Solving the 2048 game

Есть такая довольно известная игра, 2048: https://en.wikipedia.org/wiki/2048_(video_game). Цель данного проекта – узнать, можно ли гарантированно выигрывать, как бы ни не везло с ответными плитками. В этой постановке вопроса можно считать, что теперь у нас игра двух игроков – пользователя и злого компьютера, цель пользователя – получить плитку 2048, цель компьютера – так ставить ответные плитки, чтобы помешать пользователю. Наш вопрос теперь можно переформулировать так: кто выигрывает (при правильной игре) в эту игру?

Есть гипотеза, что компьютер, причём на самом деле не только 2048, но и гораздо меньшие плитки невозможно набрать. Гипотеза основана на том, что были вычислены максимальные гарантированно достижимые плитки для всех меньших размеров полей, включая 3x4: см. https://github.com/elexunix/solve-2048/blob/main/old/current-results.jpg. Как мы видим, числа гораздо меньше 2048, и есть гипотеза, что и для 4x4 число небольшое, не будет вдруг резкого скачка. Если это так, то можно решить игру для гораздо меньшей цели (скажем, 256, кажется, звучит естественным предложением), такая игра имеет гораздо меньше позиций, поэтому гораздо быстрее и дешевле решается (на кластере), и если ответ для какой-то такой игры будет получен “нет”, что ожидается, то и ответ на весь вопрос тоже будет “нет”. Такой несколько обходной путь, вера в который основана на результатах меньших экспериментов.

Решать планируется обходом неявного графа всех позиций, распределённым вычислением. Оценки показывают, что должно быть достаточно 350000 ядро-часов. Также, если сохранять промежуточные результаты на диск, что мы собираемся делать, будет ещё возможность сделать приложение (может быть, страничку на сайте Вышки или ещё где-то), которое, пользуясь сжатыми данными о том, какие позиции выигрышны, может быстро играть с пользователем, либо за пользователя, и демонстрировать выигрышную стратегию. Поскольку мы ожидаем, что игра будет выигрышна за компьютер, тем удобнее визуализировать этот результат: можно создать страничку, на которой можно будет поиграть в 2048, но у неё будет база, с помощью которой гарантированно компьютер сможет помешать пользователю выиграть, а на самом деле даже набрать сколько-нибудь большую плитку, потому что ожидается же, что мы покажем проигрышность за пользователя игры и с довольно маленькой целью. Такую программу мы также планируем сделать.

Более подробно о способе проведения вычисления: Мы собираемся

рассматривать позиции, начиная с позиций с большей суммой. Заметим, что полуход из позиции, где выставляется новая плитка, всегда увеличивает сумму ровно на 2 или на 4, а полуход, где пользователь делает сдвиг, всегда не меняет сумму. Это довольно сильные ограничения, позволяющие организовать вычисление эффективно. Можно разбить все позиции на слои по сумме, затем идти, начиная со слоёв с большой суммой, и обрабатывать, в конце будет ответ, в маленьких слоях суммы ≤ 4. С помощью несложных методов динамического программирования возможно индексироваться внутри слоя константной суммы, то есть быстро получать позицию по номеру, и номер по позиции. Ещё заметим, что крайне желательно хранить слои в оперативной памяти, что как раз возможно благодаря тому, что переходы из слоя бывают только в два больших слоя, так что за раз можно хранить только 3 слоя, размеры которых были посчитаны вспомогательной программой и оказались как раз удачными.

Были проведены все меньшие вычисления, самое большое из которых, для поля 3x4, действительно использует все описанные в предыдущем абзаце наблюдения и методы, и оно работаем на личном ноутбуке на 7 ядрах (плюс одно раздающее задания, так оказалось эффективнее, чем занять 8) за 42 минуты. Также были дополнительно добавлены операции с диском (в нужные времена запись-чтение слоёв), без которых невозможно было бы в реальном вычислении, потому что все позиции сразу всё же не сохранить в память, и я убедился, что эти операции незначительно влияют на время работы – оно осталось 42 минуты (кстати, саму программу можно взять на https://github.com/elexunix/solve-2048/tree/main/reachability-256/jul/field3by4-cpu3.cpp, скомпилировать с `g++ -pthread -O2`).

Мы считаем, что само вычисление можем провести с 350000 ядро-часами. Расчёт такой: 42 минуты на семи ядрах это 5 ядро-часов. Для решения игры на 3x4 (где максимальная гарантированно достижимая плитка 64) нужно было рассмотреть 7^{12} позиций (-, 2, 4, 8, 16, 32 или 64 в каждой из 12 клеточек), а для 4x4 (где мы проверяем, можно ли набрать 256) – 8^{16} позиций. Таким образом, где-то 100000 ядро-часов. Однако обрабатывать поля 4x4 немного сложнее, чем 3x4, поэтому нужно это число немного увеличить, и ещё нужно добавить запас, мы считали, что так 180000 нормально. Но было проведено дополнительное вычисление, которое показало, что для настоящего вычисления потребуется использовать тип `long long` в некоторых местах, где раньше хватало `int`, и маленькое вычисление стало занимать 76 минут вместо 42, если в него намеренно добавить это изменение. Поэтому в итоге мы считаем, что нормально 350000 ядро-часов.