Strengir et cetera

Bergur Snorrason

27. mars 2019

Google Code Jam

- Aðfaranótt 6. apríl hefst fyrsta umferð í Google Code Jam.

Google Code Jam

- Aðfaranótt 6. apríl hefst fyrsta umferð í Google Code Jam.
- Hún stendur yfir í 24 tíma.

Google Code Jam

- Aðfaranótt 6. apríl hefst fyrsta umferð í Google Code Jam.
- Hún stendur yfir í 24 tíma.
- Ég mæli með.

Efnisyfirlit

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að
  - "Dulkóðun".

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að
  - "Dulkóðun".
  - Finna tíðna (stafa eða orða).

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að
    - "Dulkóðun".
    - Finna tíðna (stafa eða orða).
    - Lesa inn leiðinlegt inntak.

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að
  - "Dulkóðun".
  - Finna tíðna (stafa eða orða).
  - Lesa inn leiðinlegt inntak.
  - Skila leiðinlegu úttaki.

Ad hoc strengjadæmi

- Í keppnum koma stundum létt strengjadæmi.
- Til dæmis gæti þurft að
  - "Dulkóðun".
  - Finna tíðna (stafa eða orða).
  - Lesa inn leiðinlegt inntak.
  - Skila leiðinlegu úttaki.
- Við höfum nú þegar látið ykkur gera flest þetta áður.

Brögð

- Ef ég gef ykkur streng og bið ykkur um að finna lengsta hlutbil í strengum sem hefur alla stafi eins, hvernig mynduð þið leysa það?

Brögð

- Ef ég gef ykkur streng og bið ykkur um að finna lengsta hlutbil í strengum sem hefur alla stafi eins, hvernig mynduð þið leysa það?
- Hvað ef strengurinn er skrifaður á hring, þ.e.a.s. hlutbil getur farið út fyrir end strengsins og heldur þá áfram í byrjun hans („urBe” er þá hlutbil í „Bergur”)?

Brögð

- Ef ég gef ykkur streng og bið ykkur um að finna lengsta hlutbil í strengum sem hefur alla stafi eins, hvernig mynduð þið leysa það?
- Hvað ef strengurinn er skrifaður á hring, þ.e.a.s. hlutbil getur farið út fyrir end strengsins og heldur þá áfram í byrjun hans („urBe" er þá hlutbil í „Bergur")?
- Sígild leið til að leysa þessa gerð dæmi er að skeyta strengum við sjálfan sig.

Brögð

- Ef ég gef ykkur streng og bið ykkur um að finna lengsta hlutbil í strengum sem hefur alla stafi eins, hvernig mynduð þið leysa það?
- Hvað ef strengurinn er skrifaður á hring, þ.e.a.s. hlutbil getur farið út fyrir end strengsins og heldur þá áfram í byrjun hans („urBe" er þá hlutbil í „Bergur")?
- Sígild leið til að leysa þessa gerð dæmi er að skeyta strengum við sjálfan sig.
- Svo, t.d., „Bergur" verður „BergurBergur" og „urBe" er bersýnilega hlutbil í seinni strengnum.

Brögð

- Ef ég gef ykkur streng og bið ykkur um að finna lengsta hlutbil í strengum sem hefur alla stafi eins, hvernig mynduð þið leysa það?
- Hvað ef strengurinn er skrifaður á hring, þ.e.a.s. hlutbil getur farið út fyrir end strengsins og heldur þá áfram í byrjun hans („urBe" er þá hlutbil í „Bergur")?
- Sígild leið til að leysa þessa gerð dæmi er að skeyta strengum við sjálfan sig.
- Svo, t.d., „Bergur" verður „BergurBergur" og „urBe" er bersýnilega hlutbil í seinni strengnum.
- Þessi aðferð virkar einnig fyrir aðra hluti en strengjaleit.

Efnisyfirlit

1. Strengir
   - Ad hoc
   - Strengjaleit
   - Knuth-Morris-Pratt-Matiyasevich
2. Et cetera
   - Hlaupabil
   - Lengsta vaxandi hlutruna
   - Næsta stærra stak
   - Línuleg rakningarvensl
   - Röðun
   - Gauss-Jordan útfærsla

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.

Strengjaleit

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

Strengjaleit

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

Strengjaleit

- Gerum ráð fyrir að s sé af lengd n og p sé af lengd m.
- Þá hefur s $n - m + 1$ hlutstrengi af lengd m.

Strengjaleit

- Gerum ráð fyrir að s sé af lengd n og p sé af lengd m.
- Þá hefur s $n - m + 1$ hlutstrengi af lengd m.
- Strengja samanburðurinn tekur línulegan tíma.

