Domáca úloha číslo 5

Náš algoritmus pracuje na tomto princípe:
Ak chceme vybrať k čísel z n zadaných tak, aby sa neopakovali 2 rovnaké šťastné čísla, vieme si rozdeliť všetky možnosti na $k+1$ prípadov podľa toho, koľko z vybraných čísel je šťastných. Teda ak zistíme počet možností ako vybrať j šťastných čísel a $k-j$ „ne-šťastných“ a sčítame všetky tieto počty pre $j \in \{0, \ldots, k\}$, zrejme dostaneme počet všetkých možností.

Počet možností, ako vybrať $k-j$ ne-šťastných čísel je celkom zrejme $\binom{n'}{k-j}$, kde n' je počet ne-šťastných čísel, lebo vyberáme hocijakú $(k-j)$-prvkovú podmnožinu indexov z množiny indexov všetkých ne-šťastných čísel (sedí aj prípad pre $k-j > n'$, kde sa hodnota binomického koeficientu definuje ako 0). Kombinačné čísla počítame od $\binom{n'}{0}$ s využitím tohto vzťahu:

$$\binom{n'}{i} = \frac{n'!}{(n'-i)!\,i!} = \frac{n'!}{(n'-i+1)!\,(i-1)!} \cdot \frac{n'-i+1}{i} = \binom{n'}{i-1} \cdot \frac{n'-i+1}{i}$$

Pretože $\binom{n'}{0} = 1$, môžeme postupne vypočítať každý potrebný koeficient. Všetky tieto hodnoty počítame modulo 10^9+7. Kvôli deleniu v modulárnej aritmetike potrebujeme na jeden koeficient logaritmický čas vzhľadom na veľkosť čísel od 0 do k, ktorých inverzy potrebujeme (rozšírený Euklidov algoritmus), teda kompletné spočítanie koeficientov bude potrebovať $O(k \log k)$ času (a $O(1)$ priestoru).

Teraz k počtu možností na výber j šťastných čísel: Urobíme to dynamickým programovaním. Najprv spočítame počet výskytov každého šťastného čísla (môžeme vstupný zoznam usporiadať a prejsť cifry každého čísla, dostaneme zoznam `luckies_count`, kde `luckies_count[i]` je počet výskytov i-teho šťastného čísla). Predstavme si teraz dvojrozmernú tabuľku `M`, kde v `M[i][j]` je odpoveď na otázku: Koľkými spôsobmi môžeme vybrať z prvých i šťastných čísel (usporiadaných podľa veľkosti) práve j tak, aby sa žiadne neopakovalo?
Zrejme pre j=0 je `M[i][j]`=1. Každú relevantnú hodnotu (kde $i \geq j$) vieme potom vypočítať ako

```
M[i][j] = M[i-1][j] + M[i-1][j-1] * luckies_count[i]
```

Ak máme totiž vybrať j čísel, máme možnosť vybrať ich z prvých $i-1$, alebo zobrať i-te (na to máme `luckies_count[i]` možností) a vybrať $j-1$ z predchádzajúcich (nemôže sa zobrať i-te druhý-krát). Takto vieme dynamickým programovaním vypočítať všetky možnosti výberu od 0 po k šťastných čísel. Potrebné hodnoty budú samozrejme v poslednom riadku, kde vyberáme zo všetkých šťastných čísel.
K zložitosti, šťastné čísla sa skladajú iba z dvoch cifier a všetky sú menšie ako 10^9, teda ich počet je rovný počtu všetkých od 1 po 9 znakov dlhých binárnych reťazcov: $\sum_{i=1}^{9} 2^i = 1022$. To je konštanta, ktorá nezávisí od vstupu, preto teoreticky časová aj priestorová zložitosť tejto časti algoritmu je $O(1)$, lebo počítame tabuľku od 0 po počet všetkých rôznych šťastných čísel (ostatné hodnoty sú teoreticky nuly). Prakticky je táto konštanta tiež v rámci medzí efektívnosti, preto, hoci sa v časovej zložitosti bude vyskytovať umocnená na druhú (kvôli vypĺňaniu dvojrozmernej tabuľky), nevplyvní zložitosť výpočtu toľko, ako ju môžu ovplyvniť vstupné parametre. Pretože potrebujeme v každom kroku dynamického programovania len aktuálny a predchádzajúci riadok, stačia na reprezentáciu v pamäti dva zoznamy.

Zhrnutie časovej a priestorovej zložitosti:
Algoritmus na začiatku načíta a usporiada vstupné pole, čo je $O(n \log n)$. Spočíta `luckies_count[i]`, na čo stačí takisto $O(n \log n)$ času (pre každé číslo zo vstupu logaritmicky prejde jeho dekadické cifry). Dynamickým programovaním spočíta tabuľku `M`, resp. jej posledný riadok (časovo ohraničené konštantou 1022^2). Spočíta binomické koeficienty pre ostatné („ne-šťastné“) čísla — to je $O(k \log k)$ času kvôli logaritmickému času Euklidovho algoritmu. Pretože $k \leq n$, celková časová zložitosť algoritmu je v $O(n \log n)$.
Čo sa týka priestorovej zložitosti, vstupné pole má n čísel, jediné ďalšie netriviálne veľké údaje sú všetky ohraničené počtom rôznych šťastných čísel vo vstupnom zozname, teda majú najviac 1022 prvkov. Teda celková priestorová zložitosť algoritmu je $O(n)$.