Szerkesztési távolság

A két karakterlánc közötti szerkesztési távolság (más néven Levenshtein távolság[1]) az egyik karakterlánc másikká alakításához szükséges műveletek (beszúrás, törlés vagy helyettesítés) minimális száma.

Például a "LOVE" és a "MOVIE" közötti szerkesztési távolság 2. A lehetséges műveletek a következők lehetnek:

Cseréljük ki az „L”-t „M”-re.

A „V” után illessze be az „I”-t.

Megközelítés dinamikus programozással:

A dinamikus programozás (DP) egy módszer a problémák megoldására úgy, hogy egyszerűbb részproblémákra bontjuk őket, és eltároljuk ezen részproblémák eredményeit a redundáns számítások elkerülése érdekében. Ez a következőképpen vonatkozik a szerkesztési távolság problémájára:

Részprobléma: A megoldandó részprobléma a következő: Mekkora a minimális szerkesztési távolság az s1 karakterlánc első i karaktere és az s2 karakterlánc első j karaktere között?

Legyen dp[i][j] az s1 első i karakterének az s2 első j karaktereivé alakításához szükséges műveletek minimális száma.

Alapesetek: Ha az egyik karakterlánc üres, az egyetlen lehetőség a másik karakterlánc összes karakterének hozzáadása vagy törlése. Így:

dp[i][0] = i (az s1 összes i karakterének törlése).

dp[0][j] = j (beszúrja az s2 összes j karakterét).

Ismétlődő kapcsolat:

Ha az s1[i-1] és s2[j-1] karakterek egyenlőek, nincs szükség szerkesztési műveletre, tehát:

- dp[i][j] = dp[i-1][j-1].

Ha a karakterek eltérőek, három lehetséges műveletünk van:

- Beszúrás: Adjunk hozzá egy karaktert az s1-hez (ami az s2-ben egy lépést visszalépésnek felel meg).
- Törlés: Távolítsunk el egy karaktert az s1-ből (ami az s1-ben egy lépéssel visszalépésnek felel meg).
- Csere: Cseréljünk ki egy karaktert az s1-ben, hogy megfeleljen az s2-nek.

Végső válasz:

A válasz dp[n][m]-ben lesz tárolva, ahol n és m a két karakterlánc hossza. Ez az érték a teljes s1 s2-vé alakításához szükséges műveletek minimális számát jelenti.

[1] https://en.wikipedia.org/wiki/Levenshtein_distance

Miért illik a dinamikus programozáshoz:

Optimális alstruktúra: A probléma kisebb részproblémákra bontható, ahol a teljes probléma megoldása a részproblémák megoldásától függ. Pontosabban, az s1 első i karakterének s2 első j karakterévé való konvertálásának megoldása a kisebb, kevesebb karaktert igénylő problémák megoldásától függ.

Átfedő részproblémák: Ugyanazok a részproblémák (a kisebb s1 és s2 előtagok szerkesztési távolsága) többször is megoldódnak. Az eredmények dp táblában való tárolásával elkerüljük a redundáns számításokat, így csökkentjük az időbonyolítást.