Strengjaleit

- Gerum ráð fyrir að s sé af lengd n og p sé af lengd m.
- Þá hefur s $n - m + 1$ hlutstrengi af lengd m.
- Strengja samanburðurinn tekur línulegan tíma.
- Svo tímaflækja leitarinnar er $\mathcal{O}(nm - m^2)$.

Strengjaleit

- Gerum ráð fyrir að s sé af lengd n og p sé af lengd m.
- Þá hefur s $n - m + 1$ hlutstrengi af lengd m.
- Strengja samanburðurinn tekur línulegan tíma.
- Svo tímaflækja leitarinnar er $\mathcal{O}(nm - m^2)$.
- Ef $m = \frac{n}{2}$ þá er $nm - m^2 = \frac{n^2}{2} - \frac{n^2}{4} = \frac{n^2}{4}$ svo $\mathcal{O}(nm - m^2) = \mathcal{O}(n^2)$.

Strengjaleit

- Gerum ráð fyrir að s sé af lengd n og p sé af lengd m.
- Þá hefur s $n - m + 1$ hlutstrengi af lengd m.
- Strengja samanburðurinn tekur línulegan tíma.
- Svo tímaflækja leitarinnar er $\mathcal{O}(nm - m^2)$.
- Ef $m = \frac{n}{2}$ þá er $nm - m^2 = \frac{n^2}{2} - \frac{n^2}{4} = \frac{n^2}{4}$ svo $\mathcal{O}(nm - m^2) = \mathcal{O}(n^2)$.
- Dæmi um leiðinlega strengi væri $s = "aaaaaaaaaaaaaaa"$ og $p = "aaaaaab"$.

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.
- Það er ekki mælt með því því hún er útfærð í mörgum málum, t.d.:

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.
- Það er ekki mælt með því því hún er útfærð í mörgum málum, t.d.:
  - Í string.h í C er strstr.

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.
- Það er ekki mælt með því því hún er útfærð í mörgum málum, t.d.:
    - Í string.h í `C` er strstr.
    - Í string í `C++` er find.

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.
- Það er ekki mælt með því því hún er útfærð í mörgum málum, t.d.:
  - Í string.h í `C` er strstr.
  - Í string í `C++` er find.
  - Í String í `Java` er indexOf.

Strengjaleit

- Þó þessi aðferð sé ekki góð þá er hún stundum nógu góð.
- Í þeim tilfellum er samt ekki mælt með því að útfæra hana.
- Það er ekki mælt með því því hún er útfærð í mörgum málum, t.d.:
  - Í string.h í `C` er strstr.
  - Í string í `C++` er find.
  - Í String í `Java` er indexOf.
- Munið bara að ef $n > 10\,000$ er þetta yfirleitt of hægt.

Efnisyfirlit

KMP

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

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

KMP

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p = $ "$aaaabbbb$".
- Ef strengja samanburðurinn misheppnast í $p[3]$ þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.

KMP

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ "$aaaabbbb$".
- Ef strengja samanburðurinn misheppnast í $p[3]$ þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að $(i + 3)$-ji stafur s er ekki "a" (því strengja samanburðurinn misheppnaðist þar) svo við getum í raun hliðrað p um 3.

KMP

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ "$aaaabbbb$".
- Ef strengja samanburðurinn misheppnast í $p[3]$ þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að $(i + 3)$-ji stafur s er ekki "a" (því strengja samanburðurinn misheppnaðist þar) svo við getum í raun hliðrað p um 3.
- Reiknirit Knuth-Morris-Pratt (KMP) notar sér þessa hugmynd til að framkvæma strengjaleit í línulegum tíma.

KMP

- Er einhver leið til að bæta strengjaleitina úr fyrri glærum?
- Skoðum betur sértilfellið $p =$ "$aaaabbbb$".
- Ef strengja samanburðurinn misheppnast í $p[3]$ þá myndi einfalda strengjaleitin okkar hliðra p um einn og reyna aftur.
- En við vitum að $(i + 3)$-ji stafur s er ekki "a" (því strengja samanburðurinn misheppnaðist þar) svo við getum í raun hliðrað p um 3.
- Reiknirit Knuth-Morris-Pratt (`KMP`) notar sér þessa hugmynd til að framkvæma strengjaleit í línulegum tíma.
- Í `KMP` er forreiknað (í línulegum tíma) hversu mikið maður getur hliðrað þegar samanburðurinn misheppnast.

