1 Popis řešení

Zavedeme si dvojrozměrné pole hodnot. Pole bude indexované hodnotami od $(0, 0)$ do (m, n), kde m je velikost jehly a n velikost sena. Na pozici (i, j) v poli bude uložena hodnota udávající, kolikrát se v prvních j znacích sena vyskytuje jako podposloupnost prvních i znaků jehly.

Ze začátku můžeme dosadit hodnotu 1 na pozice $(0, j)$ pro všechna j, protože prázdný řetězec je podposloupností libovolného řetězce. Také na všechny pozice $(i, 0)$ kromě $(0, 0)$ můžeme dosadit nulu, jelikož prázdný řetězec nemá neprázdné podřetězce. (Nulami by se dal vyplnit celý spodní trojúhelník matice, ovšem toto vyplyne z průběhu algoritmu.)

Poté budeme postupně zvětšovat i a doplňovat řádky matice pro zvětšující se j následujícím způsobem:

1. Pokud i-tý znak jehly neodpovídá j-tému znaku sena, na pozici (i, j) dosadíme hodnotu z pozice $(i, j - 1)$.
2. Pokud se i-tý znak jehly a j-tý znak sena rovnají, na pozici (i, j) dosadíme součet hodnot z $(i, j - 1)$ a $(i - 1, j - 1)$.

Po vyplnění celé matice je výsledná hodnota na pozici (m, n).

2 Důkaz správnosti

Algoritmus určitě doběhne, protože v každém kroku vyplníme dosud nevyplněný prvek pole, jež má konečnou velikost. Zároveň do prvků vždy dosazujeme definované hodnoty, protože při zpracování daného prvku jsme již všechny předchozí řádky a všechny hodnoty na tomto řádku zpracovali.

Pokud se na některé pozici sena nenachází poslední znak hledané části jehly, nemůže nám na této pozici vzrůst počet podposloupností této části, protože zde žádná podposloupnost nekončí. V takovém případě náš algoritmus korektně pouze opíše hodnotu z předchozí pozice.

Zbývá tedy rozmyslet, jak se změní počet podposloupností, pokud se na dané pozici nachází poslední znak dané části jehly. V tomto případě se v této části sena nachází dva druhy podposloupností:

1. Podposloupnosti, které nekončí posledním znakem. Ty nejsou nijak ovlivněny a jejich počet tedy můžeme získat z předchozí pozice řádku.
2. Podposloupnosti, které končí posledním znakem. Aby podposloupnost mohla na této pozici končit posledním znakem, musela být na předchozím znaku podposloupností o znak kratší jehly. Všechny takovéto posloupnosti získáme z předchozí pozice na předchozím řádku.

Jelikož algoritmus provádí přesně výše popsané operace, platí po doběhnutí naše definice hodnot v poli. Když tedy potřebujeme zjistit, kolikrát se celá jehla vyskytuje jako podposloupnost v celém seně, stačí se podívat na pozici (m, n).

3 Složitost

Máme pole o m řádcích a n sloupcích, v němž trávíme konstantní čas na každý prvek. Celková časová i prostorová složitost algoritmu je tedy $\Theta(m \cdot n)$