Анализ на задача „Зарчета“

Автор: Емил Инджев
Анализ и материали: Енчо Мишинев

Както в повечето output-only задачи, тук не се очаква да намерим оптимален отговор – не ни е известен бърз алгоритъм, който намира най-добрия комплект зарчета. Целите T са най-добрите резултати, които журито е намерило.

Системата взима най-добрия резултат за всеки тест поотделно, така че е добра идея подходите да се комбинират. По-долу те са подредени от най-прости към най-силни. Точките, които споменаваме, са от примерни реализации на журито и са ориентировъчни.

Няколко наблюдения

Ограниченията са сравнително малки. Редът на стените няма значение, така че едно зарче е просто разбиване на S на K неотрицателни събираеми. При малки K и S различните зарчета са малко: 34 при $K = 4$, $S = 12$ (тестове 02 и 05), 22 при $K = 8$, $S = 8$ (тест 03), 55 при $K = 10$, $S = 11$ (тестове 09 и 17), 84 в тест 06 и 108 в тест 07. При максималните ограничения $K = 10$, $S = 100$ броят на различните зарчета е около 6 милиона, което също е достатъчно малко, ако искаме да генерираме всички такива.

Посоката на предимството е по-важна от големината му. Заради добавката ± 1 дори минимална победа носи повече от 1, а минимална загуба – по-малко от -1. Съответно първо искаме да спечелим колкото може повече битки и чак след това да увеличаваме предимството. На практика голяма част от решенията директно оптимизират резултата, без да се интересуват от този детайл, но той е полезен за някои greedy подходи, евристики или ръчни конструкции.

Интуиция за една битка. Дали една стена печели срещу друга зависи само от това коя е по-голяма, а не с колко. Стена със стойност v побеждава всички по-малки стени на противника и за това „плаща“ v от сбора S. Всичко над най-голямата стена, която тя побеждава, е похарчено напразно. Затова зарче, което трябва да печели, е добре да раздели сбора си така, че всяка негова стена да е малко над възможно повече стени на противника. Зарче, което трябва да губи, е добре да концентрира сбора си в малко на брой много големи стени, всяка от които така или иначе не може да победи повече от K стени.

Ако приложим тази идея при само две зарчета, губещото зарче става $(0, 0, \ldots, 0, S)$. Тогава на печелившото е достатъчно да няма нули (това е възможно при $S \geq K$, което е изпълнено във всички тестове), а как точно е разпределен сборът му няма значение. Всяка негова стена побеждава $K - 1$ нули и губи от стената S, така че $P = K(K-1)$, $Q = K$ и $D = \frac{K-2}{K}$.

Не е задължително да има еднозначна подредба на зарчетата по сила. Възможно е зарче X да има предимство срещу Y, Y срещу Z, а Z срещу X. Точно на това разчитат тестовете, съдържащи цикли.

Конструкции (тестове 01–05)

Първите пет теста могат да се решат на ръка с малко наблюдения.

Тест 01 е една битка с $K = 6$, $S = 21$. По наблюдението по-горе: $(3, 3, 3, 4, 4, 4)$ срещу $(0, 0, 0, 0, 0, 21)$ дава $D = \frac{4}{6}$ и резултат $1.6666\ldots$

Тест 02 е цикъл от три зарчета с $K = 4$, $S = 12$: искаме 0 да бие 1, 1 да бие 2 и 2 да бие 0. Едно решение е

$$(0, 4, 4, 4) \;\to\; (2, 2, 2, 6) \;\to\; (1, 1, 5, 5) \;\to\; (0, 4, 4, 4),$$

с предимства $\frac{2}{16}$, $\frac{4}{16}$ и $\frac{4}{16}$. Зарчетата са само 34, така че можем да намерим оптималното решение и с програма, която пробва всичките 34^3 комплекта.