KMP forreikningur

```c
void _kmp(char* p, int* b, int m)
{
    int i = 0, j = -1; b[0] = -1;
    while (i < m)
    {
        while (j >= 0 && p[i] != p[j])
        {
            j = b[j];
        }
        i++; j++; b[i] = j;
    }
}
```

- Svo þurfum við einfaldlega að labba í gegnum s og hliðra þegar á við.

KMP leit

```c
void kmp(char* s, int n, char* p, int m, int* b)
{
    int i = 0, j = 0;
    while (i < n)
    {
        while (j >= 0 && s[i] != p[j])
        {
            j = b[j];
        }
        i++; j++;
        if (j == m)
        {
            printf("%d\n", i - j); j = b[j];
        }
    }
}
```

KMP sýniforrit

```
#include <stdio.h>
int get_string(char* b, char t)
{
    int i = 0, c = getchar();
    while (c != t) { b[i++] = c; c = getchar(); }
    b[i] = '\0';
    return i;
}
void _kmp(char* p, int* b, int m)
{
    ...
}
void kmp(char* s, int n, char* p, int m, int* b)
{
    ...
}
int main()
{
    char s[1000001], p[1001];
    int i;
    printf("Langi strengur: "); fflush(stdout);
    int n = get_string(s, 10);
    printf("Stutti strengur: "); fflush(stdout);
    int m = get_string(p, 10);
    int b[m];
    _kmp(p, b, m);
    for (i = 0; i < m; i++) printf("%4d ", b[i]); printf("\n");
    for (i = 0; i < m; i++) printf("   %c ", p[i]); printf("\n");
    kmp(s, n, p, m, b);
}
```

Aho-Corasick

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

Aho-Corasick

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

Aho-Corasick

- 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 stöðuvél.

Aho-Corasick

- 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 stöðuvél.
- `KMP` er í raun sértilfelli af Aho-Corasick sem vill svo heppilega til að það sé þægilegt að útfæra.

Efnisyfirlit

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 $\mathcal{O}(n \log n)$.

Hlaupabil - Dæmi

- Skoðum dæmi:

Hlaupabil - Dæmi

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengsta bils í $(a_n)_n$ sem inniheldur bara 1 ef þú mátt breyta allt að k tölum.

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengsta bils í $(a_n)_n$ sem inniheldur 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 lengsta bili í $(a_n)_n$ sem hefur mesta k stök jöfn 0.

Hlaupabil - Dæmi

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengsta bils í $(a_n)_n$ sem inniheldur 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 lengsta bili í $(a_n)_n$ sem hefur mesta k stök jöfn 0.
- Gefum okkur nú hlaupabil. Það byrjar tómt.

Hlaupabil - Dæmi

- Skoðum dæmi:
- Gefið n, k og svo n tölur a_i, þ.a. $a_i \in \{0, 1\}$ finndu lengd lengsta bils í $(a_n)_n$ sem inniheldur 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 lengsta bili í $(a_n)_n$ sem hefur mesta k stök jöfn 0.
- Gefum okkur nú hlaupabil. Það byrjar tómt.
- Við löbbum svo í gegnum $(a_n)_n$ og lengjum bilið að aftan. Ef það eru einhvern tímann fleiri en k stök í bilinu sem eru 0 þá minnkum við bilið að aftan þar til svo er ekki lengur.

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

```
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]
|       |
```

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Dæmi

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

Hlaupabil - Útfærsla á dæmi

```c
#include <stdio.h>

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);
}
```

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski í auðveldari kantinum.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski í auðveldari kantinum.
- Skoðum annað dæmi:

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski í auðveldari kantinum.
- Skoðum annað dæmi:
- Byjrum á nokkrum undirstöðu atriðum.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski 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.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski 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.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski 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 $b - a$.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski 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.

Hlaupabil - Annað dæmi

- Þetta dæmi er nú kannski 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.

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.
- Einhverjar hugmyndir?

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.
- Einhverjar hugmyndir?
- Þetta er hægt að leysa á fleiri en eina vegu, en hér er mín lausn:

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.
- Einhverjar hugmyndir?
- Þetta er hægt að leysa á fleiri en eina vegu, en hér er mín lausn:
- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur einhversbils og seinna stakið segir hvaða bili punkturinn tilleyrir.

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.
- Einhverjar hugmyndir?
- Þetta er hægt að leysa á fleiri en eina vegu, en hér er mín lausn:
- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur einhversbils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.

Hlaupabil - Annað dæmi

- Gefið n bil hver er lengd sammengis þeirra.
- Stillum n þannig að $\mathcal{O}(n^2)$ sé of hægt.
- Einhverjar hugmyndir?
- Þetta er hægt að leysa á fleiri en eina vegu, en hér er mín lausn:
- Geymum í lista tvenndir þar sem fyrra stakið er endapunktur einhversbils og seinna stakið segir hvaða bili punkturinn tilleyrir.
- Röðum þessum punktum svo í vaxandi röð.
- Hvað gerum við næst?

