Фортуна – Анализ

Автор: Емил Инджев

1 Въведение

Може да разглеждаме това, което се иска от нас в задачата по следния начин: трябва да построим краен автомат (с до 500 състояния), който да действа като произволен генератор за M възможни стойности, използвайки такъв за N. Крайният автомат е насочен граф, такъв че всеки връх е или краен (и маркиран с кое число "връща"), или не е (и тогава има N излизащи ребра, маркирани с числата от 0 до $N-1$). Така едно произволно генериране е разходка, която започва от връх 0 и върви по ребрата с дадените стойности на произволното число R докато не стигне краен връх, в който случай връща числото, с което е маркиран той. Различните решения могат да се имплементират както с експлицитно строене и номериране на този граф (с DFS), така и с просто енкод и декод функции, които кодират текущото състояние (което най-често е двойка променливи).

Пълното решение не е специфично за различните групи тестове от условието. Те са там само защото правят някои междинни идеи по лесни за имплементиране (най-основното е, че първата група позволява да се изкарат точки без използване на състояние различно от 0).

2 Наивен rejection sampling

В полето на генериране на произволни числа от някаква дистрибуция, rejection sampling се нарича генерирането на произволно число от друга (по-лесна) дистрибуция, последвано от проверка дали стойността е валидна или не – ако е валидна, връщаме нея, а иначе рестартираме.

Тук това се състои от генериране на k цифрено число в N-ична бройна система (използвайки k стъпки), където k е най-малкото цяло число, за което $N^k \geq M$. След това първите M върха (от този k-ти слой на автомата) връщат стойността на числото, а останалите всъщност се пренасочват към корена (имаме предвид, че няма отделно ребро към такъв връх от k-ти слой и после ребро към корена, а просто още първото ребро сочи към корена).

Самото генериране на числото с N цифри може да стане по много начини, като най-общият (полезен по-късно) е да пазим като състояние (с енкод и декод) двойка числа – C (колко възможни стойности има в момента) и V (самата стойност в момента); за сега C винаги е степен на N. Т.е. разходката ще изглежда като (освен ако не спре по рано):

$(1,0) \to (N, R_1) \to (N^2, N \times R_1 + R_2) \to ...$

Забележете, че $C < M$ и $V < C$. Т.е. кодировката може да стане с триъгълно кодиране (т.е. за всяка двойка, кодът е коя подред е тя сред възможните двойки), като максималният код е $\frac{M(M-1)}{2} - 1$, което изъплнява ограниченията за задачата.

Така описано, решението прави преходи от вида $C' = NC$ и $V' = NV + R$ към (C', V'), докато не получи $C' \geq M$. Тогава, ако $V < M$, то връща V, а иначе прехода е към $(1,0)$.

Друг възможен начин да запазваме тези състояния е използвайки $P = \log_M C$ и V и преобразувайки към и от C постоянно, това не изисква триъгълната кодировка (можем да ползваме просто $MP + V$), но се оказва недостатъчно за пълното решение.

Това решение изкарва около 20 точки.

3 Rejection sampling по модул

Горното решение прави един много очевиден пропуск: ако например $C' \geq 2M$, решението ще върне V', само когато $V' < M$. Вместо това, то би могло да върне $V \mod M$, ако $C < 2M$. В общият случай, нека t е максималното t, за което $tM \leq C'$, тогава връщаме $V' \mod M$, ако $V' < tM$, а иначе отиваме към $(1, 0)$.

Това решение изкарва около 38 точки.

4 Early rejection

Една възможна оптимизация (която не води към пълното решение) е да видим, че понякога знаем, че разходката със сигурност ще стигне до връщане към $(1, 0)$. Проверката е просто да симулираме получавания само на $R = 0$ и да видим дали, дори тогава, $V' > tM$. В такъв случай, можем още тогава да се върнем към корена, вместо да чакаме това да се случи.

Това решение изкарва около 58 точки.

5 Rerouting по степени

Друга възможна оптимизация (несъвместима с предната, но по близка до пълното решение като идея) е да се замислим за случай като $N = 2$, $M = 12$. Стигаме до ситуацията, в която $C' = 16$. Тук или връщаме (когато $V' < 12$) или рестартираме (отиваме към $(1, 0)$). Това рестартиране се случва от точно $16 - 12 = 4$ върха (с равни вероятности). Вместо това бихме могли да ги пратим към $(4, 0)$, $(4, 1)$, $(4, 2)$ и $(4, 3)$, които са върхове вече в автомата (и представляват целият втори слой на автомата). По-общо, ако $C' - tM$ е степен на N можем да заменим преходите към $(1, 0)$ с преходи към съответния слой.

Още по-общо, можем да намерим най-голямата степен $N^s \leq C' - tM$ и да препратим тези N^s върха към съответните N^s върха от слой s. След това остават $C' - tM - N^s$ върха (които биха били препратени към $(1, 0)$); за тях повтаряме същото (отново търсим най-голямата степен и т.н.). Това повтаряне продължава, докато не останат 0 върха.

Това решение се оказва абсолютно еквивалентно на предното, просто препраща върховете по-късно. Всеки път когато предното би рестартирало s стъпки по-рано спрямо нормалния rejection sampling, след като изминат тези s стъпки, този рестарт ще стига точно до N^s-те върха от слой s. Еквивалентно, текущото решение ще си стигне до най-долния слой по нормален начин, но след това ще препрати N^s върха директно към този слой. Единствената разлика идва от точно кои пътища къде водят (това може да се елиминира, ако препращаме последните N^s върха, вместо първите N^s).

Това решение, тъй като е еквивалентно, също изкарва около 58 точки.