Тест 03 е верига $0 \to 1 \to 2 \to 3$ с $K = 8$, $S = 8$. Тук оптимално е всяко следващо зарче да е „по-събрано“ от предишното:

$$(1,1,1,1,1,1,1,1), \quad (0,0,0,0,0,1,1,6), \quad (0,0,0,0,0,0,4,4), \quad (0,0,0,0,0,0,0,8).$$

Тест 04 изглежда голям (12 зарчета, 36 битки), но ако погледнем входа, виждаме, че всяко от зарчетата 0–5 трябва да бие всяко от зарчетата 6–11. Това е на практика еквивалентно на една битка: на първите шест даваме $(5,5,5,5,5,5)$, а на вторите шест – $(0,0,0,0,0,30)$. Целта T е същата като на тест 01, което подсказва за сходността.

Тест 05 пък е сходен с тест 02. Зарчетата са в три групи по пет: всяко от 0–4 трябва да бие всяко от 5–9; всяко от 5–9 да бие всяко от 10–14; и всяко от 10–14 да бие всяко от 0–4. Параметрите K и S са като в тест 02, затова просто даваме на всяка група по едно от трите зарчета оттам. Целта T отново съвпада.

Пълно изчерпване (тестове 06, 07)

Задачата е output-only, така че програмата ни може да работи и минути.

В **тестове** 06 **и** 07 има по 5 зарчета, а възможните различни зарчета са съответно 84 и 108. Това са $84^5 \approx 4 \cdot 10^9$ и $108^5 \approx 1.5 \cdot 10^{10}$ възможни комплекта. Ако предварително сметнем таблица с резултата на всяко зарче срещу всяко друго, пълното изчерпване отнема няколко минути.

Оптимално решение за прости цикли (тестове 07, 09, 17)

Тестове 07, 09 **и** 17 са цикли (със съответно 5, 9 и 25 зарчета), а за цикъл се оказва, че има бързо и точно решение с динамично програмиране.

Нека различните възможни зарчета са m на брой и предварително сме сметнали таблица $W[p][q]$ – резултата от битка, в която зарче p трябва да победи зарче q. Нека цикълът е: $v_0 \to v_1 \to \cdots \to v_{N-1} \to v_0$.

Фиксираме зарчето f на позиция v_0 и пускаме динамичното за всяко f поотделно. Нека $dp[i][p]$ е най-големият сбор от резултатите на битките $v_0 \to v_1$, …, $v_{i-1} \to v_i$, ако на позиция v_i е зарче p. В началото $dp[0][f] = 0$, а за $p \neq f$ стойността $dp[0][p]$ е $-\infty$. Пресмятаме, пробвайки всяко зарче q за предишната позиция:

$$dp[i][p] = \max_q \big(dp[i-1][q] + W[q][p]\big).$$

Накрая затваряме цикъла с битката $v_{N-1} \to v_0$, като най-добрият сбор при това f е

$$\max_p \big(dp[N-1][p] + W[p][f]\big).$$

Отговорът е най-голямата от тези стойности от всички f. Самите зарчета възстановяваме, като за всеки стейт помним от кое q е получен.

Сложността е $O(N \cdot m^3)$. При $m = 108$ и $N = 5$ (тест 07) това са около 6 милиона операции, а при $m = 55$ – около 1.5 милиона за $N = 9$ (тест 09) и около 4 милиона за $N = 25$ (тест 17).

Заедно с ръчните конструкции дотук имаме оптималните отговори на девет теста, т.е. 45 точки. По-долу за по-кратко наричаме пълното изчерпване и решението за циклите заедно просто „изчерпване“.

Greedy стратегии

За големите тестове първата естествена greedy идея е да построим зарчетата едно по едно. Например:

- Нареждаме зарчетата според това колко битки трябва да спечелят и колко да загубят. На тези, които предимно губят, даваме „събрани“ зарчета (много нули и една-две големи стени), а на тези, които предимно печелят – по-равномерни конфигурации.
- Обхождаме зарчетата в някакъв ред и за всяко избираме най-доброто зарче срещу вече избраните му противници (измежду всички възможни зарчета или измежду някакъв набор от кандидати). Може да направим и втори такъв обход.

