Лов – Анализ

Автор: Емил Инджев

1 Начален оглед

В задачата ключовите елементи са да открием как изглеждат оптималните решения (как да ги параметризираме без експоненциалност) и с какъв алгоритъм да ги намерим бързо. Също е важно да се справим с цикличността на пространството (като няколко подзадачи имат ограничение, което гарантира, че няма нужда да мислим за нея).

Както е показано и в подзадачите, важно е да се замислим за случаите $N = 1$ и $N = 2$, за да започнем със задачата. След това е въпрос на откриване на свойства, оптимизации и/или алгебрични манипулации, за да свалим сложността.

2 Подзадачи 1, 3, 5, 7 и 9

Тези подзадачи имат допълнително ограничение, което гарантира, че няма нужда да вземаме предвид цикличността на пространството. Т.е. не е нужно да работим с модулна аритметика, нито да правим съображения за интеракции между P_0 и P_{N-1} минаващи през позиция $L - 1$.

3 Подзадачи 1 и 2: $N = 1$

В ден H заекът може да е на позиции в интервала от $P_0 - H$ до $P_0 + H$ (но само на тези с четността на $P_0 \pm H$). След това след всеки капан този интервал ще се разширява с по едно от всяка страна (т.е. с 2), докато не бъде спрян от капан или докато не обхване цялата гора.

Веднъж, като заекът бъде ограничен в някакъв интервал, който няма повече да се разширява, трябва да гарантираме, че ще го заловим. Това става, като сложим капан на всяка втора позиция в интервала. Можем да видим, че това гарантира залавянето, защото всяка свободна позиция (където може да е заекът) съседства с позиции с капани, т.е заекът ще скочи в капан. Също можем да видим, че ако оставим две поредни празни позиции, заекът може само да ги редува, т.е. това не е валидно. Следва, че това е оптималната процедура, като ограничим заека в даден интервал.

Тъй като интервалът се разширява всеки ден, трябва веднага да сложим капан в единия му край: $P_0 - H$; а следващия ден в другия му край, който вече се е преместил с 1: $P_0 + H + 1$. След това запълваме всяка втора позиция: $P_0 - H + 2, P_0 - H + 4, \ldots, P_0 + H$. Заради разликата в четностите на двата крайни капана се налага някъде да има два съседни капана. Това отнема общо $H + 2$ капана.

Друга възможност е още преди ден H или в рамките на дните, в които ограничаваме заека, интервалът да се слее със себе си, т.е. да обхване цялата гора. В този случай, вместо да слагаме тези начални ограничаващи капани (които могат да се окажат неоптимални), трябва директно да слагаме капани на всяка втора позиция от гората. Това отнема $\lceil \frac{L}{2} \rceil$ капана; закръглянето нагоре е защото при нечетно L трябва веднъж да има два съседни капана.

Можем да изведем (не много сложно) условие кога интервалът ще се слее със себе си преди да го ограничим, но по-просто е да видим, че отговорът е минимума от тези два възможни отговора.

4 Подзадачи 3 и 4: $N \leq 2$

При $N = 2$ решението ще изглежда доста подобно на $N = 1$, но има още няколко случая. Двете начални позиции формират два интервала. Един вариант е да ги решим поотделно (като първо ограничим двата интервала, за да спрем растежите им, и после ги запълним). Възможно е обаче двата да се слеят от едната или от другата страна (заради цикличността). Последно е възможно да се слеят и от двете страни и така да покрият цялата гора (което е $\lceil \frac{L}{2} \rceil$ случаят).

Ако се сливат само от едната страна, ще ги решим като просто един интервал. Например, ако сливането е отдясно на P_0 и отляво на P_1, ще сложим капани на позиции $P_0 - H$ и на $P_1 + H + 1$. След това запълваме всяка втора позиция, като това отнема общо $\lfloor \frac{P_1 - P_0}{2} \rfloor + H + 2$ капана. Закръглянето е такова, защото, когато P_1 и P_0 имат различни четности, двата ограничаващи капана имат еднакви. Такъв интервал ще наричаме нечетен, а интервал с еднакви четности на двете крайни P-та – четен; в частност, интервал от само една начална стойност е четен. Случаят, в който интервалите се сливат от другата страна, е аналогичен и отнема общо $\lfloor \frac{L + P_0 - P_1}{2} \rfloor + H + 2$ капана.

Най-интересният случай е да ги ограничим поотделно. Наивно можем да решим, че първо ограничаваме първия интервал, после втория и накрая запълваме и двата. Това би отнело $(H + 2) + (H + 4) = 2H + 6$ капана, но така формираме два четни интервала и после имаме "лошо" закръгляне и при двата (т.е. и двата интервала имат съседни капани). Вместо това можем първо да ограничим левия край на първия интервал, после левия на втория, после десния на първия и накрая десния на втория (или каквато и да друга редуваща се поредица). Така ще сложим 4 ограничаващи капана, както и следните запълващи: $P_0 - H + 2, P_0 - H + 4, \ldots, P_0 + H$ и $P_1 - H + 1, P_0 - H + 3, \ldots, P_0 + H + 1$. Това отнема общо $(H + 2) + (H + 3) = 2H + 5$ капана; ефективно превръща двата четни интервала в два нечетни, като удължава първия с едно (което не добавя капани) и скъсява втория с едно (което спестява един капан).

За да намерим отговора, пресмятаме четирите случая и връщаме минималната от всички стойности.

5 Подзадачи 5 и 6: $N \leq 4$

Тук са възможни решения с много ръчни случаи или по-лоши пълни изчерпвания. Ключово е да забележим, че винаги е оптимално първо да групираме някои от последователните зайци, а после да поставим крайните капани в някакъв ред и да запълним всички интервали. Например, е възможно да итерираме през всички групирания на зайци и пермутации от крайни капани. Сложността е $O((2N)!2^N)$.