Hlaupabil - Annað dæmi

- 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.

Hlaupabil - Annað dæmi

- 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 okkur.

Hlaupabil - Annað dæmi

- 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 okkur.
- 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.

Hlaupabil - Annað dæmi

- 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 okkur.
- 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.

Hlaupabil - Annað dæmi

- 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 okkur.
- 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.
- Þessi lausn er $\mathcal{O}(n \log n)$ því við þurftum að raða.

Hlaupabil - Annað dæmi

```
   |
 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
```

Hlaupabil - Annað dæmi

```
     |
 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
```

Hlaupabil - Annað dæmi

```
     |
 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
```

Hlaupabil - Annað dæmi

```
        |
 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
```

Hlaupabil - Annað dæmi

```
          |
 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
```

Hlaupabil - Annað dæmi

```
                 |
 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
```

Hlaupabil - Annað dæmi

```
                 |
 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
```

Hlaupabil - Annað dæmi

```
                     |
 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
```

Hlaupabil - Annað dæmi

```
                            |
 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 = 24
```

Hlaupabil - Annað dæmi

```
                                |
 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 = 24
```

Hlaupabil - Annað dæmi

```
                                |
 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 = 24
```

Hlaupabil - Annað dæmi

```
                                  |
 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 = 24
```

Hlaupabil - Annað dæmi

```
                                     |
 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 = 24
```

Hlaupabil - Annað dæmi

```
                                       |

 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 = 24
```

Hlaupabil - Annað dæmi

```
                                       |

 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 = 32
```

Hlaupabil - Annað dæmi

```
                                             |
 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 = 32
```

Hlaupabil - Annað dæmi

```
                                                   |
 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 = 32
```

Hlaupabil - Annað dæmi

```
                                                       |
 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 = 32
```

Hlaupabil - Annað dæmi

```
                                                          |

 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 = 32
```

```
                                                          |
 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 = 32
```

Hlaupabil - Annað dæmi

```
                                                          |

 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 = 46
```

Hlaupabil - Útfærsla á dæmi

```c
#include <stdlib.h>
#include <stdio.h>
typedef struct { int x, y; } par;
int cmp(const void* p1, const void* p2) { return ((par*)p1)->x - ((par*)p2)->x; }

int main()
{
    int n, r, i, j, k;
    scanf("%d", &n);
    par 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);
}
```

Efnisyfirlit

- Hlutruna í talnarunu er runa af tölum, allar úr upprunalegu rununni, sem eru í sömu röð og í upprunalegu röðinni.

LIS

- Hlutruna í talnarunu er runa af tölum, allar úr upprunalegu rununni, sem eru í sömu röð og í upprunalegu röðinni.
- Hvernig getum við fundið lengtsu vaxandi hlutrunu (e. longest increasing subsequence (`LIS`)) í gefinni runu?

LIS

- Hlutruna í talnarunu er runa af tölum, allar úr upprunalegu rununni, sem eru í sömu röð og í upprunalegu röðinni.
- Hvernig getum við fundið lengtsu vaxandi hlutrunu (e. longest increasing subsequence (`LIS`)) í gefinni runu?
- Sem dæmi er $[2\ 3\ 5\ 9]$ ein ef lengstu vaxandi hlutrunum $[2\ 3\ 1\ 5\ 9\ 8\ 7]$.

LIS $n2^n$

- Gerum ráð fyrir að við höfum talnarunu af lengd $1 \leq n \leq 15$.

LIS $n2^n$

- Gerum ráð fyrir að við höfum talnarunu af lengd $1 \leq n \leq 15$.
- Við getum þá prófað allar hlutrunur, skoðað hvort þær séu vaxandi og geymt þá lengstu.

LIS $n2^n$

```c
#include <stdio.h>
int main() {
    int n, i, j; scanf("%d", &n); int a[n];
    for (i = 0; i < n; i++) {
        scanf("%d", &(a[i]));
    }
    int mx = 0, mxi;
    for (i = 1; i < (1 << n); i++) {
        int s[n], sc = 0;
        for (j = 0; j < n; j++) {
            if (((1 << j)&i) != 0) {
                s[sc++] = j;
            }
        }
        for (j = 1; j < sc; j++)
            if (a[s[j]] < a[s[j - 1]]) break;
        if (j == sc && sc > mx) {
            mx = sc;
            mxi = i;
        }
    }
    printf("%d\n", mx);
    for (i = 0; i < n; i++) {
        if (((1 << i)&mxi) != 0) {
            printf("%d ", a[i]);
        }
    }
    printf("\n");
}
```

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.
- Við getum notað kvika bestun!