Съществуват много видове подобни евристики и можем да пробваме всички, които ни хрумнат, избирайки най-добрия отговор. Такива решения са бързи за писане, но сравнително слаби: на големите тестове стигат до 30%–70% от целта. Заради третата степен във формулата за оценяване това са малко точки – например 70% от целта носи $5 \cdot 0.7^3 \approx 1.7$ точки от 5. Самостоятелно greedy решенията на журито получават между 11 и 32 точки.

В комбинация. Ръчните конструкции и изчерпването покриват девет теста, а на останалите единадесет едно greedy решение носи до около две точки на тест. Така 45-те точки стават между 47 и 60 според това колко е добро greedy решението. Ако сме направили само ръчните конструкции, без изчерпването, с едно greedy решение стигаме до 28–45 точки.

Системата взима най-доброто решение за всеки тест поотделно, затова си струва да се събмитнат няколко различни решения: най-доброто по тестове измежду деветте greedy решения на журито носи 42 точки самостоятелно, 51 заедно с ръчните конструкции и 63 заедно с конструкциите и изчерпването.

Random Search

Още по-просто е да генерираме много пъти напълно случайни комплекти от зарчета и да запазим най-добрия. Случайно зарче се генерира например като разпределим S-те единици независимо по случайни стени.

При малко на брой зарчета това работи изненадващо добре: с милион опита се намират оптимумите на тестове 01 и 02 и почти оптимумите на 03 и 07. При много зарчета обаче шансът всички битки да са с правилното предимство едновременно е много малък: на тестовете с 20 и повече зарчета random search стига до 25%–55% от целта, което е под една точка на тест. Самостоятелно това решение достига около 35 точки.

В комбинация. Random search печели точките си предимно на малките тестове, а точно тях ръчните конструкции и изчерпването решават напълно. Затова заедно с тях добавя малко: от 45 точките стават 52, като около три от тях идват от тестове 08 и 10. Ако имаме и добро greedy решение, random search добавя най-много още една точка – от останалите тестове то е по-добро от greedy само на тест 08. Всички подходи дотук заедно (ръчни конструкции, изчерпване, няколко greedy решения и random search) дават общо около 64 точки.

Random search е полезно и като евтин заместител на изчерпването. Само с ръчните конструкции и random search се достигат 42 точки, защото на тестове 06, 07 и 09 то взима съответно 3.5, 4.1 и 2.0 точки, без да пишем нищо специално за тях.

Hill Climbing

Hill climbing е може би най-простият подход за подобряване на едно решение, без да разбираме как изглеждат добрите решения на дадена задача. Идеята е да поддържаме едно текущо решение и многократно да му правим малка случайна промяна. Ако резултатът не се влоши, запазваме промяната, а иначе я отменяме.

При задачи като тази това е една от първите идеи, които трябва да ни хрумнат, по няколко причини:

- Трябват ни само две неща: функция, която оценява дадено решение (нея така или иначе трябва да я напишем), и „малка промяна“, която запазва решението валидно. Цялата програма е доста кратка.
- Задачата е output-only, така че програмата може да работи минути.
- Малка промяна в едно зарче променя резултата малко: засяга само битките на това зарче и само няколко двойки стени в тях. Затова близо до добро решение обикновено има още по-добро и постепенното подобряване работи.
- Точките зависят от това колко близо сме до целта, а не от това дали сме я достигнали. Тоест оценяващата ни функция се променя сравнително постепенно.
- Почти всички по-силни подходи (вижте следващия раздел) надграждат hill climbing, така че такова решение винаги е добра основа.

За имплементацията на тази идея в нашата задача започваме от случаен комплект. Най-простата промяна е:

избираме случайно зарче и две негови стени; от едната махаме 1, на другата добавяме 1.

