нами вселенную возможностей для создания эффективных алгоритмов.

С нашими новыми знаниями мы получим уникальную возможность разработать точные экспоненциальные, параметризованные алгоритмы для решения данной задачи, а также в некоторых случаях решить ее за полиномиальное время. Это открывает необычайно широкие перспективы для применения методов и техник, которые помогут нам решать самые разнообразные задачи в различных областях.

Поэтому мы приглашаем вас присоединиться к нашему увлекательному путеше-

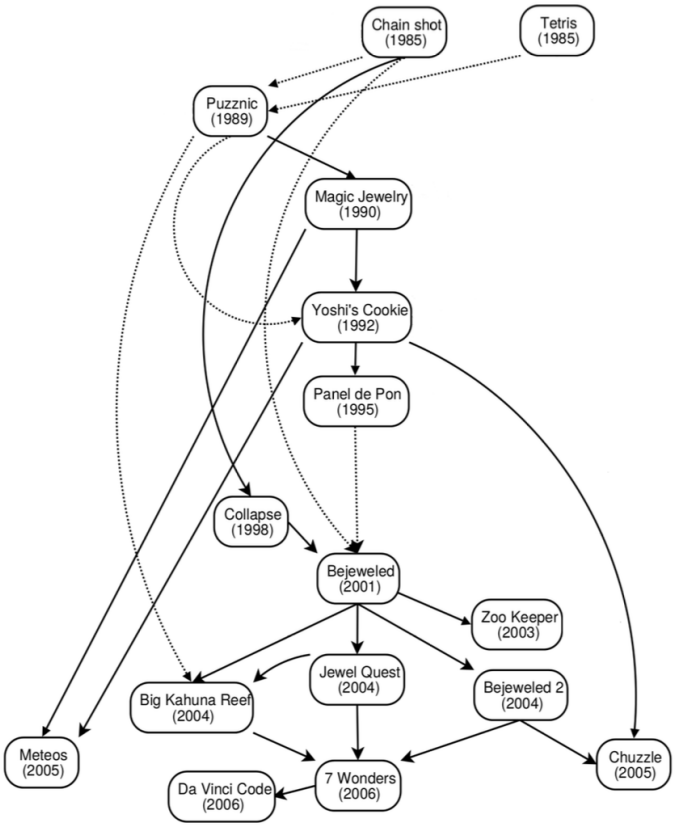


Рис. 1. Хронология

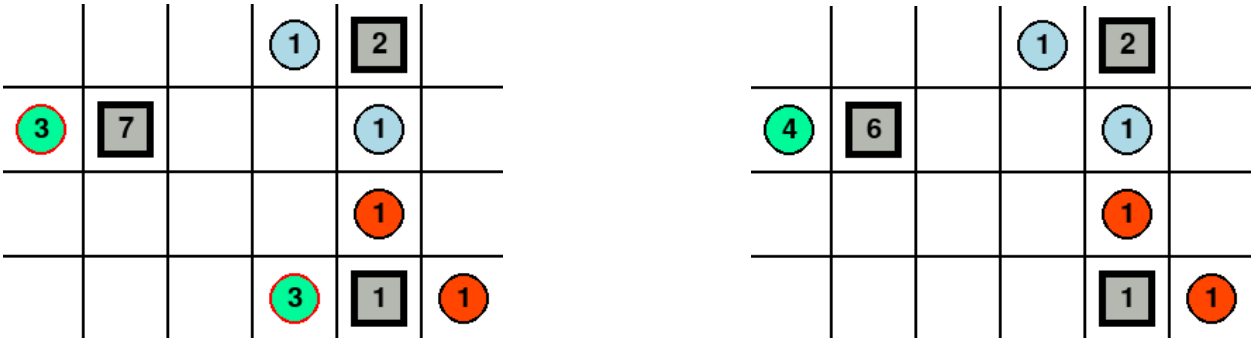
ствию по миру алгоритмической сложности и насладиться процессом изучения и разработки новых алгоритмов и методов для решения самых интересных и актуальных задач!

1.1. Игра Merge War

Merge War (ссылка) — однопользовательская мобильная игра, ключевой механикой в этой игре является merge. Игрок проходит уровни, сражаясь с ботами на двумерной карте. Одной из стратегий в игре является призыв как можно большего количество героев с целью увеличения своей силы. Так как свободных клеток на карте ограниченное число, возникает задача разрушения камней для освобождения свободного места.

Таким образом, по мотивам данной игры возникает следующая задача. Игра происходит на двумерной карте, разбитой на клетки. В некоторых клетках стоят *камни*, в некоторых клетках стоят *герои*. Каждый герой характеризуется типом и уровнем — натуральным числом, изначально у всех героев уровень 1. Каждый камень характеризуется целочисленными очками здоровья hp. Цель игры — уничтожить как можно больше камней.

Для этого можно произвольное количество раз выполнять следующую операцию — выбрать двух героев одинакового уровня, и объединить их в одного героя уровнем на 1 выше. Новый герой помещается в любую из клеток, где стояли объединенные в него герои, и наносит урон 1 соседним по стороне камням.



(a) До объединения (b) После объединения

Рис. 2. Объединение (merge) героев

На рисунке 2 герои изображены в виде кругов разных цветов. Каждый цвет обозначает свой тип героев, а число внутри круга – это их уровень. Камни нарисованы в виде квадратов, а число внутри каждого квадрата показывает количество очков здоровья hp. В конкретном случае два зеленых героя 3 уровня объединились в нового зеленого героя 4 уровня, и нанесли урон камню, уменьшив очки здоровья камня с 7 до 6.

2. Определения и постановка задачи

Для формализации игры введем вспомогательные определения, которые определят все типы объектов в нашей игре.

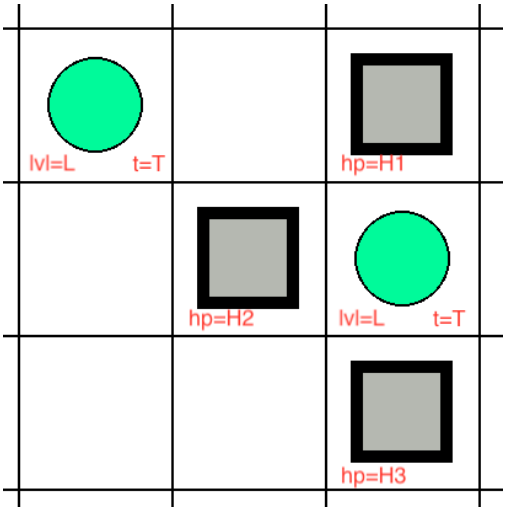
2.1. Основные определения

Определение 1. *Карта* — квадратная таблица $n \times n$, состоящая из n^2 ячеек. Система координат задана на этой таблице таким образом, что левая нижняя клетка имеет координаты $(1, 1)$, а правая верхняя имеет координаты (n, n).

