Urejevalna razdalja

Avtorji: Nino Cajnkar, Gregor Kikelj
Mentorica: Anja Petković

1 Motivacija

Tajnica v posadki MARS - a je pridna delavka, ampak se velikokrat zmoti. Na srečo piše v programu Microsoft Word, ki ji napačne besede popravi, oziroma ji ponudi besede, ki so podobne njeni nepravilno napisani besedi. Toda, kako lahko program ve, katero besedo smo mislili? Če si pri tem pomaga s slovarjem, ki ga uporabi kot množico vseh možnosti, lahko besedo razdeli na posamezne črke ter ji dodeli številko (razliko med napačno napisano besedo in besedo v slovarju, nato pa kot predlog poda tiste besede, ki se od napisane najmanj razlikujejo), nato pa črke, ki tvorijo besedo v pravilnem zaporedju primerja s slovarjem.

2 Urejevalna razdalja

Vprašamo se, kaj je merilo podobnosti med besedami. Zaslutimo, da nam razdalja med besedami pove, koliko operacij je potrebno, da iz ene besede dobimo drugo. Dovoljene operacije so odstranitev znaka, dodajanje znaka, in zamenjava znaka.

Definicija 1. *Abeceda* je poljubna končna množica znakov. *Niz* je zaporedje znakov iz abecede.

Besede v slovenščini so le kratki nizi, mi pa bomo računali razdaljo med poljubnimi nizi.

Definicija 2. *Urejevalna razdalja* je število, ki nam pove, najmanj koliko operacij potrebujemo, da iz enega niza dobimo drugega. Dovoljene operacije so:

- odstranitev znaka
- dodajanje znaka

- zamenjava znaka.

Cilj naše naloge je bil napisati program v Javi, ki bo določil urejevalno razdaljo med dvema nizoma.

3 Problem tramvaj, razvada

Za osnoven primer, ki smo ga rešili kar peš (s poskušanjem) smo vzeli besedi tramvaj in razvada. Ugotovili smo, da moramo narediti 4 poteze, da spremenimo besedo tramvaj v besedo razvada. Če odstranimo t iz tramvaja, dobimo kar štiri črke, ki se ujemajo na enakih mestih(RAmVAj, RAzVAda). Nato smo črko m iz besede ramvaj spremenili v črko z in smo dobili že pet enakih črk zaporedoma (RAZVAj, RAZVAda). Nato smo še črko j spremenili v črko d (RAZVAD, RAZVADa) in na koncu še dodali črko a. Dobili smo dve enaki besedi RAZVADA. Tako smo uganili, da je urejevalna razdalja med besedama TRAMVAJ in RAZVADA največ 4, ko bomo zagnali algoritem na tem primeru bomo s tem dokazali da je urejevalna razdalja natančno 4.

4 Algoritem

Računanja urejevalne razdalje se bomo lotili z algoritmom. Algoritem deluje na osnovi dvodimenzionamelne tabele in dinamičnega programiranja. Problema se bomo lotili tako, da ga bomo razdelili na manjše dele, s pomočjo katerih bomo potem rešitev počasi sestavili. Naj bosta a in b niza dolžin m in n. Označimo z $E(i, j)$ urejevalno razdaljo med podnizoma $a[0 : i]$ in $b[0 : j]$, to pomeni, da vzamemo le prvih i znakov niza a in prvih j znakov niza b. Denimo da poznamo $E(x, j)$ za vse $x < i$ ter $E(i, y)$ za vse $y < j$, ter $E(x, y)$ za $x < i, y < j$. Iz teh podatkov želimo izračunati $E(i, j)$. Izbiramo med tremi možnostmi:

