Samansóp

Bergur Snorrason

3. apríl 2022

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.
- Mér lýst best á að hafa keppnina í miðvikudagstímanum okkur.

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.
- Mér lýst best á að hafa keppnina í miðvikudagstímanum okkur.
- Við höfum þá stoðtíma í mánudagstímanum.

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.
- Mér lýst best á að hafa keppnina í miðvikudagstímanum okkur.
- Við höfum þá stoðtíma í mánudagstímanum.
- Í keppninni verða fimm dæmi.

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.
- Mér lýst best á að hafa keppnina í miðvikudagstímanum okkur.
- Við höfum þá stoðtíma í mánudagstímanum.
- Í keppninni verða fimm dæmi.
- Skil fást fyrir að leysa eitt dæmi.

Lokakeppnin

- Í næstu viku verður lokakeppnin, og þar með síðasti tíminn í þessu námskeiði.
- Mér lýst best á að hafa keppnina í miðvikudagstímanum okkur.
- Við höfum þá stoðtíma í mánudagstímanum.
- Í keppninni verða fimm dæmi.
- Skil fást fyrir að leysa eitt dæmi.
- Ef þið leysið þrjú þeirra fáið þið aukaskil.

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 - j + 1] = 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 $p =$ „$aaaaaab$".

- 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 goð ef $\mathcal{O}(n^2)$ er ekki of hægt.

- Aðferðin er líka nógu goð 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 goð 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 goð 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 goð 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.

Reiknirit Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

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

Reiknirit Knuth, Morrisar og Pratts (`KMP`) strengjaleit (1970)

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

Reiknirit Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

- 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.

Reiknirit Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

- 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 Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

- 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.

Reiknirit Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

- 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.

Reiknirit Knuth, Morrisar og Pratts (KMP) strengjaleit (1970)

- 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 *forstrengs fall* (e. *prefix function*) strengsins p.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengs fall* (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 *forstrengs fall* (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 $f(j + 1) \leq f(j) + 1$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengs fall* (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 $f(j+1) = k + 1$.

- Til að finna hversu mikið á að hliðra hverju sinni þurfum við að reikna *forstrengs fall* (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 *forstrengs fall* (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 *forstrengs fall* (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 *forstrengs fall* (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)
{ // r[i] segir 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 $\mathcal{O}($ $)$.

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

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

- 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 *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 *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.

Hlaupabil

- Aðferð hlaupabila (e. sliding window) er stundum hægt að nota til að taka dæmi sem hafa augljósa $\mathcal{O}(n^2)$ og gera þau $\mathcal{O}(n)$ eða $\mathcal{O}(n \log n)$.

- Skoðum dæmi:

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n\in\mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n \in \mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.
- Sjáum strax að maður vill alltaf breyta 0 í 1 og aldrei öfugt.

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n\in\mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.
- Sjáum strax að maður vill alltaf breyta 0 í 1 og aldrei öfugt.
- Sjáum því að við erum að leita að lengstu bilunum í $(a_n)_{n\in\mathbb{N}}$ sem hefur í mesta lagi k stök jöfn 0.

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n \in \mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.
- Sjáum strax að maður vill alltaf breyta 0 í 1 og aldrei öfugt.
- Sjáum því að við erum að leita að lengstu bilunum í $(a_n)_{n \in \mathbb{N}}$ sem hefur í mesta lagi k stök jöfn 0.
- Gefum okkur nú hlaupabil. Það byrjar tómt.

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n \in \mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.
- Sjáum strax að maður vill alltaf breyta 0 í 1 og aldrei öfugt.
- Sjáum því að við erum að leita að lengstu bilunum í $(a_n)_{n \in \mathbb{N}}$ sem hefur í mesta lagi k stök jöfn 0.
- Gefum okkur nú hlaupabil. Það byrjar tómt.
- Við löbbum svo í gegnum $(a_n)_{n \in \mathbb{N}}$ og lengjum bilið að aftan.