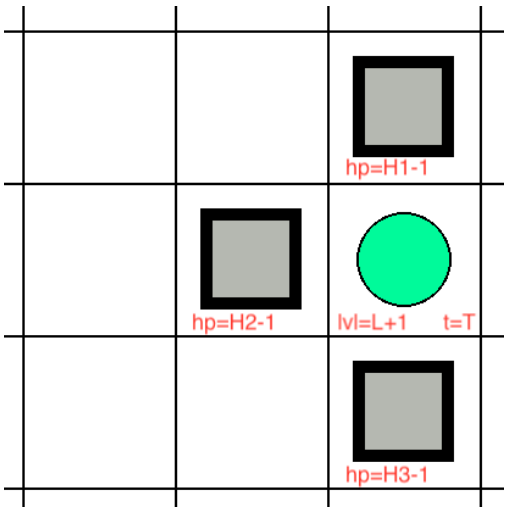
Определение 2. *Камень* — объект, который мы можем расположить в одной из ячеек карты. Камень S характеризуется очками здоровья $S_{hp} \in \mathbb{N}_0$ и координатами на карте $S_{pos} = (S_x, S_y)$.

Определение 3. *Герой* — объект, который мы можем расположить в одной из ячеек карты. Герой H характеризуется своим типом $H_t \in \mathbb{N}$, уровнем $H_{lvl} \in \mathbb{N}$ и координатами на карте $H_{pos} = (H_x, H_y)$.

Определение 4. *Конфигурацией* карты будем называть множество *героев* и *камней*, расположенных на ней.



(a) До merge-операции



(b) После merge-операции

Рис. 3. Применение merge-операции в общем случае

Merge-операция (также *merge*, *объединение*) является центральной в нашей игре, и определяется следующим образом:

1. Выбирается два героя A и B одного типа ($A_t = B_t$) и одного уровня ($A_{lvl} = B_{lvl}$).
2. Герой A исчезает с карты.
3. Уровень героя B увеличивается на 1: $B_{lvl} := B_{lvl} + 1$
4. Для всех камней D, которые являются соседними по стороне с героем B, уровень здоровья уменьшается на единицу: $D_{hp} = D_{hp} - 1$.
5. Все камни, уровень здоровья D_{hp} которых стал равен нулю, "уничтожаются"и исчезают с карты, клетка становится свободной.

Часто удобно говорить *смерджили двух героев* вместо *провели merge-операцию с двумя героями*.

Определение 5. *Начальной конфигурацией* карты будем называть такую конфигурацию, каждый герой H которой имеет первый уровень ($H_{lvl} = 1$).

Теперь, когда мы ввели все термины, мы можем перейти к описанию *игрового процесса*:

1. Есть начальная конфигурация.
2. Игра состоит из последовательного применения merge-операций.
3. Игра заканчивается, когда мы не можем сделать очередную merge-операцию, то есть когда на карте нет двух героев одного типа и уровня. Также мы можем досрочно завершить игру.

2.2. Задача MERGE-GAME и её версии

Теперь мы можем сформулировать вычислительную задачу, которая связана с описанной в предыдущем разделе игрой.

MERGE-GAME
Вход:

- Натуральное число n – размер карты.
- Множество из h героев $H^1, H^2, \ldots, H^h$. Каждый герой находится в клетке H^i_{pos}, имеет свой тип H^i_t и начальный первый уровень.
- Множество из s камней $S^1, S^2, \ldots, S^s$. Каждый камень находится в клетке S^i_{pos} и имеет свой начальный уровень здоровья S^i_{hp}.
- Целое число x — ожидаемое количество уничтоженных камней.

Выход: Существует ли последовательность мерджей, которая уничтожает хотя бы x камней?

Зачастую интересно рассматривать разные версии задачи, чтобы можно было увидеть, какие именно ограничения приводят к NP-трудности. Мы будем рассматривать различные модификации MERGE-GAME, которые обычно задаются новыми ограничениями. Для каждого ограничения было придумано кодовое название:

- **DESTROY-ALL** — $x = s$, то есть мы хотим узнать, можно ли уничтожить все камни.
- **ONE-HP** — для всех камней изначально очки здоровья равны единице.
- **EQUAL-TYPE** — все герои принадлежат одному типу.
- **LIMITED-LEVEL** — максимальный уровень, которого может достигнуть герой, ограничен константой.

Также для некоторых модификаций мы будем придумывать отдельные имена. В частности, достаточно интересная модификация включает в себя ограничения DESTROY-ALL, ONE-HP, EQUAL-TYPE, и называется **MERGE-GAME-SIMPLE**. В следующем разделе мы подробнее рассмотрим эту модификацию, и затем сформулируем эту задачу в более простом виде.

3. Результаты

3.1. Сложность MERGE-GAME

В этом разделе мы будем доказывать, что наша задача принадлежит классу NP и является NP-трудной задачей.

Теорема 1. *Задача **MERGE-GAME** принадлежит классу* NP.

Доказательство. В качестве сертификата мы будем получать последовательность мерджей. Каждый merge задается парой (i, j) — номерами героев, участвующих в объединении. Заметим, что количество мерджей не может быть больше чем $h - 1$ (h — изначальное количество героев). Это так, потому что после каждого объединения количество героев уменьшается на 1. Значит, размер подсказки будет полиномиальным от h. В то же время мы можем легко проверить, какое количество камней уничтожаются с применением этого сертификата — достаточно промоделировать предложенную последовательность действий.

□

Теперь покажем NP-трудность. Для этого мы будем рассматривать "упрощение" изначальной задачи, которая является частным случаем MERGE-GAME и легко к ней сводится.

Задачей **MERGE-GAME-SIMPLE** называется модификация MERGE-GAME, в которой применены ограничения DESTROY-ALL, ONE-HP и EQUAL-TYPE. Иными словами, мы должны уничтожить все камни, у каждого камня изначально единичный hp, и все герои принадлежат одному типу.