6 Пълно решение

Най-добрият случай за предното решение е когато $C' - tM$ е точно степен на N. Проблемите идват, когато броят оставащи върхове не вече слой в автомата. В текущата ни формулировка с двойки (C, V), това обаче въобще не е проблем (просто трябва C вече да не е винаги степен на N). Новата ни логика ще е просто: ако $V' \geq tM$, отиваме към $(C' - tM, V' - tM) = (C' \mod M, V' \mod M)$.

Това решение ще обходи много различни C-та (потенциално всички числа от 0 до $M - 1$), но рано или късно ще посети вече посетена стойност (тъй като $C < M$), т.е. решението все пак си е краен автомат.

Това е решението за 100 точки, което също е и теоретично оптималното (дори игнорирайки нуждата решението да е краен автомат).

7 Оптималност

Остава само да докажем, че решението е оптимално. Доказателството тук е по идея на Веселин Маркович.

Нека, вместо за автомат, да мислим за безкрайно N-ично дърво, в което връх или е листо (маркиран със стойността, която връща) или има N деца. Такова дърво понякога е еквивалентно на краен автомат (ако структурата му се повтаря), но ние ще докажем, че решението ни е оптимално дори сред такива дървета.

Забележете, че вероятността за стигане до който и да е връх от слой d е $\frac{1}{N^d}$, т.е. всички върхове от един слой са еквивалентни и ни интересуват само бройки. Сега, нека F_d е броят листа от слой d, а F_d^a е броят такива листа, които връщат a. За да може решението да е валиден произволен генератор, трябва, за $0 \leq a < M$:

$$\sum_{d \geq 0} \frac{F_d^a}{N^d} = \frac{1}{M}$$

От друга страна, искаме да минимизираме:

$$\sum_{d \geq 0} d \frac{F_d}{N^d} = \sum_{0 \leq a < M} \sum_{d \geq 0} d \frac{F_d^a}{N^d}$$

Ще докажем, че вътрешната сума G може да се минимизира за всяко a поотделно, и съответно общата сума да е минимална. Нека $A_d = F_d^a$ е произволна редица от бройки листа връщащи a; и O_d е тази на описаното ни решение (която твърдим че е оптимална). Искаме да докажем, че $G(O) \geq G(A)$, където:

$$G(B) = \sum_{d \geq 0} d \frac{B_d}{N^d}$$

Можем да опишем нашето текущо решение (произвеждащо редицата O) като алчно такова, защото то ефективно строи дървото слой по слой, като винаги прави максимален брой листа на текущия слой. Поточно, следим текущата префиксна сума $P_{q-1}(O)$ на $\frac{O_d}{N^d}$ до преди q-тия елемент ($P_{-1}(O) = 0$). Тогава, q-тия елемент се избира да е максималният такъв, че префиксната сума $P_q(O)$ да не надвиши $\frac{1}{M}$. (Интересно е, че това е точно и алгоритъмът за представяне на числото $\frac{1}{M}$ като N-ична дроб). За доказателството, също ще разглеждаме и суфиксните суми $S_q(O)$:

$$P_q(B) = \sum_{0 \leq d \leq q} \frac{B_d}{N^d} \qquad S_q(B) = \sum_{d > q} \frac{B_d}{N^d}$$

Важно е да видим, че суфиксните суми изпълняват $S_q(O) < \frac{1}{N^q}$. Да примем обратното: $S_q(O) \geq \frac{1}{N^q}$. Тогава, префиксната сума до позиция q изпълнява $P_q(O) \leq \frac{1}{M} - \frac{1}{N^q}$ (защото $P_q(O) + S_q(O) = \frac{1}{M}$). Това е невъзможно за алчната ни стратегия, защото тя просто щеше да избере по-голямо O_q (то да е по-голямо с 1, качва префиксната сума с $\frac{1}{N^q}$, което все още би било $\leq \frac{1}{M}$).

Също, ще докажем, че всяка префиксна сума на O е по-голяма или равна на еквивалентната на A, т.е. че $P_q(O) \geq P_q(A)$. Можем да видим, че всеки член на тези суми е кратен на $\frac{1}{N^q}$ (равен е на цяло число по $\frac{1}{N^q}$), т.е. и самите суми. Следва, че ако $P_q(O) < P_q(A)$, то $P_q(O) \leq P_q(A) - \frac{1}{N^q}$, а от там стигаме до:

$$\frac{1}{M} = P_q(O) + S_q(O) < P_q(O) + \frac{1}{N^q} \leq P_q(A) - \frac{1}{N^q} + \frac{1}{N^q} = P_q(A) \leq P_q(A) + S_q(A) = \frac{1}{M}$$

Това е противоречие, т.е. $P_q(O) \geq P_q(A)$. От това пък следва, че $S_q(O) \leq S_q(A)$. Можем да видим обаче, че $G(B)$ е равно сумата на всички суфиксни суми $S_q(B)$ на B:

$$\sum_{q \geq 0} S_q(B) = \sum_{q \geq 0} \sum_{d > q} \frac{B_d}{N^d} = \sum_{d \geq 0} \frac{B_d}{N^d} \sum_{0 \leq q < b} 1 = \sum_{d \geq 0} d \frac{B_d}{N^d} = G(B)$$

Забележете, че втората стъпка е просто пренареждане на членовете, което може да се прави със безкрайни суми на положителни членове (но не винаги може, ако има отрицателни членове).

Финално, тъй като всички суфиксни суми на O са не по-големи от съответните на A ($S_q(O) \leq S_q(A)$ за всяко q), следва, че $G(O) \leq G(A)$ (защото $G(B)$ е сумата на всички $S_q(B)$). Това доказва, доказва оптималността на редицата O и съответно на решението ни.