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengstu bilanna í rununni $(a_n)_{n \in \mathbb{N}}$ sem innihelda bara 1 ef þú mátt breyta allt að k tölum.
- Sjáum strax að maður vill alltaf breyta 0 í 1 og aldrei öfugt.
- Sjáum því að við erum að leita að lengstu bilunum í $(a_n)_{n \in \mathbb{N}}$ sem hefur í mesta lagi k stök jöfn 0.
- Gefum okkur nú hlaupabil. Það byrjar tómt.
- Við löbbum svo í gegnum $(a_n)_{n \in \mathbb{N}}$ og lengjum bilið að aftan.
- Ef það eru einhvern tímann fleiri en k stök í bilinu sem eru 0 þá minnkum við bilið að framan þar til svo er ekki lengur.

```
k = 2
l = 0
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|
```

```
k = 2
l = 1
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|  |
```

```
k = 2
l = 2
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|   |
```

```
k = 2
l = 3
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|     |
```

```
k = 2
l = 4
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|       |
```

```
k = 2
l = 5
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
|         |
```

```
k = 2
l = 4
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
  |       |
```

```
k = 2
l = 5
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
  |         |
```

```
k = 2
l = 4
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
    |       |
```

```
k = 2
l = 3
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
    |     |
```

```
k = 2
l = 2
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
        |   |
```

```
k = 2
l = 3
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
        |     |
```

```
k = 2
l = 2
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
          |   |
```

```
k = 2
l = 1
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
            | |
```

```
k = 2
l = 2
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
            |   |
```

```
k = 2
l = 3
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
            |     |
```

```
k = 2
l = 4
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
            |       |
```

```
k = 2
l = 5
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
          |         |
```

```
k = 2
l = 6
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
            |           |
```

```
k = 2
l = 5
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
              |         |
```

```
k = 2
l = 6
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
              |           |
```

```
k = 2
l = 5
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
                |         |
```

```
k = 2
l = 6
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
                |           |
```

```
k = 2
l = 7
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
        |              |
```

```
k = 2
l = 8
[0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1]
                |               |
```

```c
int main()
{
    int n, k, i;
    scanf("%d%d", &n, &k);
    int a[n];
    for (i = 0; i < n; i++) scanf("%d", &(a[i]));
    int b = 0, z = 0, mx = 0;
    for (i = 0; i < n; i++)
    {
        if (a[i] == 0) z++;
        while (z > k)
        {
            if (a[b] == 0) z--;
            b++;
        }
        if (i - b + 1 > mx) mx = i - b + 1;
    }
    printf("%d\n", mx);
    return 0;
}
```

- Hver tala í rununni er sett einu sinni í hlaupabilið og mögulega fjarlægð úr því.

- Hver tala í rununni er sett einu sinni í hlaupabilið og mögulega fjarlægð úr því.
- Svo tímaflækjan er $\mathcal{O}(\quad)$.

- Hver tala í rununni er sett einu sinni í hlaupabilið og mögulega fjarlægð úr því.
- Svo tímaflækjan er $\mathcal{O}(\ n\ )$.

- Þetta dæmi var í auðveldari kantinum.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.
- Tvö bil kallast *næstum sundurlæg* ef sniðmengi þeirra er tómt eða bara einn punktur.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.
- Tvö bil kallast *næstum sundurlæg* ef sniðmengi þeirra er tómt eða bara einn punktur.
- Sammengi bila má skrifa sem sammengi næstu sundurlægra bila.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.
- Tvö bil kallast *næstum sundurlæg* ef sniðmengi þeirra er tómt eða bara einn punktur.
- Sammengi bila má skrifa sem sammengi næstu sundurlægra bila.
- *Lengd bilsins* $[a, b]$ er $b - a$.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.
- Tvö bil kallast *næstum sundurlæg* ef sniðmengi þeirra er tómt eða bara einn punktur.
- Sammengi bila má skrifa sem sammengi næstu sundurlægra bila.
- *Lengd bilsins* $[a, b]$ er $b - a$.
- Til að finna *lengd sammengis bila* skrifum við sammengið sem sammengi næstum sundurlægra bila og tökum summu lengda þeirra.

