

Deila og drottna og kvik bestun

Bergur Snorrason

2. febrúar 2021

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc*.

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc.*
 - ▶ *Tæmandi leit eða ofbeldis aðferðin* (e. *complete search, brute force*),

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc*.
 - ▶ *Tæmandi leit* eða *ofbeldis aðferðin* (e. *complete search*, *brute force*),
 - ▶ *Gráðug reiknirit* (e. *greedy algorithms*),

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc.*
 - ▶ *Tæmandi leit eða ofbeldis aðferðin* (e. *complete search, brute force*),
 - ▶ *Gráðug reiknirit* (e. *greedy algorithms*),
 - ▶ *Deila og drottna* (e. *divide and conquer*),

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc*.
 - ▶ *Tæmandi leit* eða *ofbeldis aðferðin* (e. *complete search*, *brute force*),
 - ▶ *Gráðug reiknirit* (e. *greedy algorithms*),
 - ▶ *Deila og drottna* (e. *divide and conquer*),
 - ▶ *Kvik bestun* (e. *dynamic programming*).

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc*.
 - ▶ *Tæmandi leit* eða *ofbeldis aðferðin* (e. *complete search, brute force*),
 - ▶ *Gráðug reiknirit* (e. *greedy algorithms*),
 - ▶ *Deila og drottna* (e. *divide and conquer*),
 - ▶ *Kvik bestun* (e. *dynamic programming*).
- ▶ Í síðustu vikum fjölluðum við um *Ad hoc* dæmi, *tæmandi leit* og *gráðug reiknirit*.

Almennar nálganir lausna

- ▶ Þegar við leysum dæmi í keppnisforritun notumst við oftast við eina af eftirfarandi aðferðum:
 - ▶ *Ad hoc*.
 - ▶ *Tæmandi leit* eða *ofbeldis aðferðin* (e. *complete search*, *brute force*),
 - ▶ *Gráðug reiknirit* (e. *greedy algorithms*),
 - ▶ *Deila og drottna* (e. *divide and conquer*),
 - ▶ *Kvik bestun* (e. *dynamic programming*).
- ▶ Í síðustu vikum fjölluðum við um *Ad hoc* dæmi, *tæmandi leit* og *gráðug reiknirit*.
- ▶ Í þessari viku fjöllum við um *deila og drottna reiknirit* og *kvika bestun*.

Deila og drottna

- ▶ Sum dæmi má endurkvæmt skipta upp þangað til þau verða fáfengileg.

Deila og drottna

- ▶ Sum dæmi má endurkvæmt skipta upp þangað til þau verða fáfengileg.
- ▶ Síðan má líma fáfengilegu lausnirnar saman í heildarlausn í lokinn.

Deila og drottna

- ▶ Sum dæmi má endurkvæmt skipta upp þangað til þau verða fáfengileg.
- ▶ Síðan má líma fáfengilegu lausnirnar saman í heildarlausn í lokinn.
- ▶ Slík reiknirit kallast *deila og drottna* reiknirit.

Deila og drottna

- ▶ Sum dæmi má endurkvæmt skipta upp þangað til þau verða fáfengileg.
- ▶ Síðan má líma fáfengilegu lausnirnar saman í heildarlausn í lokinn.
- ▶ Slík reiknirit kallast *deila og drottna* reiknirit.
- ▶ Þessi flokkur er sjaldgæfastur.

Deila og drottna

- ▶ Sum dæmi má endurkvæmt skipta upp þangað til þau verða fáfengileg.
- ▶ Síðan má líma fáfengilegu lausnirnar saman í heildarlausn í lokinn.
- ▶ Slík reiknirit kallast *deila og drottna* reiknirit.
- ▶ Þessi flokkur er sjaldgæfastur.
- ▶ Það eru þó mörg þekkt reiknirit sem nýta sér deila og drottna.

Deila og drottna, þekkt dæmi

- ▶ Mergesort.

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).
- ▶ Þriðjungaleit (e. *ternary search*).

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).
- ▶ Þriðjungaleit (e. *ternary search*).
- ▶ Margföldunar reiknirit Karatsuba.