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.
- Við getum notað kvika bestun!
- Látum a vera runu af n tölum.

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.
- Við getum notað kvika bestun!
- Látum a vera runu af n tölum.
- Látum $f(k)$ vera lengd lengstu hlutrunu sem endar í k-ta staki a.

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.
- Við getum notað kvika bestun!
- Látum a vera runu af n tölum.
- Látum $f(k)$ vera lengd lengstu hlutrunu sem endar í k-ta staki a.
- Við getum þá reiknað gildi f í línulegum tíma og gerum það fyrir sérhverja tölu $1, 2, ..., n$.

LIS n^2

- Það er, ótrúlegt en satt, hægt að gera þetta hraðar.
- Við getum notað kvika bestun!
- Látum a vera runu af n tölum.
- Látum $f(k)$ vera lengd lengstu hlutrunu sem endar í k-ta staki a.
- Við getum þá reiknað gildi f í línulegum tíma og gerum það fyrir sérhverja tölu $1, 2, ..., n$.
- **Æfing:** Finna rununa.

LIS n^2

```
#include <stdio.h>

int main()
{
    int n, i, j;
    scanf("%d", &n);
    int a[n], c[n];
    for (i = 0; i < n; i++)
    {
        scanf("%d", &(a[i]));
    }
    for (i = 0; i < n; i++)
    {
        c[i] = 1;
        for (j = 0; j < i; j++)
        {
            if (a[i] > a[j] && c[i] < c[j] + 1)
            {
                c[i] = c[j] + 1;
            }
        }
    }
    for (i = 0; i < n; i++) printf("%4d", a[i]); printf("\n");
    for (i = 0; i < n; i++) printf("%4d", c[i]); printf("\n");
}
```

```
ARCH% ./lisn2
7 2 3 1 5 9 8 7
   2   3   1   5   9   8   7
   1   2   1   3   4   4   4
```

LIS $n \log n$

- En er hægt að gera þetta ennþá hraðar?

LIS $n \log n$

- En er hægt að gera þetta ennþá hraðar?
- Heldur betur!

LIS $n \log n$

- En er hægt að gera þetta ennþá hraðar?
- Heldur betur!
- Við getum notað helmingunarleit.

LIS $n \log n$

- En er hægt að gera þetta ennþá hraðar?
- Heldur betur!
- Við getum notað helmingunarleit.
- Skoðum first reiknirit sem er ekki hentugt að útfæra.

LIS $n \log n$

- Höfum lista af listum.

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.
- Listalistinn okkar byrjar tómur.

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.
- Listalistinn okkar byrjar tómur.
- Löbbum í gegnum a í réttri röð og fyrir hvert stak $a[i]$ finnum við (með helmingunarleit) þann lista sem hefur stærsta aftasta stakið sem er minna en $a[i]$.

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.
- Listalistinn okkar byrjar tómur.
- Löbbum í gegnum a í réttri röð og fyrir hvert stak $a[i]$ finnum við (með helmingunarleit) þann lista sem hefur stærsta aftasta stakið sem er minna en $a[i]$.
- Við afritum nú listann sem við fundum, setjum hann fyrir aftan, bætum stakinu okkar við hann og fjarlægjum listann fyrir aftann nýja listann (ef það er einhver).

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.
- Listalistinn okkar byrjar tómur.
- Löbbum í gegnum a í réttri röð og fyrir hvert stak $a[i]$ finnum við (með helmingunarleit) þann lista sem hefur stærsta aftasta stakið sem er minna en $a[i]$.
- Við afritum nú listann sem við fundum, setjum hann fyrir aftan, bætum stakinu okkar við hann og fjarlægjum listann fyrir aftann nýja listann (ef það er einhver).
- Hvert skref í þessu reikniriti er $\mathcal{O}(\log n)$ ef við getum afritað lista í föstum tíma (sem er ekki beint eðlilegt).

LIS $n \log n$

- Höfum lista af listum.
- Skilgreinum röðun þannig að listar eru bornir saman eftir síðasta staki.
- Listalistinn okkar byrjar tómur.
- Löbbum í gegnum a í réttri röð og fyrir hvert stak $a[i]$ finnum við (með helmingunarleit) þann lista sem hefur stærsta aftasta stakið sem er minna en $a[i]$.
- Við afritum nú listann sem við fundum, setjum hann fyrir aftan, bætum stakinu okkar við hann og fjarlægjum listann fyrir aftann nýja listann (ef það er einhver).
- Hvert skref í þessu reiknu riti er $\mathcal{O}(\log n)$ ef við getum afritað lista í föstum tíma (sem er ekki beint eðlilegt).
- Rúllum í gegnum þetta fyrir listann $[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]$.