- Þetta dæmi var í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.
- Tvö bil kallast *næstum sundurlæg* ef sniðmengi þeirra er tómt eða bara einn punktur.
- Sammengi bila má skrifa sem sammengi næstu sundurlægra bila.
- *Lengd bilsins* $[a, b]$ er $b - a$.
- Til að finna *lengd sammengis bila* skrifum við sammengið sem sammengi næstum sundurlægra bila og tökum summu lengda þeirra.
- Til dæmis eru bilin $[1, 2]$ og $[2, 3]$ næstum sundurlæg (en þó ekki sundurlæg) en $[1, 3]$ og $[2, 4]$ eru það ekki. Nú $[1, 3] \cup [2, 4] = [1, 4]$ svo lengd $[1, 3] \cup [2, 4]$ er 3.

- Gefið n bil hver er lengd sammengis þeirra.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.
- Við löbbum í gegnum þennan raðaða lista og höldum utan um hlaupabil þannig að við bætum við bili í hlaupabilið þegar við rekumst á vinstri endapunkt þess og fjarlægjum það þegar við rekumst á hægri endapunkt þess.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.
- Við löbbum í gegnum þennan raðaða lista og höldum utan um hlaupabil þannig að við bætum við bili í hlaupabilið þegar við rekumst á vinstri endapunkt þess og fjarlægjum það þegar við rekumst á hægri endapunkt þess.
- Við skoðum svo sérstaklega tilfellin þegar við erum ekki með nein bil í hlaupabilinu okkar.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.
- Við löbbum í gegnum þennan raðaða lista og höldum utan um hlaupabil þannig að við bætum við bili í hlaupabilið þegar við rekumst á vinstri endapunkt þess og fjarlægjum það þegar við rekumst á hægri endapunkt þess.
- Við skoðum svo sérstaklega tilfellin þegar við erum ekki með nein bil í hlaupabilinu okkar.
- Sammengi þeirra bila sem við höfum farið í gegnum þá síðan hlaupabilið var síðast tómt er nú sundurlægt öllum öðrum bilum sem okkur var gefið í byrjun.

- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur bils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.
- Við löbbum í gegnum þennan raðaða lista og höldum utan um hlaupabil þannig að við bætum við bili í hlaupabilið þegar við rekumst á vinstri endapunkt þess og fjarlægjum það þegar við rekumst á hægri endapunkt þess.
- Við skoðum svo sérstaklega tilfellin þegar við erum ekki með nein bil í hlaupabilinu okkar.
- Sammengi þeirra bila sem við höfum farið í gegnum þá síðan hlaupabilið var síðast tómt er nú sundurlægt öllum öðrum bilum sem okkur var gefið í byrjun.
- Við skilum því summu lengda þessara sammengja.

```
   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
   |
[]
r = 0
```

```
     |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                       x------x
 6:                                              x--x
 7:                                          x------x
 8:                         x--x
 9:                                    x------------x
10:                       x------x
     |
[]
r = 0
```

```
     |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                          x------x
 6:                                                x--x
 7:                                            x------x
 8:                            x--x
 9:                                      x------------x
10:                          x------x
     |
[1]
r = 0
```

```
     |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
     |
[1, 3]
r = 0
```

```
        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                          x------x
 6:                                                 x--x
 7:                                             x------x
 8:                            x--x
 9:                                       x------------x
10:                          x------x
        |
[1, 3]
r = 0
```

```
        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
        |
[1, 2, 3]
r = 0
```

```
          |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
          |
[1, 2, 3]
r = 0
```

```
          |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                          x------x
 6:                                             x--x
 7:                                         x------x
 8:                            x--x
 9:                                   x------------x
10:                          x------x
          |
[1, 2]
r = 0
```

```
             |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                            x--x
 7:                                        x------x
 8:                           x--x
 9:                                  x------------x
10:                         x------x
             |
[1, 2]
r = 0
```

```
             |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                               x--x
 7:                                           x------x
 8:                          x--x
 9:                                     x------------x
10:                        x------x
             |
[1, 2, 4]
r = 0
```

```
             |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
             |
[1, 4]
r = 0
```

```
                  |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                  |
[1, 4]
r = 0
```

```
                  |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                  |
[4]
r = 0
```

```
                        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:         x----------x
 5:                        x------x
 6:                                               x--x
 7:                                           x------x
 8:                          x--x
 9:                                     x------------x
10:                        x------x
                        |
[4]
r = 0
```

```
                        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                        |
[]
r = 0
```

```
                        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                        |
[]
r = 20
```

```
                            |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                               x--x
 7:                                           x------x
 8:                           x--x
 9:                                     x------------x
10:                         x------x
                            |
[]
r = 20
```

```
                            |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                            |
[5]
r = 20
```

```
                            |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                            |

[5, 10]
r = 20
```

```
                              |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                              |

[5, 10]
r = 20
```

```
                              |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                               x--x
 7:                                           x------x
 8:                          x--x
 9:                                     x------------x
10:                        x------x
                              |
[5, 8, 10]
r = 20
```

```
                                 |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                          x------x
 6:                                                x--x
 7:                                            x------x
 8:                            x--x
 9:                                      x------------x
10:                          x------x
                                 |

[5, 8, 10]
r = 20
```

```
                                 |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                 |

[5, 10]
r = 20
```

```
                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                   |

[5, 10]
r = 20
```

```
                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                   |

[5]
r = 20
```

```
                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                   x--x
 7:                                               x------x
 8:                           x--x
 9:                                         x------------x
10:                         x------x
                                   |

[]
r = 20
```

```
                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                                   x--x
 7:                                               x------x
 8:                          x--x
 9:                                         x------------x
10:                        x------x
                                   |

[]
r = 28
```

```
                                         |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                         |
[]
r = 28
```

```
                                         |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                         |
```

