Mintaillesztés

Kiegészítő anyag az Algoritmuselmélet tárgyhoz IV.

(a Rónyai–Ivanyos–Szabó: Algoritmusok könyv mellé)

Friedl Katalin
BME SZIT
friedl@cs.bme.hu

2017. február 1.

Alapfeladat: egy adott szövegrészt (mintát) keresünk egy adott szövegben. Pl. ez lehet egy szövegszerkesztő feladata, vagy csak úgy keresünk egy fájlban, vagy akár genetikai kód megfelelő darabját keressük, stb.

1. Alapfogalmak, jelölések

Σ jelöljön egy véges jelkészletet (ábécé). Ez lehet pl. az angol ábécé, a számjegyek, a bitek, vagy pl. A, G, C, T a genetikában. Ennek elemeit *betűk*nek vagy *karakterek*nek fogjuk hívni.

A Σ elemeiből képzett véges (!) hosszú sorozatok a *szavak*.

Az összes Σ feletti szó halmazának jele Σ^*.

A keresett *minta* egy $m > 1$ hosszú szó, azaz egy $M[1]M[2]\dots M[m]$ betűsorozat, ahol $M[i] \in \Sigma$.

Hasonlóan, a *szöveg* egy $n \geq m$ hosszú szó, azaz egy $S[1]S[2]\dots S[n]$ betűsorozat, ahol $S[i] \in \Sigma$.

Azt mondjuk, hogy az M minta k *eltolással előfordul* az S szövegben ($k \geq 0$), ha $M[j] = S[k+j]$ teljesül minden $j = 1, \dots, m$ esetén, vagy rövidebben: $M = S[k+1..k+m]$.

Az M *előfordul* az S szövegben, ha van olyan $0 \leq k \leq n-m$, melyre k eltolással előfordul.

Más szavakkal, az S szó felbontható 3 részre, $S = UMV$, ahol U vagy V hossza akár 0 is lehet. (U hossza $|U| = k$ és V hossza, $|V| = n-k-m$.)

Többféle természetes kérdést is feltehetünk:

- előfordul M az S szövegben?
- hol van M első előfordulása az S szövegben?
- hol fordul elő M az S szövegben? (Az összes előfordulásra kíváncsiak vagyunk.)

Mindegyik típusra sokféle eljárás ismert. Az hogy melyiket érdemes használni több dologtól függhet

- mekkora a Σ ábécé;
- mekkora n és m;
- ugyanazt a mintát egyszer vagy sokszor keressük különböző szövegekben;
- ugyanabban a szövegben többször is keresünk-e különböző mintákat.

Nagy irodalma van annak, hogy mikor milyen eljárást lehet célszerű használni ahhoz, hogy valóban gyors eljárást kapjunk. Mi most csak egy pár egyszerű és könnyen elemezhető eljárást nézünk meg, számos további található például Christian Charras és Thierry Lecroq: Exact String Matching Algorithms című könyvében (`http://www-igm.univ-mlv.fr/~lecroq/string`).

2. Egyszerű algoritmus

Sorban, a $k = 0, 1, 2, \ldots, n - m$ értékekre ellenőrzi, hogy a minta k eltolással előfordul-e a szövegben. Egy ilyen ellenőrzés az $M[j] \stackrel{?}{=} S[k + j]$ összehasonlításokból áll, ahol $j = 1, 2, \ldots, m$ (vagy abbahagyhatjuk, ha a két karakter eltér egymástól).

Világos, hogy ezzel az eljárással megtaláljuk az összes előfordulást (és ha csak az elsőre, vagy arra lennénk kíváncsiak, szerepel-e egyáltalán a minta a szövegben, akkor az első előfordulásnál leállhatunk).

Összehasonlítások száma

Minden k értékre legfeljebb m összehasonlítást végzünk, így az összehasonlítások száma $(n - m + 1) \cdot m$, ami $O(nm)$.

1. Példa. *Ha S csupa* `a` *betűből áll, M pedig $m - 1$ darab* `a` *betűből és a végén egy* `b`*-ből, akkor ez az algoritmus ténylegesen használ is $(n - m + 1) \cdot m$ összehasonlítást.*

2. Példa. *Ha S csupa* `a` *betűből áll, M pedig csupa* `b` *betűből, akkor minden illesztési kísérlet egy összehasonlítást használ, ilyenkor összesen $n - m + 1$ összehasonlítás elég.*

3. Gyorskeresés (Quick Search)

Az ötlet az, hogy a k eltolásos illesztés ellenőrzése után a következő illesztésben az $S[k+m+1]$ karakter biztos szerepel majd, tehát csak olyan eltolással érdemes próbálkozni, ahol ehhez a helyhez a mintában egy igyanezt a karaktert tartalmazó pozíció illeszkedik.

