

Reiknirit Knuths, Morrisar og Pratts (1970)

Bergur Snorrason

April 8, 2024

- ▶ Gefum okkur langan streng s og styttri streng p .
- ▶ Hvernig getum við fundið alla hlutstrengi s sem eru jafnir p .
- ▶ Fyrsta sem manni dettur í hug er að bera p saman við alla hlutstrengi s af sömu lengd og p .

```

5 void frumstaed_strengjaleit(char* s, int n, char* p, int m, int *r)
6 {
7     int i, j;
8     for (i = 0; i < n; i++) r[i] = 0;
9     for (i = 0; i < n - m + 1; i++)
10    {
11        for (j = 0; j < m; j++) if (s[i + j] != p[j]) break;
12        if (j >= m) r[i] = 1;
13    }
14 }

```

- ▶ Gerum ráð fyrir að strengurinn s sé af lengd n og strengurinn p sé af lengd m .
- ▶ Fjöldi hlutstrengja í s að lengd m er $n - m + 1$.
- ▶ Strengjasamanburðurinn tekur línulegan tíma.
- ▶ Svo tímaflækja leitarinnar er $\mathcal{O}(nm - m^2)$.
- ▶ Ef $m = n/2$ þá er $nm - m^2 = n^2/2 - n^2/4 = n^2/4$ tímaflækjan er í raun $\mathcal{O}(n^2)$.
- ▶ Dæmi um leiðinlega strengi væri $s = „aaaaaaaaaaaaaaaaa”$ og $p = „aaaaaab”$.
- ▶ Þessi aðferð virkar þó sæmilega ef strengirnir eru nógu óreglulegir.
- ▶ Dæmi um það hvenær þessi aðferð er góð er ef maður er að leita að orði í skáldsögu.

- ▶ Aðferðin er líka nógu góð ef $\mathcal{O}(n^2)$ er ekki of hægt.
- ▶ Það er þó óþarfi að útfæra hana því hún fylgir með flestum forritunarmálum, til dæmis:
 - ▶ Í `string.h` í C er `strstr(..)`.
 - ▶ Í `string` í C++ er `find(..)`.
 - ▶ Í `String` í Java er `indexOf(..)`.
- ▶ Munið bara að ef $n > 10^4$ er þetta yfirleitt of hægt.

- ▶ Við getum bætt tímaflækjun með því að nota svokallað *forstrengsfall* (e. *prefix function*) strengsins a .
- ▶ Forstrengsfallið f er gefið með
$$f(j) = \max\{k \in \mathbb{N} : k < |a|, a[1, k] = a[j - k + 1, j]\}.$$
- ▶ Sjáum fyrst að þetta fall uppfyllir $f(j + 1) \leq f(j) + 1$.
- ▶ Látum $k = f(j)$ og sjáum að ef $a[j + 1] = a[k]$ þá er $f(j + 1) = k + 1$.
- ▶ Ef $a[j + 1] \neq a[k]$ þá þurfum við að minnka k þangað til við fáum jöfnuð.
- ▶ Við minnkum k með því að láta $k' = f(k - 1)$.

```
12 void prefix_function(char *a, int *b)
13 {
14     int i, j, m = strlen(a);
15     for (i = 0, j = b[0] = -1; i < m; b[++i] = ++j)
16         while (j >= 0 && a[i] != a[j]) j = b[j];
17 }
```

- ▶ Takið eftir að hver ítrun innri lykkjanna svarar til einnar ítrunar ytri lykkjanna.
- ▶ Svo innri lykkjan keyrir, í heildina, ekki oftari en ytri lykkjan.
- ▶ Því er tímaflækjan í heildina $\mathcal{O}(n)$ fyrir streng af lengd n .

- ▶ En hvernig getum við notað forstrengsfallið til að framkvæma strengjaleit?
- ▶ Gerum ráð fyrir að við séum að leit að streng p í stærri streng s .
- ▶ Látum þá $a = p + \alpha + s$, þar sem $+$ táknar samskeytingu strengja og α er stafur sem er hvorki í p né s .
- ▶ Við látum svo f vera forstrengsfall strengsins a .
- ▶ Við höfum þá að $f \leq |p|$ og $f(j) = |p|$ þá og því aðeins að $j > |p|$ og hlutstrengurinn í s sem byrjar á vísi $j - |p| - 1$ og er $|p|$ af lengd er sá sami og p .

```

12 void prefix_function(char *a, int *b)
13 {
14     int i, j, m = strlen(a);
15     for (i = 0, j = b[0] = -1; i < m; b[++i] = ++j)
16         while (j >= 0 && a[i] != a[j]) j = b[j];
17 }
18
19 void kmp(char *s, char *p, int *r)
20 {
21     int i, j, n = strlen(s), m = strlen(p), b[n + m + 2];
22     char a[n + m + 2];
23     strcpy(a, p), strcat(a, "\n"), strcat(a, s);
24     prefix_function(a, b);
25     for (i = 0; i < n; i++) r[i] = i < n - m + 1 && b[i + 2*m + 1] >= m;
26 }

```

- ▶ Gerum ráð fyrir að $|s| = n$ og $|p| = m$.
- ▶ Þá er $|a| = n + m + 1$, svo tímaflækjan er $\mathcal{O}(n + m)$.