[9]

r = 28

```
                                               |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                                 x--x
 7:                                             x------x
 8:                          x--x
 9:                                        x------------x
10:                        x------x
                                               |

[9]
r = 28
```

```
                                               |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                               |

[7, 9]
r = 28
```

```
                                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                          x------x
 6:                                                x--x
 7:                                            x------x
 8:                            x--x
 9:                                      x------------x
10:                          x------x
                                                   |

[7, 9]
r = 28
```

```
                                                   |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                                   |

[6, 7, 9]
r = 28
```

```
                                                        |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                               x--x
 7:                                           x------x
 8:                          x--x
 9:                                     x------------x
10:                        x------x
                                                        |
```

[6, 7, 9]

r = 28

```
                                                      |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                                      |

[7, 9]
r = 28
```

```
                                                      |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                               x--x
 7:                                           x------x
 8:                           x--x
 9:                                     x------------x
10:                         x------x
                                                      |

[9]
r = 28
```

```
                                                      |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                         x------x
 6:                                                x--x
 7:                                            x------x
 8:                           x--x
 9:                                      x------------x
10:                         x------x
                                                      |

[]
r = 28
```

```
                                                      |
 1:  x------------x
 2:     x----x
 3:  x----x
 4:          x----------x
 5:                        x------x
 6:                                               x--x
 7:                                           x------x
 8:                          x--x
 9:                                     x------------x
10:                        x------x
                                                      |

[]
r = 42
```

```c
typedef struct { int x, y; } ii;
int cmp(const void* p1, const void* p2) { return ((ii*)p1)->x - ((ii*)p2)->x; }

int main()
{
    int n, r, i, j, k;
    scanf("%d", &n);
    ii a[2*n];
    int b[n];
    for (i = 0; i < n; i++)
    {
        scanf("%d%d", &(a[2*i].x), &(a[2*i + 1].x));
        a[2*i].y = i; a[2*i + 1].y = i; b[i] = 0;
    }
    qsort(a, 2*n, sizeof(a[0]), cmp);
    i = 0, r = 0;
    while (i < 2*n)
    {
        k = 1, j = i + 1, b[a[i].y] = 1;
        while (k > 0)
        {
            if (b[a[j].y] == 1) k--;
            else b[a[j].y] = 1, k++;
            j++;
        }
        r = r + a[j - 1].x - a[i].x; i = j;
    }
    printf("%d\n", r);
    return 0;
}
```

- Við byrjum á að raða í $\mathcal{O}(\qquad)$ tíma.

- Við byrjum á að raða í $\mathcal{O}(n \log n)$ tíma.