1. Dodamo znak $b[j]$ na konec $a[0 : i]$
2. Izbrišemo znak
3. Zamenjamo znak ali ga preskočimo, če sta znaka enaka.

Če se odločimo za 1. možnost, bo število potrebnih operacij enako $E(i-1, j)+1$, saj naredimo eno dodatno operacijo (dodamo znak) ter toliko operacij, kot potrebujemo za izračun urejevalne razdalje med $a[i-1]$ in $b[j]$. Če se odločimo za 2. možnost, bo število potrebnih operacij enako $E(i, j-1)+1$, saj naredimo eno dodatno operacijo (odstranimo znak) ter toliko operacij, kot potrebujemo

za izračun urejevalne razdalje med $a[i-1]$ in $b[j]$. Če pa se odločimo za tretjo možnost, bo število potrebnih operacij enako $E(i-1,j-1)+\delta(a,b,i,j)$, kjer je

$$\delta(a,b,i,j) = \begin{cases} 0; & \text{če } a[i] = b[j] \\ 1; & \text{če } a[i] \neq b[j]. \end{cases}$$

Ta δ nam pove, da porabimo operacijo, če sta znaka različna, če pa sta enaka potem jih ni potrebno zamenjati in ne porabimo poteze. Izberemo najmnjšo od teh možnosti, torej

$$E(i,j) = \min\{E(i-1,j)+1, E(i,j-1)+1, E(i-1,j-1)+\delta(a,b,i,j)\}.$$

Kaj pa so naši začetni pogoji? Če imamo prazen niz in ga hočemo prevesti na nepraznega, moramo vse črke dodati, torej lahko sklepamo da je $E(i,0) = i$ in $E(0,j) = j$. Sedaj naredimo tabelo E velikosti $m \times n$, kjer bomo zračunali urejevalno razdaljo. Število $E(i,j)$ bomo zapisali v tabelo na mesto (i,j). Prvo vrstico in prvi stolpec tabele izpolnimo z začetnimi pogoji (primer vidimo na tabeli 4).

Tabela 1: Tabela z začetnimi pogoji

0	0	t	r	a	m	v	a	j
0	0	1	2	3	4	5	6	7
r	1							
a	2							
z	3							
v	4							
a	5							
d	6							
a	7							

Če si izberemo neko polje, ga lahko izpolnimo, če vemo vrednosti polj, ki so nad njim, levo od njega in diagonalno zgoraj levo od njega ($E[i-1][j]$, $E[i][j-1]$, $E[i-1][j-1]$). To naredimo s formulo, ki smo jo že prej napisali

$$E(i,j) = \min\{E(i-1,j)+1, E(i,j-1)+1, E(i-1,j-1)+\delta(a,b,i,j)\}.$$

Ko tabelo tako izpolnimo, dobimo tabelo 4.

To je bil ročni postopek, ki pa ga hitreje naredi program zapisan v dodatku.

Tabela 2: Dokončana tabela

0	0	t	r	a	m	v	a	j
0	0	1	2	3	4	5	6	7
r	1	1	1	2	3	4	5	6
a	2	2	2	1	2	3	4	5
z	3	3	3	2	2	3	4	5
v	4	4	4	3	3	2	3	4
a	5	5	5	4	4	3	2	3
d	6	6	6	5	5	4	3	3
a	7	7	7	6	6	5	4	4

5 Analiza algoritma

Ko ocenjujemo algoritem, merimo, koliko časa se program izvaja. Tega ne merimo v sekundah, ampak v številu aritmetičnih operacij. Število uporabljenih operacij želimo izraziti kot funkcijo velikosti vhodnih podatkov. Natančneje, zanima nas red naraščanja števila operacij, ko večamo vhodni podatek.

Definicija 3. Naj bosta $f, g : \mathbb{R} \to \mathbb{R}$ funkciji realnih števil. Tedaj pišemo $f(x) = O(g(x))$, če obstaja taka pozitivna konstanta M in $x_0 \in \mathbb{R}$, da velja $\forall x \geq x_0 : \ f(x) \leq M \cdot g(x)$

Ta definicija pomeni, da bo imela funkcija $M \cdot g(x)$ za velike x (asimptotično) večje vrednosti kot funkcija $f(x)$. Naš algoritem deluje v času $O(m \cdot n)$, kjer sta m in n dolžini vhodnih nizov, namreč imamo tabelo velikosti $m \times n$, izpolniti moramo $m \cdot n$ polj, vsako izpolnjevanje pa vzame konstantno število opreacij. To skupaj združimo v $O(m \cdot n)$.