Лемма 1. *Пусть есть ограничения DESTROY-ALL, ONE-HP и EQUAL-TYPE.в Тогда в любой последовательности мерджей можно убрать некоторые мерджи так, чтобы множество уничтоженных камней не поменялось, и все мерджи происходили между героями первого уровня.*

Доказательство. Пусть $I = (i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)$ — данная последовательность мерджей. Рассмотрим последовательность I', которая является самой короткой подпоследовательностью I, и уничтожает ровно те камни, что и I. Докажем от противного, что в нем не будет мерджей среди персонажей уровнем выше первого.

Пусть не так, и найдется мердж с персонажами уровня выше первого. Рассмотрим самый правый такой мердж (i_q, j_q), и уберем его. Заметим, что последовательность мерджей все еще будет корректной, ведь герой с номером j_q нигде больше в мерджах не встретится (иначе (i_q, j_q) был бы не самым правым мерджем). При этом ясно, что вокруг героя j_q к q-ому ходу уже не было камней, потмоу что они уничтожились как только камень j_q достиг второго уровня.
Значит, из I' можно убрать мердж (i_q, j_q) без потери уничтоженных камней, и, значит, I' — не самая короткая среди таких подпоследовательностей I. Получили противоречие.

□

Замечание. Заметим, что в данной лемме мы пользовались только ограничением HP-ONE. Данная лемма справедлива и без ограничений DESTROY-ALL и EQUAL-TYPE.

Лемма 2. *Теперь с использованием предыдущей леммы мы можем переформулировать задачу MERGE-GAME-SIMPLE в более лаконичную эквиалентную формулировку:*

- *Даны h героев и s камней на карте $n \times n$.*
- *Нужно выбрать не более $\lfloor \frac{h}{2} \rfloor$ героев так, чтобы рядом (по стороне) с каждым камнем был хотя бы один выбранный герой.*

Доказательство. Действительно, если мы мерджим героев первого уровня, то мы фактически выбираем множество непересекающихся пар (i_q, j_q), и наносим единичный урон по соседям j_q-ого героя. □

Заметим, что в такой формулировке пропадают такие параметры, как уровень героев lvl, тип героев t и очки здоровья камней hp. Таким образом, теперь мы можем определить задачу **MERGE-GAME-SIMPLE**, которая будет выглядеть куда более простым образом. Для удобства в будущем мы будем использовать выражение *герой бьет камень*, которое означает, что герой находится в соседней с камнем клетке.

MERGE-GAME-SIMPLE
Вход:

- Натуральное число n – размер карты.
- Множество из h героев $H^1, H^2, \ldots, H^h$. Каждый герой находится в клетке H^i_{pos}.
- Множество из s камней $S^1, S^2, \ldots, S^s$. Каждый камень находится в клетке S^i_{pos}.

Выход: Можно ли выбрать не более $\lfloor \frac{h}{2} \rfloor$ героев так, чтобы каждый камень бился хотя бы одним выбранным героем?

На Рис. 4 привел пример, в котором из $h = 5$ героев было выбрано $\lfloor \frac{h}{2} \rfloor = 2$ героя, которые бьют все камни.

Докажем NP-трудность нашей задачи, для этого сведем к ней NP-трудную задачу о вершинном покрытии на планарных графах степени не более 3 [Lic82]. Ниже мы приводим определение задачи о вершинном покрытии.

PLANAR VERTEX COVER-DEG3
Вход: Дан граф $G = (V, E)$, и число k. Граф является планарным, и степень каждой его вершины не превосходит 3.
Выход: Существует ли множество $V' \subset V$ размера k такое, что для любого ребра $e \in E$ некоторая вершина $v' \in V'$ является его концом?

Теорема 2. *Задача **MERGE-GAME-SIMPLE** является NP-полной.*

Рис. 4. Пример выбора героев, чтобы они били все камни

Доказательство. Принадлежность показана в теореме 1. Докажем NP-трудность нашей задачи. Для этого сведем PLANAR VERTEX COVER-DEG3 к нашей задаче. Пусть у нас есть граф планарный граф $G = (V, E)$ степени 3, и мы хотим найти в нем вершинное покрытие размера k.

Шаг 1. *Укладка графа на грид.* В этой части мы будем заниматься достаточно естественной вещью – поскольку в MERGE-GAME-SIMPLE все происходит на таблице, достаточно логично положить исходный граф G на целочисленную решетку, структура которой очень похожа на таблицу. В своей работе Kant [Kan92] показал, что любой планарный граф размера n степени 3 можно уложить на целочисленной решетке за линейное время так, чтобы все ребра стали непересекающимися друг с другом ломанными, состоящими только из вертикальных и горизонтальных отрезков, проходящих по целочисленным вертикалям и горизонталям. При этом новое представление графа можно вписать в квадрат со стороной $O(n)$. Вот пример такой укладки графа:

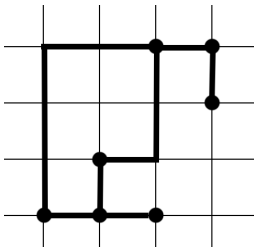


Рис. 5. Пример плоской укладки на целочисленной решетке

Шаг 2. *Удлинняем ребра так, чтобы их длина представлялась в виде* $4l + 2, l \in \mathbb{Z}$. Давайте сначала "участим"сетку, то есть сделаем её в 8 раз более частой (Рис. 6). Ясно, что теперь длина каждого ребра делится на 8 и, значит, представима в виде $4l, l \in \mathbb{Z}$.

Теперь, если мы сможем увеличить длину каждого ребра на 2, то мы добьемся желаемого в этом шаге результата. Сделать это достаточно просто: для каждого ребра возьмем середину его вертикального или горизонтального куска, и добавим 3 маленьких изгиба, которые увеличат длину ребра на 2 (Рис. 7).

Шаг 3. *Расставляем героев и камни на новом графе.* Обычно мы ставили героев и

Рис. 6. Увеличили частоту сетки в 8 раз с сохранением структуры графа.