- Við byrjum á að raða í $\mathcal{O}(n \log n)$ tíma.
- Síðan ítrum við í gegnum alla endapunktana sem tekur $\mathcal{O}(\quad)$ tíma.

- Við byrjum á að raða í $\mathcal{O}(n \log n)$ tíma.
- Síðan ítrum við í gegnum alla endapunktana sem tekur $\mathcal{O}(\ n\ )$ tíma.

- Við byrjum á að raða í $\mathcal{O}(n \log n)$ tíma.
- Síðan ítrum við í gegnum alla endapunktana sem tekur $\mathcal{O}(n)$ tíma.
- Svo lausnin hefur tímaflækjuna $\mathcal{O}(\quad)$.

- Við byrjum á að raða í $\mathcal{O}(n \log n)$ tíma.
- Síðan ítrum við í gegnum alla endapunktana sem tekur $\mathcal{O}(n)$ tíma.
- Svo lausnin hefur tímaflækjuna $\mathcal{O}(n \log n)$.

Næsta stærra stak (`NGE`)

- Látum a vera lista af n tölum.

Næsta stærra stak (`NGE`)

- Látum a vera lista af n tölum.
- Okkar langar, fyrir hvert stak í listanum, að finna næst stak í listanum sem er stærra (e. *next greater element* (`NGE`)).

Næsta stærra stak (NGE)

- Látum a vera lista af n tölum.
- Okkar langar, fyrir hvert stak í listanum, að finna næst stak í listanum sem er stærra (e. *next greater element* (NGE)).
- Sem dæmi er NGE miðju stakins 4 í listanum $(2, 3, 4, 8, 5)$ talan 8.

Næsta stærra stak (NGE)

- Látum a vera lista af n tölum.
- Okkar langar, fyrir hvert stak í listanum, að finna næst stak í listanum sem er stærra (e. *next greater element* (NGE)).
- Sem dæmi er NGE miðju stakins 4 í listanum $(2, 3, 4, 8, 5)$ talan 8.
- Til þæginda segjum við að NGE tölunnar 8 í listanum $(2, 3, 4, 8, 5)$ sé -1.

Næsta stærra stak (`NGE`)

- Látum a vera lista af n tölum.
- Okkar langar, fyrir hvert stak í listanum, að finna næst stak í listanum sem er stærra (e. *next greater element* (`NGE`)).
- Sem dæmi er `NGE` miðju stakins 4 í listanum $(2, 3, 4, 8, 5)$ talan 8.
- Til þæginda segjum við að `NGE` tölunnar 8 í listanum $(2, 3, 4, 8, 5)$ sé -1.
- Það er auðséð að við getum reiknað `NGE` allra talnanna með tvöfaldri `for`-lykkju.

```c
void nge(int* a, int* b, int n)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n - i; j++) if (a[i] < a[i + j]) break;
        b[i] = (j == n - i ? -1 : i + j);
    }
}
```

- Þar sem þessi lausnir er tvöföld `for`-lykkja, hvor af lengd n, þá er lausnin $\mathcal{O}(\quad)$.

- Þar sem þessi lausnir er tvöföld `for`-lykkja, hvor af lengd n, þá er lausnin $\mathcal{O}(n^2)$.

- En þetta má bæta.

- En þetta má bæta.
- Gefum okkur hlaða h.

- En þetta má bæta.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.

- En þetta má bæta.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.
- Tökum nú tölur úr hlaðan og setjum `NGE` þeirra talna sem a_i á meðan a_i er stærri en toppurinn á hlaðanum.

- En þetta má bæta.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.
- Tökum nú tölur úr hlaðan og setjum `NGE` þeirra talna sem a_i á meðan a_i er stærri en toppurinn á hlaðanum.
- Þegar toppurinn á hlaðanum er stærri en a_i þá látum við a_i á hlaðann og höldum svo áfram.

- En þetta má bæta.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.
- Tökum nú tölur úr hlaðan og setjum `NGE` þeirra talna sem a_i á meðan a_i er stærri en toppurinn á hlaðanum.
- Þegar toppurinn á hlaðanum er stærri en a_i þá látum við a_i á hlaðann og höldum svo áfram.
- Bersýnilega er hlaðinn ávallt raðaður, svo þú færð allar tölur sem eiga að hafa a_i sem `NGE`.

