АНАЛИЗ НА ЗАДАЧА
ТРИ

Решение за 20-30 точки

Ограничението $N \leq 15$ позволява да решим задачата чрез пълно изчерпване на всички възможности за разпределение на топчетата в трите групи. Всяко едно такова разпределение реално е вариация с повторение на N елемента от 3 опции т.е. имаме общо 3^N такива. Един лесен начин да ги получим, е да разгледаме представянията на числата от 0 до 3^{N-1} в троична бройна система – така всяка цифра посочва в коя група да поставим поредното топче. За всяко представяне трябва да проверим дали води изравняване на сумите в трите групи, и ако това е така, да преброим колко са топчетата, разпределени в група, различна от първоначалната им.

Сложност: $O(3^N \times N)$

Решение за 40-50 точки

Една модификация на предходното решение, която може да разшири с още няколко стойности диапазона на N, за които можем да решим задачата, е да генерираме разпледеленията рекурсивно. Така ще можем да поддържаме сумите по време на самото генериране и няма да се налага да "декодираме" разпределението накрая. Освен това във всеки един момент, когато сумата на топчетата в една от групите надхвърли $\frac{S}{3}$, може да прекратим изследването на разпределенията, които могат да бъдат получени текущото частично такова. Това до известна степен забързва решението, но ефектът е ограничен и концептуално не променя експоненциалната му сложност.

Сложност: $O(3^N)$

Решение за 70-80 точки

Оттук нататък е необходимо да приложим метода на динамичното програмиране. Трябва да се сетим за подходящ стейт т.е. няколко параметъра, които да дават достатъчно информация за задачата, така че да можем да я сведем до решаването на една или няколко по-лесни подзадачи (т.е. стейтове с по-малки параметри). Този стейт се явява своеобразно обобщение на повече от един конкретен вариант за разпределяне на топчетата, което премахва нуждата да разглеждаме всеки от тях по-отделно. В случая стейтът е сравнително интуитивен – налага се да пазим сумите на топчетата в трите групи. Нека разгледаме така дефинирания стейт $dp[s_1][s_2][s_3]$ – той не ни дава информация за това кои топчета са включени в тези три частични суми. Това е проблем, защото без значение от начина на изчисление на динамичното, ние не трябва да допускаме едно и също топче да бъде включено няколко пъти. Ето защо се налага да добавим още един параметър към стейта: $dp[i][s_1][s_2][s_3]$, описващ ситуацията, в която ще разпределим първите i топчета в трите групи, получавайки съответно суми s_1, s_2 и s_3 и за този стейт ще пазим минималния брой топчета, които ще бъдат сложени в група, различна от първоначалната.

Сега нека видим какви рекуренти зависимости са в сила и как да ги използваме при изчисляването на динамичното. Нека разгледаме стейта $dp[i][s_1][s_2][s_3]$ и възможните преходи, които ще получим, ако сведем задачата до разпределяне само на първите $i-1$ топчета. Имаме три варианта как може да сме поставили i-тото топче:

- ако сме го поставили в първата група, текущият стейт зависи само от $dp[i-1][s_1 - b[i]][s_2][s_3]$, защото сумите се запазват същите с изключение на първата група, която трябва да е била със стойността на i-тото топче по-малка; тогава:
  - ако i-тото топче първоначално се намира в първата група, стойността на $dp[i][s_1][s_2][s_3]$ трябва да е най-много $dp[i-1][s_1 - b[i]][s_2][s_3]$, защото поставяйки го в нея не увеличаваме броя на размените
  - ако i-тото топче първоначално се намира в друга група, стойността на $dp[i][s_1][s_2][s_3]$

трябва да е най-много $dp[i-1][s_1-b[i]][s_2][s_3]+1$, защото поставяйки го в нея увеличаваме броя на размените с 1

* когато казваме, че стойността на $dp[i][s_1][s_2][s_3]$ не може да е повече от тази на $dp[i-1][s_1-b[i]][s_2][s_3]$ (потенциално увеличена с единица), ние разчитаме на това, че винаги ще е възможно с някакъв брой размени да поставим първите i топчета в трите групи и да получим съответните суми, но това не винаги е така; затова, ако подобно разпределение не е възможно, можем да съхравяняваме някаква много голяма стойност INF, която да обозначава това и да сме сигурни, че никога няма да предпочетем този вариант.

- аналогично на първия случай, ако сме поставили i-тото топче във втората група: $dp[i][s_1][s_2][s_3]$ може да е най-много $dp[i][s_1][s_2-b[i]][s_3]+\mathbb{1}\{a[i]!=b[i]\}$, където обозначението $\mathbb{1}\{a[i]!=2\}$ приема стойност 1, ако условието в къдравите скоби е вярно, и 0 в противен случай
- ако сме го поставили в третата група: $dp[i][s_1][s_2][s_3]$ може да е най-много $dp[i][s_1][s_2][s_3-b[i]]+\mathbb{1}\{a[i]!=3\}$

* разглеждайки трите случая, трябва да вземем най-малката от трите получени стойности за броя на размените, или ако всички от тях са INF (т.е. подзадачите са нерешими), да поставим стойност INF в $dp[i][s_1][s_2][s_3]$

Въпросната динамична таблица се обхожда по начин, гарантиращ, че стейтовете, съответстващи на необходимите подзадачи за изчисляването на текущия, са вече изчислени. Един такъв ред е с цикли по четирите параметъра. Големината на таблицата, при ограниченията $N \leq 50$ и $S \leq 100$, е 50 млн. цели числа – 200 MB при използване на целочислен тип `int` или 100 MB при използване на 16-битовия му вариант `short`.

Сложност: $O(N \times S^3)$

Решение за 100 точки

Наблюдение: повечето от стейтовете имат стойност INF, защото задачата $dp[i][s_1][s_2][s_3]$ има решение само ако $s_1+s_2+s_3$ е равно на сумата от стойностите на първите i топчета. Тази вътрешна зависимост между параметрите на стейта ни позволява да премахнем едно от измеренията на динамичното и да поздразбираме s_3 като $(\sum_{j=1}^{i} b[j]) - s_1 - s_2$.

Това е напълно достатъчно да решим задачата за 100 точки, понеже такова решение се вмества в лимитите по време и памет (ако се използва `short` за съхранение на динамичната таблица). Все пак е възможно да оптимизираме допълнително паметта чрез стандартния трик – забелязваме, че стейтовете $dp[i][s_1][s_2]$ за всички стойности на s_1 и s_2 зависят само от стейтове, за които първото измерение има стойност $i-1$. Ето защо може да използваме само две таблици $dp_1[s_1][s_2]$ и $dp_2[s_1][s_2]$, в които да съхраняваме стейтовете само за текущото и предходното i.

Сложност: $O(N \times S^2)$

Коментари

Задачата е значително по-трудна от нормалното ниво на първи кръг на националната олимпиада по информатика. Тя беше предвидена главно с обучителна цел – състезателите в група С (особено учениците в 7. клас) да свикнат с приложението на техниката динамично програмиране с повече измерения. Затова и анализът е малко по-подробен.

Автор: Добрин Башев