Előfeldolgozás – az ugrófüggvény

Az algoritmus előkészítő szakaszában csak a minta alapján meghatározunk egy *ugrófüggvényt*. Ezt megadhatjuk egy U tömbben, mely a Σ ábécé elemeivel van indexelve, $U[x]$ értéke legyen az, hogy az $x \in \Sigma$ hátulról nézve hányadik helyen fordul elő először a mintában, és legyen $m+1$ ha x nem szerepel a mintában.

Az U gyorsan kitölthető a minta ismeretében:

1. először legyen U minden értéke $m+1$
2. sorban $i = 1, 2, \ldots, m$ esetén legyen $U[M[i]] = m + 1 - i$

3. Példa. *Ha a minta a GCAGAGAG sorozat, és az ábécé* $\Sigma = \{A, C, G, T\}$*, akkor* $m = 8$*, és a végeredmény* $U[A] = 2$*,* $U[C] = 7$*,* $U[G] = 1$*,* $U[T] = 9$*.*

Az algoritmus

A $k = 0$ eltolással kezdünk. Mindig, amikor új eltolás jönne, $k+1$ helyett $k + U[S[k+m+1]]$ következik.

4. Példa (folytatás). *A szöveg legyen GCATCGCAGAGAGTATACAGTACG* *(*$n = 24$*)*

G	C	A	T	C	G	C	A	G	A	G	A	G	T	A	T	A	C	A	G	T	A	C	G	
G	**C**	**A**	**G**	A	G	A	G	•																$U[G] = 1$
	G	C	A	G	A	G	A	G	•															$U[A] = 2$
			G	C	A	G	A	G	A	G	•													$U[A] = 2$
					G	**C**	**A**	**G**	**A**	**G**	**A**	**G**	•											$U[T] = 9$
														G	C	A	G	A	G	A	G	•		$U[C] = 7$

Az utolsó eltolás már kilógna a szövegből, az algoritmus itt véget ér. Eredmény: a keresett minta egyszer fordul elő (a 4. eltolásnál).
Az algoritmus 5 eltolást próbált ki a lehetséges 17 helyett.
Az elvégzett összehasonlítások száma: $4 + 1 + 1 + 8 + 1 = 15$

Az algoritmus helyes, hiszen minden alkalommal olyan ugrást végzünk, aminél kisebb biztos nem járna eredménnyel, mert kisebb ugrás esetén valahol a minta és a szöveg megfelelő karaktere eltér, ha máshol nem, az algoritmusunkban az ugrást meghatározó pozíciónál.

Összehasonlítások száma

Az előfeldolgozás ideje $O(m + |\Sigma|)$, ami, mivel az ábécé mérete konstans $O(m)$. Magára az algoritmusra most sem garantálhatunk kevesebb összehasonlítást, mint az egyszerű algoritmusnál, ez a rész $O(nm)$.

Láthatjuk, hogy a lépésszám becslése nem javult (sőt romlott egy kicsit az előfeldolgozással), de a gyakorlatban ez az eljárás mégis többnyire jól viselkedik. Például, míg az egyszerű algoritmus mindig legalább $n - m + 1$ összehasonlítást használ, addig a gyorskeresésnek van, hogy elég $n/(m + 1)$. (Miért?)

1. Megjegyzés. *Vannak olyan algoritmusok is, amelyek a már illesztett karakterek során összegyűjtött információkat is használják, ezzel csökkentve a sikertelen eltolások számát. Igazából a gyorskeresés (Sunday, 1990) egy ilyen általánosabb módszer (Boyer–Moore-algoritmus, 1977) egyszerűsítéseként keletkezett, és vált népszerűvé.*

2. Megjegyzés. *Van olyan algoritmus is (Knuth-Morris-Pratt, 1977), amelynek a lépésszáma lineáris, azaz $O(n + m)$. A részletek megtalálhatók a már említett helyen vagy a Rónyai–Ivanyos–Szabó: Algoritmusok c. könyvben.*

4. Véges automatás megoldás

Egy kínálkozó eszköz a véges automaták (állapotgépek) használata. Az ötlet, hogy az előkészítő fázisban a minta alapján definiálunk egy automatát, aminek segítségével a szöveggel való illeszkedések már $O(n)$ lépésben meghatározhatók. Az automata elkészítése, tárolása $O(m|\Sigma|)$.

Az informatikusok képzésében ez az ötlet elő szokott kerülni Digiten, és egy kicsit bonyolultabb változatban Prog1-en is (ly számláló). De a részletek előtt előbb definiáljuk, mi is az a véges automata ...