Reiknirit Knuths, Morrisar og Pratts (KMP 1970)

Bergur Snorrason

4. apríl 2022

Strengjaleit

- Gefum okkur langan streng s og styttri streng p.

Strengjaleit

- Gefum okkur langan streng s og styttri streng p.
- Hvernig getum við fundið alla hlutstrengi s sem eru jafnir p.

Strengjaleit

- 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.

```c
void naive(char* s, int n, char* p, int m, int *r)
{
    int i, j;
    for (i = 0; i < n; i++) r[i] = 0;
    for (i = 0; i < n - m + 1; i++)
    {
        for (j = 0; j < m; j++)  if (s[i + j] != p[j]) break;
        if (j >= m) r[i] = 1;
    }
}
```

- Gerum ráð fyrir að strengurinn s sé af lengd n og strengurinn p sé af lengd m.

- 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 $n - m + 1$.

- 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.

- 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 $\mathcal{O}(\qquad\qquad)$.

- 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 $\mathcal{O}(nm - m^2)$.

- 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 $\mathcal{O}(\quad)$.

- 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 $\mathcal{O}(n^2)$.

- 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 $s =$ „$aaaaaaaaaaaaaaaa$" og $p =$ „$aaaaaaab$".

- 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 $s =$ „$aaaaaaaaaaaaaaaa$" og $p =$ „$aaaaaaab$".
- Þessi aðferð virkar þó sæmilega ef strengirnir eru nógu óreglulegir.

- 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 $s =$ „$aaaaaaaaaaaaaaaa$" og $p =$ „$aaaaaaab$".
- 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.

- 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:

- 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(..)`.

- 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(..)`.

- 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(..)`.

- 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.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*”.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*".
- Ef strengjasamanburðurinn misheppnast í p_3 þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*".
- Ef strengjasamanburðurinn misheppnast í p_3 þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að fyrstu þrír stafnirnir í næsta hlutstreng stemma, svo við getum byrjað í p_2.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*".
- Ef strengjasamanburðurinn misheppnast í p_3 þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að fyrstu þrír stafnirnir í næsta hlutstreng stemma, svo við getum byrjað í p_2.
- Reiknirit Knuths, Morrisar og Pratts notar sér þessa hugmynd til að framkvæma strengjaleit.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*".
- Ef strengjasamanburðurinn misheppnast í p_3 þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að fyrstu þrír stafnirnir í næsta hlutstreng stemma, svo við getum byrjað í p_2.
- Reiknirit Knuths, Morrisar og Pratts notar sér þessa hugmynd til að framkvæma strengjaleit.
- Reikniritið byrjar á að forreikna hversu mikið maður veit eftir misheppnaðan samanburð.