LIS $n \log n$

[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
|

[0]

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
    |

[0]
[0, 8]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
       |

[0]
[0, 4]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
           |

[0]
[0, 4]
[0, 4, 12]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                |

[0]
[0, 2]
[0, 4, 12]
```

LIS $n \log n$

[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

[0]
[0, 2]
[0, 2, 10]

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                     |

[0]
[0, 2]
[0, 2, 6]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                |
```

```
[0]
[0, 2]
[0, 2, 6]
[0, 2, 6, 14]
```

LIS $n \log n$

[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

[0]
[0, 1]
[0, 2, 6]
[0, 2, 6, 14]

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                |

[0]
[0, 1]
[0, 2, 6]
[0, 2, 6, 9]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                      |

[0]
[0, 1]
[0, 1, 5]
[0, 2, 6, 9]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                           |

[0]
[0, 1]
[0, 1, 5]
[0, 2, 6, 9]
[0, 2, 6, 9, 13]
```

LIS $n \log n$

[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]

[0]
[0, 1]
[0, 1, 3]
[0, 2, 6, 9]
[0, 2, 6, 9, 13]

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                              |

[0]
[0, 1]
[0, 1, 3]
[0, 2, 6, 9]
[0, 2, 6, 9, 11]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                                 |

[0]
[0, 1]
[0, 1, 3]
[0, 1, 3, 7]
[0, 2, 6, 9, 11]
```

LIS $n \log n$

```
[0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
                                                   |

[0]
[0, 1]
[0, 1, 3]
[0, 1, 3, 7]
[0, 2, 6, 9, 11]
[0, 2, 6, 9, 11, 15]
```

LIS $n \log n$

- Hvernig útfærum við þetta?

- Hvernig útfærum við þetta?
- Við nýtum okkur það að allir listarnir sem við vorum með eru (að mestu) óþarfir því við þurfum bara aftasta stakið í hverjum þeirra.

LIS $n \log n$

- Hvernig útfærum við þetta?
- Við nýtum okkur það að allir listarnir sem við vorum með eru (að mestu) óþarfir því við þurfum bara aftasta stakið í hverjum þeirra.
- Við erum þá með lista af tölum, sem gerir allt mun auðveldara.

LIS $n \log n$

```
#define INF 2000000000

int bs(int* a, int n, int t)
{
    int r = 0, s = n;
    while (r < s)
        if (a[(r + s)/2] < t) r = (r + s)/2 + 1;
        else s = (r + s)/2;
    return r;
}

int lis(int* a, int n)
{
    int i, j;
    int b[n + 2];
    for (i = 0; i < n + 2; i++)
    {
        b[i] = INF;
    }
    b[0] = -INF;
    for (i = 0; i < n; i++)
    {
        j = bs(b, n + 1, a[i]);
        if (b[j - 1] < a[i] && a[i] < b[j]) b[j] = a[i];
    }
    for (i = 0; b[i] != INF; i++);
    return i - 1;
}
```

Efnisyfirlit

Næsta stærra stak

- Látum a vera lista af n tölum.

Næsta stærra stak

- Látum a vera lista af n tölum.
- Við segjum að næsta stak stærra en $a[i]$ (next greater element (`NGE`)) sé minnsta stak $a[j]$ þ.a. $j > i$.

Næsta stærra stak

- Látum a vera lista af n tölum.
- Við segjum að næsta stak stærra en $a[i]$ (next greater element (`NGE`)) sé minnsta stak $a[j]$ þ.a. $j > i$.
- Sem dæmi er `NGE` 4 í $[2, 3, 4, 8, 5]$ er 8.

Næsta stærra stak

- Látum a vera lista af n tölum.
- Við segjum að næsta stak stærra en $a[i]$ (next greater element (NGE)) sé minnsta stak $a[j]$ þ.a. $j > i$.
- Sem dæmi er NGE 4 í $[2, 3, 4, 8, 5]$ er 8.
- Til þæginda segjum við að NGE 8 í $[2, 3, 4, 8, 5]$ er -1.

Næsta stærra stak

- Látum a vera lista af n tölum.
- Við segjum að næsta stak stærra en $a[i]$ (next greater element (NGE)) sé minnsta stak $a[j]$ þ.a. $j > i$.
- Sem dæmi er NGE 4 í $[2, 3, 4, 8, 5]$ er 8.
- Til þæginda segjum við að NGE 8 í $[2, 3, 4, 8, 5]$ er -1.
- Það er auðséð að við getum reiknað NGE allra talnanna í $\mathcal{O}(n^2)$.