Сборът на зарчето се запазва автоматично. Няколко неща, които могат да подобрят решението:

- Добре е да приемаме и промени, които *не променят* резултата. Така търсенето може да разглежда различни конфигурации, които имат еднакъв резултат, позволявайки му да обходи по-разнообразни посоки.
- Когато дълго време няма подобрение, можем да рестартираме процеса от нов случаен комплект и накрая да изведем най-доброто, намерено изобщо.
- След промяна в едно зарче се променят само неговите битки. Ако преизчисляваме само тях, а не целия резултат, можем да правим доста повече стъпки за същото време.

Дори най-простата реализация (с пълно преизчисляване и с по 60 секунди на тест) достига целта на 9/10 от тестове 01–10 (на тест 08 – почти) и стига до 83%–95% от нея на тестове 11–20. Това са около 86 точки. Повече време почти не помага: с 10 секунди на тест точките са около 82, а с 5 минути – около 87. Hill climbing-ът просто засяда в локален максимум.

Друга възможна промяна е „заменяме цялото зарче с ново случайно“. Тя е по-слаба (около 75 точки), защото такива стъпки са по-агресивни и колкото по-добър става комплектът, толкова по-рядко случайно зарче го подобрява.

В комбинация. Щом имаме добър hill climbing, greedy решенията и random search вече не добавят нищо: той е не по-лош от тях на всеки тест. Единственото, което все още помага, е точното решение за циклите. На тест 17 (цикъл от 25 зарчета) hill climbing стига само до около 90% от целта, т.е. 3.5 точки, а динамичното взима всичките 5. Заедно това са около 87 точки. При по-слаб hill climbing комбинирането помага повече: вариантът със замяна на цялото зарче минава от 75 на 85 точки заедно с ръчните конструкции и изчерпването.

По-умни подходи

Защо hill climbing-ът засяда? Добрите зарчета обикновено се състоят от няколко *блока* от равни стени – например девет стени със стойност 8 и една със стойност 28. За да се вдигне целият блок с единица, трябва да се променят девет стени една по една, а междинните комплекти са по-лоши от началния. Hill climbing-ът никога не приема влошаване, затова не може да извърши такава промяна. Има няколко начина това да се поправи.

1-OPT за едно зарче. Вместо малка промяна, нека за произволно избрано зарче намираме *най-доброто възможно* зарче срещу текущите му противници. Повтаряме това за всички зарчета, докато нищо не се подобрява, и после започваме отначало от нов случаен комплект. Такова решение е изненадващо ефективно и достига около 94 точки. Понеже S и K са малки, е реалистично да пробваме абсолютно всяко възможно зарче на всяка такава итерация.

Simulated annealing. Запазваме простата промяна „премести единица“, но понякога приемаме и влошаване: ако резултатът намалява с Δ, приемаме промяната с някаква вероятност. Класическият подход е тази вероятност да е $e^{-\Delta/t}$, т.е. да намалява експоненциално с параметър t, който наричаме „температура“. Температурата t в началото е висока (приемаме почти всичко) и постепенно намалява към нула (накрая това е обикновен hill climbing). С 30 секунди на тест това носи 97–99 точки.

Най-доброто решение на журито е базирано на simulated annealing с няколко добавки:

- промени, които местят цял блок от съседни по стойност стени нагоре, а друг блок надолу, така че сборът да се запази;
- много кратки и независими итерации на simulated annealing вместо една дълга;
- след всяка итерация прилагаме 1-OPT за всяко зарче, докато има подобрение.

Целите T са най-доброто от няколко пускания на това решение, по около минута на тест.

Обобщение

В таблицата са приблизителните точки на всеки подход: самостоятелно или в комбинации.

Подход	**Самостоятелно**	**С ръчни конструкции**	**С конструкции и изчерпване**
(нищо друго)	–	25	45
едно greedy решение	11–32	28–45	47–60
няколко greedy решения	42	51	63
random search	35	42	52
няколко greedy решения и random search	52	55	64
hill climbing	75–86	79–86	82–87
1-OPT за едно зарче	94	94	94
simulated annealing	97–99	97–99	97–99
simulated annealing и 1-OPT	99	99	99