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).
- ▶ Þriðjungaleit (e. *ternary search*).
- ▶ Margföldunar reiknirit Karatsuba.
- ▶ Margföldunar reiknirit Strassen.

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).
- ▶ Þriðjungaleit (e. *ternary search*).
- ▶ Margföldunar reiknirit Karatsuba.
- ▶ Margföldunar reiknirit Strassen.
- ▶ Nálægustu punktar í plani.

Deila og drottna, þekkt dæmi

- ▶ Mergesort.
- ▶ Helmingunarleit (e. *binary search*).
- ▶ Þriðjungaleit (e. *ternary search*).
- ▶ Margföldunar reiknirit Karatsuba.
- ▶ Margföldunar reiknirit Strassen.
- ▶ Nálægustu punktar í plani.
- ▶ Fourier ummyndun (e. *fast Fourier transform* (FFT)).

Merge

- Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.
- ▶ Skeytum svo því sem er eftir aftan á c .

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.
- ▶ Skeytum svo því sem er eftir aftan á c .
- ▶ Nú inniheldur c þau stök sem voru í a og b áður.

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.
- ▶ Skeytum svo því sem er eftir aftan á c .
- ▶ Nú inniheldur c þau stök sem voru í a og b áður.
- ▶ Einnig er c raðaður.

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.
- ▶ Skeytum svo því sem er eftir aftan á c .
- ▶ Nú inniheldur c þau stök sem voru í a og b áður.
- ▶ Einnig er c raðaður.
- ▶ Ef fjöldi staka í a og b er n þá er þetta $\mathcal{O}(\quad)$.

Merge

- ▶ Gerum ráð fyrir að við séum með tvo raðaða lista, a og b .
- ▶ Búum til nýjan, tóman lista c .
- ▶ Berum saman fremstu stök a og b og tökum minna stakið og setjum aftast í c .
- ▶ Endurtökum þangað til a eða b er tómur.
- ▶ Skeytum svo því sem er eftir aftan á c .
- ▶ Nú inniheldur c þau stök sem voru í a og b áður.
- ▶ Einnig er c raðaður.
- ▶ Ef fjöldi staka í a og b er n þá er þetta $\mathcal{O}(n)$.

```
a = [1, 2, 5, 6, 8, 9]
b = [0, 3, 4, 7, 10]
c = []
```

```
a = [1, 2, 5, 6, 8, 9]
b = [3, 4, 7, 10]
c = [0]
```



```
a = [2, 5, 6, 8, 9]
b = [3, 4, 7, 10]
c = [0, 1]
```

```
a = [5, 6, 8, 9]
b = [3, 4, 7, 10]
c = [0, 1, 2]
```

```
a = [5, 6, 8, 9]  
b = [4, 7, 10]  
c = [0, 1, 2, 3]
```

```
a = [5, 6, 8, 9]
b = [7, 10]
c = [0, 1, 2, 3, 4]
```

```
a = [6, 8, 9]  
b = [7, 10]  
c = [0, 1, 2, 3, 4, 5]
```

```
a = [8, 9]
b = [7, 10]
c = [0, 1, 2, 3, 4, 5, 6]
```

```
a = [8, 9]  
b = [10]  
c = [0, 1, 2, 3, 4, 5, 6, 7]
```

```
a = [9]  
b = [10]  
c = [0, 1, 2, 3, 4, 5, 6, 7, 8]
```



```
a = []  
b = [10]  
c = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
a = []  
b = []  
c = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Mergesort

- ▶ Við getum notað þessa aðferð til að raða almennum lista.

Mergesort

- ▶ Við getum notað þessa aðferð til að raða almennum lista.
- ▶ Við skiptum listanum okkar í tvo jafna hluta og köllum endurkvæmt á fallið okkar þangað til við erum með tómann lista.

Mergesort

- ▶ Við getum notað þessa aðferð til að raða almennum lista.
- ▶ Við skiptum listanum okkar í tvo jafna hluta og köllum endurkvæmt á fallið okkar þangað til við erum með tómann lista.
- ▶ Á leiðinni upp úr endurkvæmninni sameinum við svo helmingana eins og rætt var á undan.