NGE 2^n

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

NGE n

- En að sjálfsögðu getum við gert þetta betur.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- Gefum okkur hlaða h.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- Gefum okkur hlaða h.
- Löbbum í gegnum a í réttri röð.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- 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.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- 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`.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- 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.

NGE n

- En að sjálfsögðu getum við gert þetta betur.
- 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.
- Þar sem maður setur hverja tölu einu sinni á hlaðann og tekur hana svo af þá er þetta reiknirit $\mathcal{O}(n)$.

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
|

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


h: []
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
 |

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


h: [2]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
   |

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


h: [3]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
     |

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


h: [3 1]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
       |

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


h: [5]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
         |

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


h: [9]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
           |

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


h: [9 6]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
               |

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


h: [9 8]
```

LIS $n \log n$

```
 0 1 2 3 4 5 6 7
[2 3 1 5 9 6 8 7]
               |

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


h: [9 8 7]
```

NGE n

```
void nge(int* a, int* b, int n)
{
    int s1[n], s2[n], c = 0, i;
    for (i = 0; i < n; i++)
    {
        while (c > 0 && s1[c - 1] < a[i]) b[s2[--c]] = i;
        s1[c] = a[i], s2[c] = i; c++;
    }
    while (c > 0) b[s2[--c]] = -1;
}
```

Efnisyfirlit

1. Strengir
   - Ad hoc
   - Strengjaleit
   - Knuth-Morris-Pratt-Matiyasevich
2. Et cetera
   - Hlaupabil
   - Lengsta vaxandi hlutruna
   - Næsta stærra stak
   - Línuleg rakningarvensl
   - Röðun
   - Gauss-Jordan útfærsla

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?
- Skoðum það aðeins nánar.

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?
- Skoðum það aðeins nánar.
- Hugmyndin er vigurinn

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} a \\ b \end{pmatrix}$$

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?
- Skoðum það aðeins nánar.
- Hugmyndin er vigurinn

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} a \\ b \end{pmatrix}$$

- inniheldur n-tu og $(n+1)$-tu Fibonacci-töluna með fræi a og b (s.s. venjulegu Fibonacci fást með $a = b = 1$).

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?
- Skoðum það aðeins nánar.
- Hugmyndin er vigurinn

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} a \\ b \end{pmatrix}$$

- inniheldur n-tu og $(n+1)$-tu Fibonacci-töluna með fræi a og b (s.s. venjulegu Fibonacci fást með $a = b = 1$).
- Við getum síðan reiknað þetta í $\log n$ með aðferðum úr síðasta fyrirlestri.

Fibonacci í $\log n$

- Munið þið þegar Atli talaði um að unnt væri að reikna n-tu Fibonacci-töluna í $\log n$?
- Skoðum það aðeins nánar.
- Hugmyndin er vigurinn

$$\left(\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array}\right)^n \left(\begin{array}{c} a \\ b \end{array}\right)$$

- inniheldur n-tu og $(n+1)$-tu Fibonacci-töluna með fræi a og b (s.s. venjulegu Fibonacci fást með $a = b = 1$).
- Við getum síðan reiknað þetta í $\log n$ með aðferðum úr síðasta fyrirlestri.
- Nánar tiltekið reiknum við þetta $k^3 \log n$ þar sem k er vídd fylkisins ($k = 2$ hér).