6 Подзадачи 7 и 8: $N \leq 15$

Тук се изисква по-добро пълно изчерпване. Трябва да видим, че единственото важно е как разделяме P-тата на интервали, които ще ограничим и запълним (като преди това сме сортирали P-тата). Важно е, че "спестяваме" по един капан за всеки два четни интервала (спрямо наивната сметка). Т.е. наивната цена за i-тия интервал е $\lfloor \frac{d(P_{Li}, P_{Ri})}{2} \rfloor + H + 2 + i$, където $d(x, y)$ е разстоянието между x и y (т.е. $y - x$ или $L + x - y$). Членът $+i$ идва от разширяването на интервала, докато ограничаваме всеки от предните. Обаче, пестим $\lfloor \frac{E}{2} \rfloor$ капана, където E е броя четни интервали. Самото разбиване на интервали може да стане с 2^N пълно изчерпване, т.е. общо $O(2^N N)$.

7 Подзадачи 9 и 10: $N \leq 100$

Тук вече се изисква полиномиално решение. Доста е лесно да превърнем горното решение в DP. Нека сортираме P-тата и първо да помислим за нецикличния случай. Стейтът би бил позиция на последния десен край на интервал и брой интервали, а сметката е итериране през възможни леви краища на интервала. Освен това обаче трябва да следим и четността на броя четни интервали, та е нужен още един бит в стейта. Това е общо $O(N^3)$,

Проблем е и цикличността, защото решението никога не прави интервал, където P_{N-1} е преди P_0. Можем просто да пробваме всяко i да действа като 0, защото ще има "дупка" пред поне едно i (освен ако не сме в случая $\lceil \frac{L}{2} \rceil$, който смятаме отделно). Това може да стане, като N пъти направим цикличен шифт с 1 на списъка P-та и прибавим константа на всяко (по модул L), така че $P_0 = 0$. Така стигаме до $O(N^4)$.

8 Подзадачи 11 и 12: $N \leq 420$ и $N \leq 9000$

За тези две подзадачи се изискват $O(N^3)$ и $O(N^2)$ решения. Към горното решение има две възможни независими оптимизации, които свалят по една степен на N от сложността. Със само една от тях се хваща подзадача 11, а с и двете – подзадача 12.

8.1 $O(1)$ смятане на DP-то

Вместо за всеки десен край да итерираме през всеки възможен ляв край, можем да видим, че сметката (ако разделим левите краища по четности) е независима функция на десния и на левия край (и неговата DP стойност). Така с поддържане на помощен масив с оптимални леви краища за всяка бройка интервали и четност смятаме новата DP стойност в $O(1)$ и после ъпдейтваме помощния масив.

8.2 Премахване на N-те циклични шифта

Има два начина да направим това. Първият е чисто технически, без повече наблюдения. Нека да сметнем DP-то само веднъж (без циклични шифтове). Вместо да четем отговора само DP стейтове с последен десен край $N-1$, можем да използваме всички стейтове. Такъв с друг десен край R, ще довършим като разширим интервала на 0 назад до $R+1$. Това запазва броя интервали, но цената на това разширяване зависи от четността на интервала на 0, та и този бит трябва да участва в стейта.

Другият начин изисква грийди наблюдение: можем да разбием цикъла в най-голямата дупка между поредни P-та. Т.е. правим само един цикличен шифт, така че да сложим за 0 това i, което е след най-голямата (или една от няколко най-големи) дупка. Това наблюдение следва от пълното решение.

9 Изчистване на съображенията за четности

Това няма директно да свали сложността, но доста би изчистило логиката и кода за всяко от последните ни решения. Също така помага да открием и докажем пълното решение.

Нека вместо да смятаме отговора, да смятаме отговора по 2. Вместо да следим четността на броя четни интервали, това ще бъде включено в този удвоен отговор. Така тази двойна цената на интервал i става: $d(P_{Li}, P_{Ri}) + 2H + 3 + 2i$. Най-накрая е нужно просто да закръглим сумата нагоре при деление на 2. Възможно е и да изкараме другите членове извън DP-то, т.е. то само да следи $d(P_{Li}, P_{Ri})$, а най-накрая да прибавим $K(1 + 2H + 2K)$, където K е броят интервали.

При такова решение е изцяло ненужно да следим четности в DP стейта, което доста опростява и забързва кода. Освен това се оказва, че в $O(N^2)$ DP решението, не ни е нужно да следим самото DP, а само помощния масив.

10 Подзадача 13: $N \leq 10^5$

Тук се търси $O(N \log^2 N)$ решение. То е доста технично и по-сложно от пълното. Решението е да видим, че тази функция (след удвояването) има нужните хубави свойства, за да работи следното: ще правим търнъри сърч по това колко интервала са ни нужни за оптимално решение (защото коста намалява преди това и се увеличава след това). След това ще смятаме коста на разбиване на толкова интервали използвайки Aliens trick (защото кост функцията е изпъкнала). Няма да навлизаме в повече детайли, защото има по-добро и по-просто решение.

11 Подзадача 14: $N \leq 2 \times 10^6$

След изчистването на всички сметки, DP-то минимизира сумата на $d(P_{Li}, P_{Ri})$ (всъщност $P_{Ri} - P_{Li}$), а помощния ни масив максимизира сумата на $P_{Li} - P_{L_i - 1}$. Това обаче не изисква DP, а е просто грийди. Сортираме дупките (разликите между поредни P-та) и оптималното решение с K интервала е това което взема (т.е. разбива на) K-те най-големи дупки. Сложността е $O(N \log N)$.