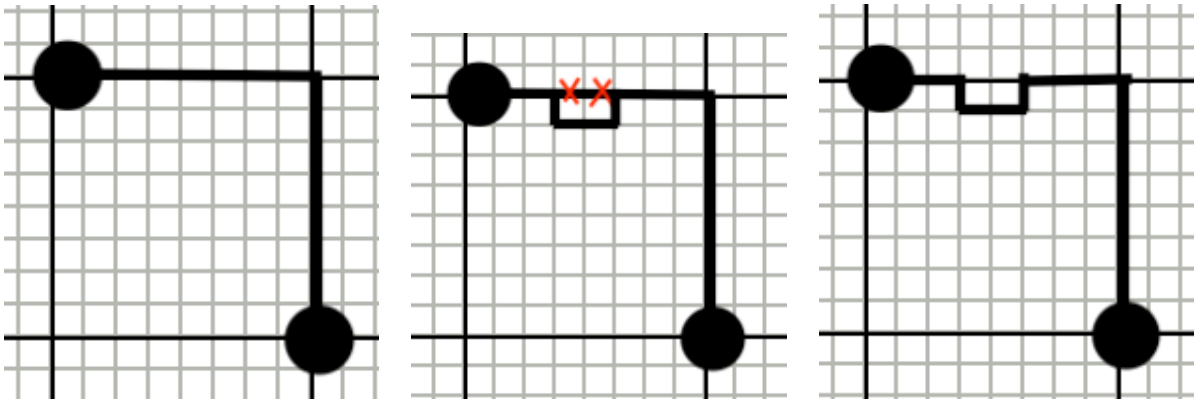


Рис. 7. Удалили часть ребра, и добавили изгибы

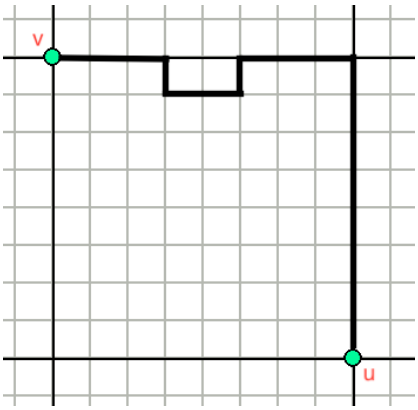
камни на таблице, но ясно, что это абсолютно то же самое, что ставить их в вершинах целочисленной сетки – ведь из каждой вершины сетки линии сетки ведут вниз, вверх, влево, вправо. Первым делом поставим героев в каждую вершину исходного графа в отображении на плоскости. Дальше возьмем каждое ребро (ребра имеют длину вида $4l + 2$), и будем ставить в целочисленных координатах на них новых героев и новые камни, чередуясь. На иллюстрации приведен пример расстановки, квадратами обозначены камни, кругами – герои. В силу построения видно, что герой бьет камень тогда и только тогда, когда между ними есть ребро в получившемся на решетке графе. И действительно, за счет того, что мы участили решетку в 8 раз, все вершины стали далеко друг от друга, теперь между любыми соседними по вертикали или горизонтали героями/камнями есть ребро. При этом новый граф имеет $O(n^2)$ вершин (не больше, чем площадь квадрата, в который вписано изображение графа).

Пусть $4l_i + 2$ – длина i-ого ребра до того, как мы поставили на нем новых героев и камни. Тогда после расстановки строго внутри этого будут стоять $2l_i$ героя и $2l_i + 1$ камень. Также мы ставили героев в исходных вершинах графа, которых ровно n. Таким образом, суммарно на новом графе у нас будет $n + 2 \sum\limits_{e_i \in E} l_i$ героев.

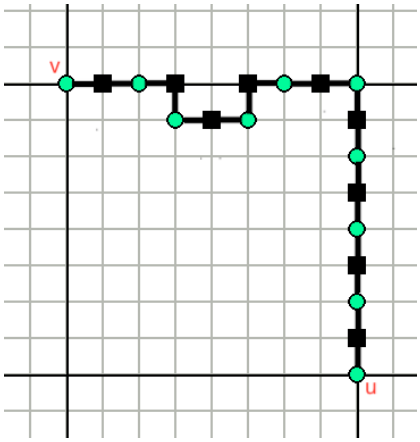
Шаг 4. *Эквиалентность задач.*

Построим вход задачи на полученном в предыдущем шаге графе. Пусть $L = \sum\limits_{e_i \in E} l_i$, k – параметр из PLANAR VERTEX COVER-DEG3. Тогда вопрос будет звучать так: можно ли выбрать не более $k + L$ героев так, чтобы все камни были побиты?

Докажем, что эта задача эквиалентна тому, чтобы найти вершинное покрытие раз-



(a) Ставим героев в вершины исходного графа



(b) Расставляем героев и камни на ребре

Рис. 8. Расстановка героев и камней на ребре (u, v) исходного графа

мера k в изначальном графе G . Рассмотрим i-е ребро (u, v) исходного графа. На рисунке изображено то, каким стало плоское изображение этого ребра на решетке (ребро превратилось в путь). По построению каждый герой на этом пути бьет не более 2 камней, и никакие другие герои не могут бить камни с этого пути. Камней всего $2l_i + 1$, и, значит, нам нужно $\lceil \frac{2l_i+1}{2} \rceil = l_i + 1$ героев, чтобы побить все камни. Более того, легко увидеть, что на внутренности этого пути (то есть без его концов u и v) мы обязаны выбрать как минимум $\lceil \frac{2l_i-1}{2} \rceil = l_i$ героев, чтобы побить все камни на пути (u, v) (независимо от того, выбрали ли мы героя в u или v).

То, что мы побили все камни, эквиалентно тому, что для любого ребра (u, v) мы побили все камни на пути между ними (потому что мы ставили камни только на путях, и больше нигде). Рассмотрим какое-то множество героев A, которое бьет все камни на графе. Рассмотрим такое ребро (u, v), что герой в u и герой в v не принадлежат выбранному множеству A. В предыдущем абзаце мы показали, что на пути (u, v) должен быть хотя бы $l_i + 1$ выбранный герой. Давайте удалим всех выбранных на пути (u, v) героев из A, встанем в вершину u, и начнем добавлять в A героев через одного. Это проиллюстрировано на Рис. 10. Заметим, что мы добавим ровно $l_i + 1$ героя и, значит, размер множества A не увеличится. При этом ясно, что множество A все еще будет быть все камни на всей решетке, потому что герои строго внутри пути (u, v) влияют только на камни внутри пути (u, v), и мы после операцией все еще бьем все камни на пути (u, v). Таким образом, если существует множество героев A, которое бьет все камни, то существует и такое множество героев $A' : |A'| \leq |A|$, которое также бьет все камни, и для которого выполняется свойство, что для любого ребра (u, v) выбран либо герой в u, либо герой в v.