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ „*aaaabbbb*".
- Ef strengjasamanburðurinn misheppnast í p_3 þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að fyrstu þrír stafnirnir í næsta hlutstreng stemma, svo við getum byrjað í p_2.
- Reiknirit Knuths, Morrisar og Pratts notar sér þessa hugmynd til að framkvæma strengjaleit.
- Reikniritið byrjar á að forreikna hversu mikið maður veit eftir misheppnaðan samanburð.
- Svo þurfum við einfaldlega að labba í gegnum s og hliðra eins og á við.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k : s[1,k] = s[j-k+1, j]\}$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k\colon s[1,k] = s[j-k+1,j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j+1) \leq f(j) + 1$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k \colon s[1, k] = s[j - k + 1, j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j + 1) \leq f(j) + 1$.
- Látum $k = f(j)$ og sjáum að ef $s[j + 1] = s[k]$ þá er $f(j + 1) = k + 1$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k \colon s[1,k] = s[j-k+1,j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j+1) \leq f(j) + 1$.
- Látum $k = f(j)$ og sjáum að ef $s[j+1] = s[k]$ þá er $f(j+1) = k+1$.
- Ef $s[j+1] \neq s[k]$ þá þurfum við að minnka k þangað til við fáum jöfnuð.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k \colon s[1,k] = s[j-k+1,j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j+1) \leq f(j) + 1$.
- Látum $k = f(j)$ og sjáum að ef $s[j+1] = s[k]$ þá er $f(j+1) = k+1$.
- Ef $s[j+1] \neq s[k]$ þá þurfum við að minnka k þangað til við fáum jöfnuð.
- Við minnkum k með því að láta $k' = f(k-1)$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k : s[1, k] = s[j - k + 1, j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j + 1) \leq f(j) + 1$.
- Látum $k = f(j)$ og sjáum að ef $s[j + 1] = s[k]$ þá er $f(j + 1) = k + 1$.
- Ef $s[j + 1] \neq s[k]$ þá þurfum við að minnka k þangað til við fáum jöfnuð.
- Við minnkum k með því að láta $k' = f(k - 1)$.
- Það tekur $\mathcal{O}(\quad)$ tíma að reikna öll þessi gildi, því $f(j + 1) \leq f(j) + 1$, svo við munum ekki þurfa að minnka k oftar en n sinnum.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengsfall* (e. *prefix function*) strengsins p.
- Við látum $f(j)$, $1 \leq j \leq |p|$, vera gefið með $f(j) = \max\{k \colon s[1, k] = s[j - k + 1, j]\}$.
- Sjáum fyrst að þetta fall uppfyllir $f(j + 1) \leq f(j) + 1$.
- Látum $k = f(j)$ og sjáum að ef $s[j + 1] = s[k]$ þá er $f(j + 1) = k + 1$.
- Ef $s[j + 1] \neq s[k]$ þá þurfum við að minnka k þangað til við fáum jöfnuð.
- Við minnkum k með því að láta $k' = f(k - 1)$.
- Það tekur $\mathcal{O}(\, n\,)$ tíma að reikna öll þessi gildi, því $f(j + 1) \leq f(j) + 1$, svo við munum ekki þurfa að minnka k oftar en n sinnum.

```c
void prefix_function(char *p, int *b)
{ // Reiknar forstrengsfall p og geymir gildin í b.
    int i, j, m = strlen(p);
    for (i = 0, j = b[0] = -1; i < m; b[++i] = ++j)
        while (j >= 0 && p[i] != p[j]) j = b[j];
}

void kmp(char *s, char *p, int *r)
{ // Eftir á segir r[i] hvort i-ta hlutstrengur s sé sá sami og p.
    int i, j, n = strlen(s), m = strlen(p), b[m + 1];
    prefix_function(p, b);
    for (i = 0; i < n; i++) r[i] = 0;
    for (i = j = 0; i < n; )
    {
        while (j >= 0 && s[i] != p[j]) j = b[j];
        i++, j++;
        if (j == m) r[i - j] = 1, j = b[j];
    }
}
```

- Takið eftir að hver ítrun innri lykkjanna svarar til einnar ítrunar ytri lykkjanna.

- Takið eftir að hver ítrun innri lykkjanna svarar til einnar ítrunar ytri lykkjanna.
- Svo innri lykkjan keyrir, í heildina, ekki oftar en ytri lykkjan.

- Takið eftir að hver ítrun innri lykkjanna svarar til einnar ítrunar ytri lykkjanna.
- Svo innri lykkjan keyrir, í heildina, ekki oftar en ytri lykkjan.
- Því er tímaflækjan í heildina $\mathcal{O}(\qquad)$.

- Takið eftir að hver ítrun innri lykkjanna svarar til einnar ítrunar ytri lykkjanna.
- Svo innri lykkjan keyrir, í heildina, ekki oftar en ytri lykkjan.
- Því er tímaflækjan í heildina $\mathcal{O}(n + m)$.

Reiknirit Ahos og Corasicks (1975)

- Til er önnur aðferð, svipuð og `KMP`, sem finnur staðsetningar margra orða í einu í streng.

Reiknirit Ahos og Corasicks (1975)

- Til er önnur aðferð, svipuð og `KMP`, sem finnur staðsetningar margra orða í einu í streng.
- Hún er kennd við Aho og Corasick.

Reiknirit Ahos og Corasicks (1975)

- Til er önnur aðferð, svipuð og `KMP`, sem finnur staðsetningar margra orða í einu í streng.
- Hún er kennd við Aho og Corasick.
- Ég fer ekki í hana hér en hún byggir á því að gera *forstrengstré* (e. *prefix tree*), stundum kallað *trie*, og nota kvika bestun til að finna *bakstrengs hlekk* (e. *suffix link*) fyrir hvern hnút.

Reiknirit Ahos og Corasicks (1975)

- Til er önnur aðferð, svipuð og `KMP`, sem finnur staðsetningar margra orða í einu í streng.
- Hún er kennd við Aho og Corasick.
- Ég fer ekki í hana hér en hún byggir á því að gera *forstrengstré* (e. *prefix tree*), stundum kallað *trie*, og nota kvika bestun til að finna *bakstrengs hlekk* (e. *suffix link*) fyrir hvern hnút.
- Reikniritið keyrir í línulegum tíma í lengd allra strengjanna, ásamt fjölda heppnaðra samanburða, að því gefnu að stafrófið sé takmarkað.