- En þetta má bæta.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.
- Tökum nú tölur úr hlaðan og setjum `NGE` þeirra talna sem a_i á meðan a_i er stærri en toppurinn á hlaðanum.
- Þegar toppurinn á hlaðanum er stærri en a_i þá látum við a_i á hlaðann og höldum svo áfram.
- Bersýnilega er hlaðinn ávallt raðaður, svo þú færð allar tölur sem eiga að hafa a_i sem `NGE`.
- Þegar búið er að fara í gegnum a látum við `NGE` þeirra staka sem eftir eru í h vera -1.

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
|

 0 1 2 3 4 5 6 7
[x x x x x x x x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
 |

 0 1 2 3 4 5 6 7
[x x x x x x x x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
 |

 0 1 2 3 4 5 6 7
[x x x x x x x x]


h: [0]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   |

 0 1 2 3 4 5 6 7
[x x x x x x x x]


h: [0]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
 ^ |

 0 1 2 3 4 5 6 7
[x x x x x x x x]


h: [0]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
 ^ |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [0]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
     |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   ^ |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
     |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
     |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1 2]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
       |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1 2]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
     ^ |

 0 1 2 3 4 5 6 7
[1 x x x x x x x]


h: [1 2]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
     ^ |

 0 1 2 3 4 5 6 7
[1 x 3 x x x x x]


h: [1 2]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   ^   |

 0 1 2 3 4 5 6 7
[1 x 3 x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
   ^   |

 0 1 2 3 4 5 6 7
[1 3 3 x x x x x]


h: [1]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
       |

 0 1 2 3 4 5 6 7
[1 3 3 x x x x x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
       |

 0 1 2 3 4 5 6 7
[1 3 3 x x x x x]


h: [3]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         |

 0 1 2 3 4 5 6 7
[1 3 3 x x x x x]


h: [3]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
       ^ |

 0 1 2 3 4 5 6 7
[1 3 3 x x x x x]


h: [3]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
       ^ |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [3]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
           |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         ^ |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
           |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]
```

```
h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
           |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
             |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
           ^ |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
             |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
             |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5 6]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
               |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5 6]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
             ^ |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x x x]


h: [4 5 6]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
             ^ |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x 7 x]


h: [4 5 6]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
          ^   |

 0 1 2 3 4 5 6 7
[1 3 3 4 x x 7 x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
           ^   |

 0 1 2 3 4 5 6 7
[1 3 3 4 x 7 7 x]


h: [4 5]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         ^     |

 0 1 2 3 4 5 6 7
[1 3 3 4 x 7 7 x]


h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
         ^     |

 0 1 2 3 4 5 6 7
[1 3 3 4 7 7 7 x]


h: [4]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
               |

 0 1 2 3 4 5 6 7
[1 3 3 4 7 7 7 x]


h: []
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
               |

 0 1 2 3 4 5 6 7
[1 3 3 4 7 7 7 x]


h: [8]
```

```
 0 1 2 3 4 5 6 7
[2 3 1 5 7 6 4 8]
```

```
 0 1 2 3 4 5 6 7
[1 3 3 4 7 7 7 x]
```

```
h: [8]
```

```c
void nge(int* a, int* b, int n)
{
    int s[n], c = 0, i;
    for (i = 0; i < n; i++)
    {
        while (c > 0 && a[s[c - 1]] < a[i]) b[s[--c]] = i;
        s[c++] = i;
    }
    while (c > 0) b[s[--c]] = -1;
}
```

- Við setjum hverja tölu í hlaðann að mestu einu sinni og tökum hana svo úr hlaðanum.

- Við setjum hverja tölu í hlaðann að mestu einu sinni og tökum hana svo úr hlaðanum.
- Svo tímaflækjan er $\mathcal{O}(\quad)$.

- Við setjum hverja tölu í hlaðann að mestu einu sinni og tökum hana svo úr hlaðanum.
- Svo tímaflækjan er $\mathcal{O}(\, n \,)$.