6 Metrični prostori

Metrični prostori so v matematiki zelo pomembni in imajo veliko lepih lastnosti, zaradi česar smo tudi želeli dokazati, da je urejevalna razdalja metrika na množici vseh nizov. Več o metričnih prostorih si lahko preberete v knjigi [2] in v članku [3].

Definicija 4. Naj bo M neprazna množica in $d : M \times M \to \mathbb{R}$ preslikava. Par (M, d) je metrični prostor, če velja:

- $d(x, y) \geq 0 \ in \ d(x, y) = 0 \iff x = y$ (pozitivna definitnost)
- $d(x, y) = d(x, y)$ (simetričnost)

- $d(x, z) \leq d(x, y) + d(y, z)$ (trikotniška neenakost).

Sedaj, ko smo definirali metrične prostore, lahko zapišemo izrek o urejevalni razdalji.

Izrek 1. *Naj bo M množica vseh nizov nad končno abecedo in d urejevalna razdalja. Tedaj je par (M, d) metrični prostor.*

Dokaz. Preveriti moramo pogoje za metrični prostor:

- simetričnost je očitna, ker lahko prvi niz pretvorimo v drugega z analognimi operacijami kot drugega v prvega.
- pozitivna definitnost: Če sta niza enaka, potem ni treba nič spremeniti in je razdalja enaka nič, drugače, če nista enaka, pa moramo nekaj spremeniti in je večja od nič.
- trikotniška neenakost: Denimo, da imammo tri nize a, b in c. Število potrebnih operacij, da prevedemo niz a v b je gotovo manjše ali enako kot če najprej prevedemo a v c in potem c v b. S tem smo pokazali, da je (M, d) res metrični prostor.

□

7 Zaključek

Ugotovili smo torej, kako lahko program pomaga naši MARS - ovski tajnici pri tipkanju. Napisali smo tudi učinkovit program, ki urejevalno razdaljo izračuna. Dokazali smo, da je urejevalna razdalja metrika na množici vseh nizov, saj ustreza vsem trem potrebnim pogojem.

8 Dodatek – program

Napisalo smo tudi program v Javi, ki izračuna urejevalno razdaljo med nizoma.

```
//naredimo delta funkcijo
public static int delta(char[] a,char[] b,int i,int j){
        if(a[i-1]==b[j-1]){
            return 0;
        }
```

```
        return 1;
    }
// napišemo funkcijo, ki izracuna urejevalno razdaljo.
// Vhodni podatek sta dva niza.
    public static int urejevalnaRazdalja(String aa,String bb){

        int m=aa.length(); // z m označimo dolžino prvega niza
        int n=bb.length(); // z n označimo dolžino drugega niza
        // preverimo, če je en niz morda prazen, saj takrat je urejevalna
        // razdalja kar dolžina drugega niza
        int o=Math.max(m,n);
        if(m==0||n==0){
            return o;
        }
        //spremenimo niza v seznama znakov
        char[] a=new char[m];
        char[] b=new char[n];
        for(int i=0;i<m;i++){
            a[i]=aa.charAt(i);
        }
        for(int j=0;i<n;j++){
            b[j]=bb.charAt(j);
        }
        // naredimo tabelo velikosti m x n
        int[][] E=new int[m+1][n+1];
        //vstavimo začetne pogoje v tabelo
        for(int i=0;i<=n;i++){
            E[0][i]=i;
        }
        for(int j=1;i<=m;j++){
            E[j][0]=j;
        }
        //napolnimo tabelo kot v predpisu
        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                E[i][j]=Math.min(Math.min(E[i-1][j]+1,
                            E[i][j-1]+1),E[i-1][j-1]+delta(a,b,i,j));
            }
        }
        //funkcija vrne (m,n)-ti element tabele
        return E[m][n];
```

```
}
```

Literatura

[1] Jurafsky, D.: Minimum edit distance; Definition of minimum edit distance: 2015, Stanford; stran 1 - 36 (https://web.stanford.edu/class/cs124/lec/med.pdf)

[2] Vrabec, J.: Metrični prostori:1. izdaja. Ljubljana: DMFA Slovenije, 1990;

[3] Maša Smajila, Ana Štuhec, Nejc Zajc, Neža Žager Korenjak: Banachovo skrčitveno načelo, MARSovski projekt 2015