Fibonacci í $\log n$

- Málið er að þetta gildir almennt fyrir línuleg rakningar vensl.

Fibonacci í $\log n$

- Málið er að þetta gildir almennt fyrir línuleg rakningar vensl.
- Látum $(a_n)_{n \geq 1}$ þ.a. $a_n = \sum_{i=1}^{m} c_i a_{n-i}$, fyrir $n > m$ og $a_i = k_i$ annars. Hér eru c_i og k_i fastar.

Fibonacci í $\log n$

- Málið er að þetta gildir almennt fyrir línuleg rakningar vensl.
- Látum $(a_n)_{n \geq 1}$ þ.a. $a_n = \sum_{i=1}^{m} c_i a_{n-i}$, fyrir $n > m$ og $a_i = k_i$ annars. Hér eru c_i og k_i fastar.
- Við segjum nú að a_n ákvarðast af m-ta stigs línulegum rakningarvenslum með upphafsskilyrði $k_1, k_2, ..., k_m$ og fasta $c_1, c_2, ..., c_m$.

Fibonacci í $\log n$

- Málið er að þetta gildir almennt fyrir línuleg rakningar vensl.
- Látum $(a_n)_{n \geq 1}$ þ.a. $a_n = \sum_{i=1}^{m} c_i a_{n-i}$, fyrir $n > m$ og $a_i = k_i$ annars. Hér eru c_i og k_i fastar.
- Við segjum nú að a_n ákvarðast af m-ta stigs línulegum rakningarvenslum með upphafsskilyrði $k_1, k_2, ..., k_m$ og fasta $c_1, c_2, ..., c_m$.
- Við getum notað jöfnuna

$$\begin{pmatrix} c_1 & c_2 & \dots & c_{m-1} & c_m \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}^n \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_m \end{pmatrix} = \begin{pmatrix} a_{n+1} \\ a_{n+2} \\ \vdots \\ a_{n+m+1} \end{pmatrix}$$

til að reikna a_n í $k^3 \log n$. Þar sem k er yfirleitt lítið getum við oftast notað þetta fyrir mjög stór n.

Efnisyfirlit

Röðun

- Ég reikna með að þið kunnið öll að raða tölum.

Röðun

- Ég reikna með að þið kunnið öll að raða tölum.
- Stundum þegar maður raðar tölum vill maður viðhalda einhverskonar röð.

Röðun

- Ég reikna með að þið kunnið öll að raða tölum.
- Stundum þegar maður raðar tölum vill maður viðhalda einhverskonar röð.
- Röðunarreiknirt kallast *stöðugt* ef jafn stór stök viðhalda upprunalegu röð sinni.

Röðun

- Ég reikna með að þið kunnið öll að raða tölum.
- Stundum þegar maður raðar tölum vill maður viðhalda einhverskonar röð.
- Röðunarreiknirt kallast *stöðugt* ef jafn stór stök viðhalda upprunalegu röð sinni.
- Sem dæmi tökum við lista af tvenndum

$$(2, 6)\ (1, 3)\ (2, 3)\ (1, 1)\ (2, 2)$$

og röðum eftir fyrst stakinu.

Röðun

- Ég reikna með að þið kunnið öll að raða tölum.
- Stundum þegar maður raðar tölum vill maður viðhalda einhverskonar röð.
- Röðunarreiknirt kallast *stöðugt* ef jafn stór stök viðhalda upprunalegu röð sinni.
- Sem dæmi tökum við lista af tvenndum

$$(2,6)\ (1,3)\ (2,3)\ (1,1)\ (2,2)$$

  og röðum eftir fyrst stakinu.
- Ef við röðum stöðugt þá fæst

$$(1,3)\ (1,1)\ (2,6)\ (2,3)\ (2,2).$$

Röðun

- Í C++ er `stable_sort` stöðugt.

Röðun

- Í C++ er `stable_sort` stöðugt.
- Í Python er `sort` stöðugt.

Röðun

- Í `C++` er `stable_sort` stöðugt.
- Í `Python` er `sort` stöðugt.
- Í `C` er hægt að raða og nota stöðu staka í upprunalega listanum til að gera upp á milli jafn stórra talna.

Efnisyfirlit

Gauss-Jordan

- Fyrir nokkrum vikum sýndum við ykkur útfærslu á Gauss-Jordan eyðingu.

Gauss-Jordan

- Fyrir nokkrum vikum sýndum við ykkur útfærslu á Gauss-Jordan eyðingu.
- Fólki fannst hún nokkuð torlesin svo við skrifuðum aðra, mögulega þægilegri.

Gauss-Jordan

- Fyrir nokkrum vikum sýndum við ykkur útfærslu á Gauss-Jordan eyðingu.
- Fólki fannst hún nokkuð torlesin svo við skrifuðum aðra, mögulega þægilegri.
- Við erum búnir að prófa hana á nokkrum dæmum og Kattis hefur ekki ennþá kvartað.

Gauss-Jordan

```c
#define EPS 1e-9
// a er n x m fylki
void gauss(double* a, int n, int m)
{
    int i, j, k, t; double p;
    for (i = 0; i < n; i++)
    {
        t = -1;
        while (++t < m && fabs(a[i*m + t]) < EPS);

        if (t == m) continue;
        p = a[i*m + t];
        for (j = t; j < m; j++)
        {
            a[i*m + j] = a[i*m + j]/p;
        }

        for (j = 0; j < n; j++)
        {
            if (i != j)
            {
                p = a[j*m + t];
                for (k = t; k < m; k++)
                {
                    a[j*m + k] = a[j*m + k] - a[i*m + k]*p;
                }
            }
        }
    }
}
```