Итак, пусть существует множество героев размера A размера не более $L+k$, которое бьет все камни. Построим по нему множество героев A' размера не более $L+k$, которое для каждого ребра (u, v) содержит либо героя в вершине u, либо героя в вершине v.

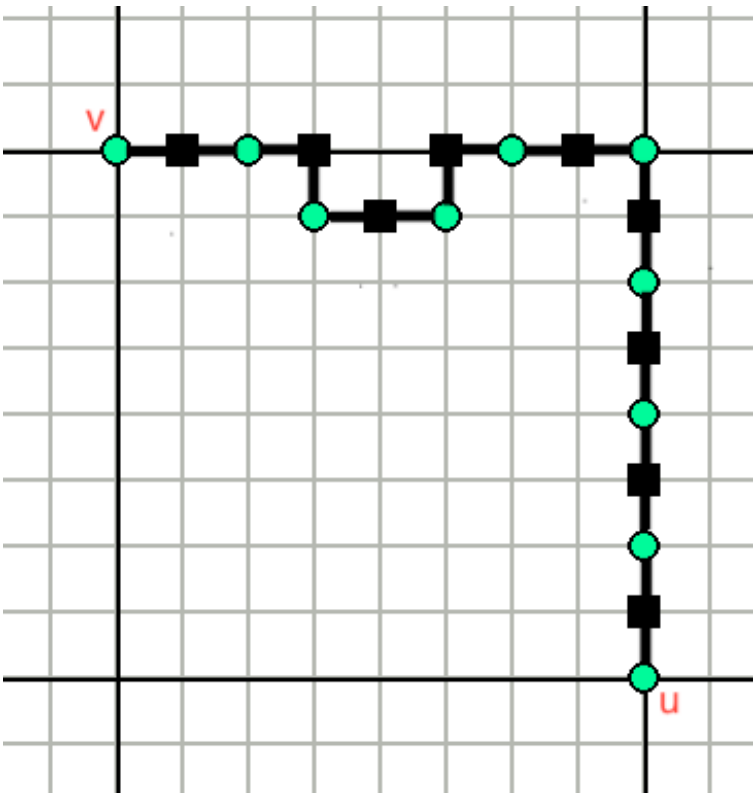


Рис. 9.

Теперь из множества A' оставим только тех героев, которые стояли в вершинах исходного графа. Заметим, что такое множество будет образовывать вершинное покрытие графа G, потому что для любого ребра (u, v) в нем будет либо u, либо v. При этом на внутренности пути (u, v), ассоциированным с i-м ребром (u, v), множеству A принадлежат не менее l_i героев. Это значит, что на всех внутренностях путей у нас хотя бы $L(= \sum_{e \in E} l_i)$. А это значит, что после их выкидывания у нас останется что-то меньше чем $(L + k) - L = k$ вершин. Это и есть вершинное покрытие исходного графа G размера не более чем k. Чтобы сделать размер ровно k, можно просто докинуть лишние вершины.

Теперь в другую сторону: пусть существует вершинное покрытие V' графа G размера k. Отметим на решетке тех героев, которые расположены в вершинах V'. Заметим, что по определению вершинного покрытия для пути на решетке, ассоциированного с любым ребром (u, v) отмечен будет либо герой в вершине u, либо в v, либо оба. Не умаляя общности в u. Тогда поставим еще l_i героев как на самом правом изображении с Рис. 10. Если в вершине v герой тоже отмечен, это нам не помешает. Заметим, что таким образом мы побьем все камни, и отметим еще ровно L героев. Это значит, что всего выбранных героев будет $L + k$, что и требовалось.

Шаг 5. *Сведение к MERGE-GAME-SIMPLE.* В предыдущем пункте мы научились сводить PLANAR VERTEX COVER-DEG3 к выбору $L + k$ из $2L + n$ героев в графе на

целочисленной решетке так, чтобы выбранные герои били все камни. В MERGE-GAME-SIMPLE мы требовали выбрать не более $\lfloor \frac{h}{2} \rfloor$ героев. Это значит, что если $L + k = \lfloor \frac{2L+n}{2} \rfloor = L + \lfloor \frac{n}{2} \rfloor$, то сведение получилось. Но что делать, если это неравенство не выполняется? Тут есть два случая.

Случай 1. $L + k > \lfloor \frac{2L+n}{2} \rfloor$.

Давайте заметим, что если добавить в граф на решетке куда-нибудь далеко q героев так, чтобы они ни с чем не стояли рядом, то в новом графе с q дополнительными героями для уничтожения всех камней нам все еще нужно будет выбрать не более $L+k$ героев, потому что добавление новых героев не мешает уничтожению. Всего героев станет $2L + n + q$. Тогда если мы выберем q так, что $L + k = \lfloor \frac{2L+n+q}{2} \rfloor$, мы построим сведение к MERGE-GAME-SIMPLE. Видно, что при $q = 2 \cdot (k - \lfloor \frac{n}{2} \rfloor)$ это равенство выполняется. Что и требовалось для сведения.

Случай 2. $L + k < \lfloor \frac{2L+n}{2} \rfloor$. Назовем *чертовой парой* камень и героя, которые соединены единичным ребром на решетке. Давайте заметим, что если добавить в граф на решетке куда-нибудь далеко q чертовых пар так, чтобы они ни с чем не стояли рядом, то в новом графе с q дополнительными героями и q дополнительными камнями для уничтожения всех камней нам уже нужно будет выбрать на q больше героев, потому что добавление чертовой пары обязывает включить героя из неё в ответ. Это значит, что в новом графе нам нужно будет найти множество, которое бьет все камни, размера не более $L + k + q$. Всего героев станет $2L + n + q$. Тогда если мы выберем q так, что $L + k + q = \lfloor \frac{2L+n+q}{2} \rfloor$, мы построим сведение к MERGE-GAME-SIMPLE. Видно, что при $q = 2 \cdot (\lfloor \frac{n}{2} \rfloor - k)$ это равенство выполняется. Что и требовалось для сведения.

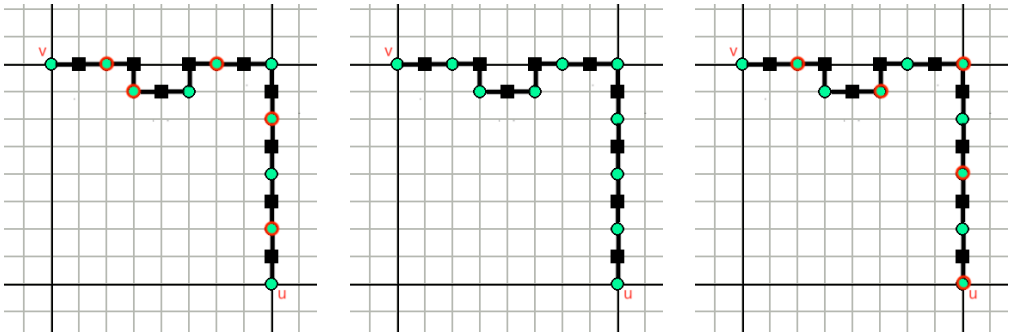


Рис. 10. Изменение выбранных на (u, v) героев

Таким образом, **PLANAR VERTEX COVER-DEG3 свелась к MERGE-GAME-SIMPLE** и, значит, **MERGE-GAME-SIMPLE является NP-полной**. Значит, и **MERGE-GAME NP-полная**. Теорема доказана. □

3.2. Алгоритмы

Теперь мы поговорим о том, какие вообще существуют алгоритмы для решения MERGE-GAME. Для удобства введем также оптимизационную версию этой задачи:

MERGE-GAME-OPTIMIZE, в которой необходимо найти максимальное количество камней, которых можно побить героями. Задача MERGE-GAME является ее очевидным следствием.

MERGE-GAME-OPTIMIZE
Вход:

- Натуральное число n – размер карты.
- Множество из h героев $H^1, H^2, \ldots, H^h$. Каждый герой находится в клетке H^i_{pos}, имеет свой тип H^i_t и начальный первый уровень.
- Множество из s камней $S^1, S^2, \ldots, S^s$. Каждый камень находится в клетке S^i_{pos} и имеет свой начальный уровень здоровья S^i_{hp}

Выход: Какое наибольшее количество камней можно уничтожить?

Теорема 3. *Существует алгоритм, который решает задачу MERGE-GAME-OPTIMIZE за* $O^*(4^h)$.

Доказательство. Сначала приведем рассуждения, когда есть только один тип героев. Заметим, что от дополнительных мерджей количество уничтоженных камней может только увеличиться. Поэтому будем считать, что мы всегда делаем мерджи по максимуму, то есть что в конце игры не остается двух героев одного уровня. Теперь давайте следующим образом переберем все мерджи. На первом шаге выберем $\lfloor \frac{h}{2} \rfloor$ героев, которые достигнут второго уровня, за счет мерджей с оставшейся половиной (в любом порядке). На втором шаге среди $\lfloor \frac{h}{2} \rfloor$ героев второго уровня тоже выберем $\lfloor \frac{\lfloor \frac{h}{2} \rfloor}{2} \rfloor$ героев, которые достигнут третьего уровня, и так далее. Тогда количество вариантов, которые мы переберем, можно оценить сверху как

$$\binom{h}{\frac{h}{2}} \cdot \binom{\frac{h}{2}}{\frac{h}{4}} \cdot \ldots \cdot \binom{2}{1}$$

Достаточно известная оценка на биномиальные коэффициенты говорит о том, что

$$\binom{2n}{n} \leq 2^{2n}$$

. Поэтому количество вариантов можно еще оценить сверху как

$$2^h \cdot 2^{\frac{h}{2}} \cdot 2^{\frac{h}{4}} \cdot \ldots \cdot 2^1 \leq 2^{2h}$$

. Таким образом, мы получили не более 2^{2h} вариантов, каждый из которых можно промоделировать за полином. Это и показывает существование алгоритма за $O^*(4^h)$.

Если типов несколько, то те же самые рассуждения нужно проводить независимо для каждого типа, и затем перемножить все варианты, чтобы получить итоговое количество. Пусть h_i – количество героев i-ого типа, тогда всего мы рассмотрим не более 4_i^h вариантов мерджей i-ого типа. Количество всех вариантов это просто

$$\prod 4^{h_i} = 4^{\sum h_i} = 4^h$$

что нас устраивает. □

Следствие 1. *Если добавить ограничение HP-ONE (т. е. уровень жизни каждого камня изначально* 1*), то задача решается за* $O^*(2^h)$

Доказательство. Действительно, если жизни каждого камня ограничены 1, то из замечания к Лемме 3.1 следует, что нет смысла создавать героев уровня больше чем 2. Поэтому перебор из предыдущей теоремы останавливается на первом же шаге, и мы просто перебираем $\binom{h}{\lfloor \frac{h}{2} \rfloor}$ способов выбрать героев, чей уровень станет вторым. Этот биномиальный коэффициент оценивается сверху как 2^h. □

3.3. Полиномиальные модификации

В предыдущих разделах мы считали, что при мердже герой наносит урон соседним по сторонам клеткам. Теперь попробуем изменить этот параметр.

Определение 6. *Вектором атаки* героя называется список клеток, стоящих вокруг него, по которым он наносит удар при мердже.

В наших задачах мы считали, что герой бьет камни, которые являются его соседями по стороне, и вектор атаки героев выглядел так, как показано на Рис. 14.

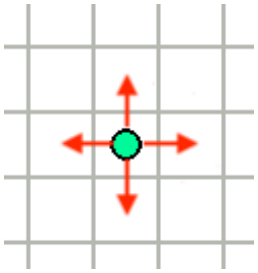
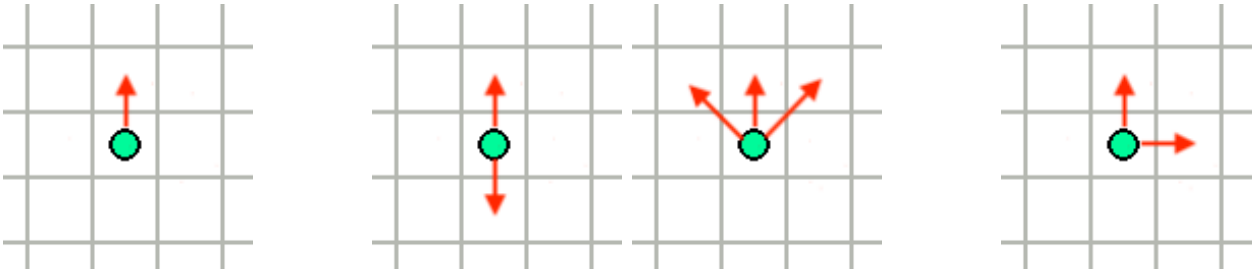


Рис. 11. Вектор атаки, который мы использовали по умолчанию

Теперь рассмотрим задачи, в которых вектор атаки героев выглядит другим образом. На Рис. 12 нарисованы эти векторы атаки.

Теорема 4. *Задача MERGE-GAME-SIMPLE имеет полиномиальное решение, если у всех героев вектор атаки I-ого типа.*



(a) I тип вектора атаки (b) II тип вектора атаки (c) III тип вектора атаки (d) IV тип вектора атаки

Рис. 12. Различные векторы атаки

Доказательство. Этот случай является очень простым. Заметим, что под каждым камнем обязан быть герой, и мы обязаны выбрать этого героя, чтобы побить камень. Таким образом, мы можем уничтожить все камни тогда и только тогда, когда под каждым камнем стоит герой, и $s \leq \lfloor \frac{s}{2} \rfloor$. Это легко проверить за полином. □

Теорема 5. *Задача MERGE-GAME-SIMPLE имеет полиномиальное решение, если у всех героев вектор атаки II-ого типа.*

Доказательство. Давайте заметим, что герои могут бить камни только внутри столбца, в котором они находятся. Решим более сильную задачу: какое минимальное количество героев нужнов выбрать, чтобы побить все камни (в оригинальной MERGE-GAME-SIMPLE у нас есть бюджет $\lfloor \frac{h}{2} \rfloor$). Будем решать задачу для каждого столбика независимо, и потом сложим ответы. Для каждого столбика будем идти по камням сверху, и действовать по следующему алгоритму: если на очередном шаге этот камень еще не побит, то выберем самого нижнего героя, который его бьет, и добавим в наше множество. Ясно, что на каждом шаге мы выберем наименьшее количество героев, которые бьют все камни на префиксе, причем из таких вариантов выберем самый лучший в том смысле, что он потенциально сможет побить еще одного героя ниже последнего камня.

Этот алгоритм, очевидно, работает за полиномиальное время. □

Теорема 6. *Задача MERGE-GAME-SIMPLE имеет полиномиальное решение, если у всех героев вектор атаки III-ого типа.*

Доказательство. В случае третьего типа нам понадобится сделать некоторую декомпозицию героев и камней. Давайте проведем ребра между камнями и героями, которые их могут побить. Разобьем получившийся граф на компоненты связности. Ясно, что каждая компонента связности будет помещаться на двух строчках, причем в верхней строке будут только камни, а в нижней только герои. Теперь сделаем все то же самое, что и в предыдущей теореме: будем считать минимальное количество героев, которыми можно побить все камни независимо в каждой компоненте связности. Для каждой компоненты связности будем точно также идти по камням слева направо, и на очередном шаге выбирать самого правого героя, который может побить текущий камень.

Ясно, что этот алгоритм работает за полиномиальное время. □

Теорема 7. *Задача MERGE-GAME-SIMPLE имеет полиномиальное решение, если у всех героев вектор атаки IV-ого типа.*

Доказательство. Повернем нашу таблицу на 90 градусов, и получим картинку, которая идентична III-ему типу вектора атаки. Точно так же построим компоненты связности, и решим задачу жадно независимо на каждом куске. □

Таким образом, MERGE-GAME-SIMPLE имеет полиномиальное решение для всех типов векторов атаки I-IV.

Попробуем убрать из теоремы 7 ограничение ONE-HP.

Теорема 8. *Задача MERGE-GAME с ограничениями DESTROY-ALL, EQUAL-TYPE и LIMITED-LEVEL может быть решена за полиномиальное время.*

Доказательство. Обозначим за k ограничение уровня героя (это ограничение берется из LIMITED-LEVEL).

По аналогии с теоремой 7 разобьем героев и камни на компоненты связности, которые можно уместить в 2 линии. Перенесем все компоненты связности и выстроим их в два ряда так, чтобы в верхнем ряду стояли только камни, в нижнем ряду стояли только герои, и между соседними компонентами связности был блок из 2×3 пустых клеток. Ясно, что длина получившейся полоски не больше, чем $4(h + s)$. Введем новую

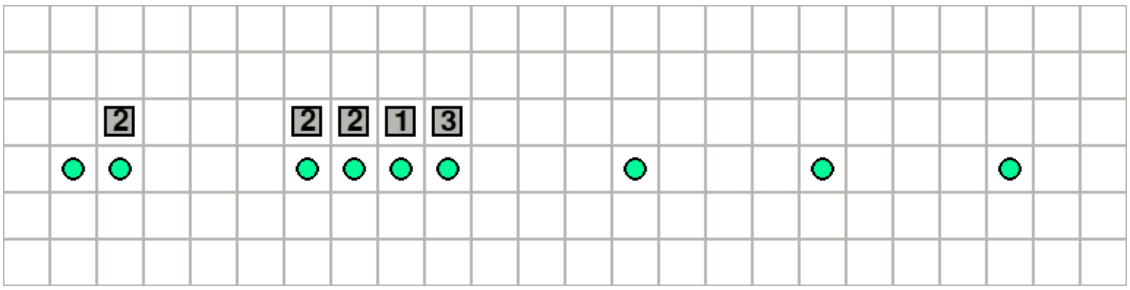


Рис. 13. Выстраивание в две линии

характеристику героя, которая будет показыть, какой урон нанес этот герой соседним камням. Обозначим её за d_i для i-ого героя. Легко увидеть, что $d_i = lvl_i - 1$ в конце игры (если герой исчез, то берется последнее значение его уровня).

Заметим, что если для каждого героя у нас известен конечный d_i, то мы можем жадным алгоритмом проверить, можно ли корректен ли набор d_i, то есть существует ли такая последовательность мерджей, что i-й герой нанесет урон хотя бы d_i.

Это дает идею, какую можно построить динамику в этой задаче — будем каким-то образом расставлять d_i. Просто так перебрать все варианты не получится, потоу что их слишком много. Чтобы сделать это эффективно, мы будем использовать идеи из задачи о рюкзаке. Введем обозначения:

- L — длина префикса на линии камней.
- C_i — количество героев среди первых $C_0 + c_1 + \cdots + c_{k-1}$, которые нанесли урон i.
- $heroes$ — количество героев, которых мы рассматриваем: $heroes = C_0 + C_1 + \cdots + C_{k-1}$
- $lastD$ — урон, который нанес последний герой среди тех, кого мы рассматриваем.
- h_1 — дополнительный урон, который пришел с неба на ячейку в верхней строке с номером L
- h_2 — дополнительный урон, который пришел с неба на ячейку в верхней строке с номером $L - 1$
- h_3 — дополнительный урон, который пришел с неба на ячейку в верхней строке с номером $L - 2$

Тогда введем динамическое программирование $dp[L][C_0][C_1]..[C_{k-1}][lastD][h_1][h_2][h_3]$, которое означает, можно ли уничтожить все камни, стоящие в первых L ячейках, если рассматривать только первые $heroes$ героев, среди которых C_i героев нанесут урон ровно i, последний герой нанесет урон ровно $lastD$, и есть дополнительные условием, что последняя ячейка префикса длины L получила перед началом игры дополнительный урон (из ниоткуда) h_1, предпоследняя — h_2, предпредпоследняя — h_3.

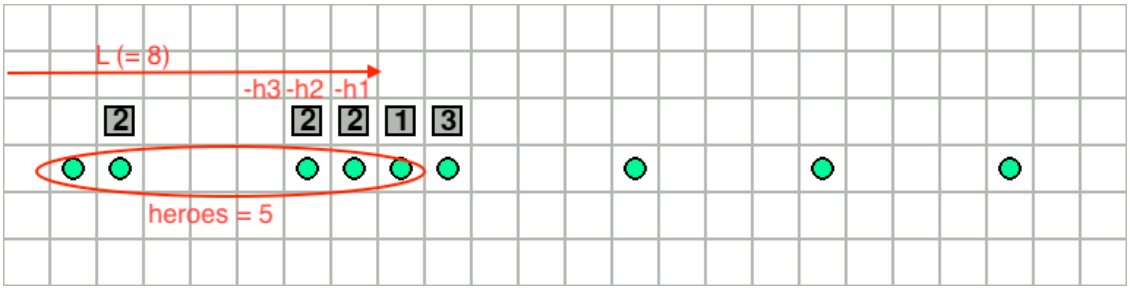


Рис. 14. Выстраивание в две линии

Чтобы пересчитать эту динамику, нужно всего лишь аккуратно рассмотреть все случаи, куда бьет последний камень, и пересчитать $dp[L][C_0][C_1]..[C_{k-1}][lastD][h_1][h_2][h_3]$ через другие состояния, в которые мы после этого перейдем. Также нельзя забывать про то, что C_{lastD} должен быть строго положительным.

Заметим, что все камни можно уничтожить тогда и только тогда, когда существует такой достижимый набор $(C_0, C_1, \ldots, C_{k-1})$ и $lastD$ такой, что $dp[N][C_0][C_1]...[C_k][lastD][0][0][0] = true$, где N — длина полоски.

Займемся теперь изучением асимптотики динамики. Заметим, что количество её состояний не больше, чем $N \cdot (h \cdot h \cdot \cdots \cdot h) \cdot k \cdot (3k) \cdot (3k) \cdot (3k)$, поскольку мы перебираем

координату L до N, количество C_i мы перебираем до h, последний уровень мы перебираем до k, и дополнительный урон мы перебираем до $(3k)$, поскольку именно такой максимальный урон может получить камень. Итоговое количество состояний динамики это $O(Nh^k k^4) = O(Nh^k)$, что является полиномом при условии того, что k — константа.

Заметим, что мы пересчитывали динамику только через ее предыдущие значения, и искали среди них $true$. Это значит, что динамика работает за полином. Получается, что и весь алгоритм работает за полиномиальное время, что и требовалось.

□

4. Заключение

В нашей работе мы определили вычислительную задачу MERGE-GAME, основанную на компьютерной игре Merge War. Было показано, что MERGE-GAME является NP-полной задачей, и, более того, в этой задаче можно создать большое количество ограничений так, что упрощенная версия игры MERGE-GAME-SIMPLE все еще будет NP-полной.

Для задачи MERGE-GAME и некоторых ее модификаций были предложены экспоненциальные алгоритмы, детерменированно находящие точное решение. Была изучена алгоритмическая сложность задачи после упрощения вектора атаки героев. Получилось доказать, что в такой постановке существуют полиномиальные алгоритмы.

Также по результатам нашего исследования мы предлагаем несколько открытых задач, ответ на которые интересно было бы узнать.

- Является ли задача MERGE-GAME NP-полной, если у героев могут быть различные вектора атаки (например, герои одного типа могут бить вверх, а другого типа – вправо)?
- Существует ли полиномиальный алгоритм для задач из теорем 4 - 7, если убрать ограничение ONE-HP (то есть у камней может быть произвольный изначальный hp) и не ограничивать максимальный уровень героев?

Список литературы

[AAD15] Ahmed Abdelkader, Aditya Acharya, and Philip Dasler. On the complexity of slide-and-merge games, 2015.

[AAD16] Ahmed Abdelkader, Aditya Acharya, and Philip Dasler. 2048 Without New Tiles Is Still Hard. In Erik D. Demaine and Fabrizio Grandoni, editors, *8th International Conference on Fun with Algorithms (FUN 2016)*, volume 49 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[GLN14] Luciano Gualà, Stefano Leucci, and Emanuele Natale. Bejeweled, candy crush and other match-three games are (np-)hard, 2014.

[Hay16] Michael Haythorpe. Reducing the generalised sudoku problem to the hamiltonian cycle problem, 2016.

[Kan92] G. Kant. Drawing planar graphs using the lmc-ordering. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 101–110, Los Alamitos, CA, USA, oct 1992. IEEE Computer Society.

[KANP07] Jussi Kuittinen, Kultima Annakaisa, Johannes Niemelä, and Janne Paavilainen. Casual games discussion. pages 105–112, 11 2007.

[KPS08] Graham Kendall, Andrew Parkes, and Kristian Spoerer. A survey of np-complete puzzles. *ICGA Journal*, 31:13–34, 03 2008.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.