```
1 #include <stdio.h>
2 #include <time.h>
3 #include <stdlib.h>
4 #include <assert.h>
5
6 void merge(int* a, int l, int m, int r)
7 {
8     int i = l, j = m, b[r - l], c = 0;
9     while (i < m && j < r)
10     {
11         if (a[j] < a[i]) b[c++] = a[j++];
12         else b[c++] = a[i++];
13     }
14     while (i < m) b[c++] = a[i++];
15     while (j < r) b[c++] = a[j++];
16     for (i = l; i < r; i++) a[i] = b[i - l];
17 }
18
19 void mergesort(int* a, int l, int r)
20 {
21     if (r - l < 2) return;
22     int m = (l + r)/2;
23     mergesort(a, l, m), mergesort(a, m, r);
24     merge(a, l, m, r);
25 }
26
27 int main()
28 {
29     srand(time(NULL));
30     int i, n;
31     scanf("%d", &n);
32     int a[n];
33     for (i = 0; i < n; i++) a[i] = rand()%(10*n);
34     mergesort(a, 0, n);
35     for (i = 0; i < n - 1; i++) assert(a[i] <= a[i + 1]);
36     return 0;
37 }
```

Mergesort

- ▶ Mergesort er sígilt dæmi um deila og drottna reiknirit.

Mergesort

- ▶ Mergesort er sígilt dæmi um deila og drottna reiknirit.
- ▶ Við helmingum alltaf listann og tökum svo saman í línulegum tíma.

Mergesort

- ▶ Mergesort er sígilt dæmi um deila og drottna reiknirit.
- ▶ Við helmingum alltaf listann og tökum svo saman í línulegum tíma.
- ▶ Hvert stak kemur fyrir í $\mathcal{O}(\log n)$ sameiningum, svo reikniritið er $\mathcal{O}(\quad)$.

Mergesort

- ▶ Mergesort er sígilt dæmi um deila og drottna reiknirit.
- ▶ Við helmingum alltaf listann og tökum svo saman í línulegum tíma.
- ▶ Hvert stak kemur fyrir í $\mathcal{O}(\log n)$ sameiningum, svo reikniritið er $\mathcal{O}(n \log n)$.

Mergesort

- ▶ Mergesort er sígilt dæmi um deila og drottna reiknirit.
- ▶ Við helmingum alltaf listann og tökum svo saman í línulegum tíma.
- ▶ Hvert stak kemur fyrir í $\mathcal{O}(\log n)$ sameiningum, svo reikniritið er $\mathcal{O}(n \log n)$.
- ▶ Þetta er mjög algeng tímaflækja í deila og drottna reikniritum.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.
- ▶ Gerum ráð fyrir að við viljum finna t í listanum.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.
- ▶ Gerum ráð fyrir að við viljum finna t í listanum.
- ▶ Látum $m = \lfloor n/2 \rfloor$.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.
- ▶ Gerum ráð fyrir að við viljum finna t í listanum.
- ▶ Látum $m = \lfloor n/2 \rfloor$.
- ▶ Ef m -ta stakið í a er stærra en t þá getur t ekki verið í seinni helming listans.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.
- ▶ Gerum ráð fyrir að við viljum finna t í listanum.
- ▶ Látum $m = \lfloor n/2 \rfloor$.
- ▶ Ef m -ta stakið í a er stærra en t þá getur t ekki verið í seinni helming listans.
- ▶ Ef m -ta stakið í a er minna en t þá getur t ekki verið í fyrri helming listans.

Helmingunarleit

- ▶ Helmingunarleit er yfirleitt sett fram sem leit í röðuðum lista.
- ▶ Skoðum það fyrst og alhæfum svo.
- ▶ Látum a vera raðaðan lista af n tölum.
- ▶ Gerum ráð fyrir að við viljum finna t í listanum.
- ▶ Látum $m = \lfloor n/2 \rfloor$.
- ▶ Ef m -ta stakið í a er stærra en t þá getur t ekki verið í seinni helming listans.
- ▶ Ef m -ta stakið í a er minna en t þá getur t ekki verið í fyrri helming listans.
- ▶ Svo við getum útilokað helming listans í hverri ítrun.

```

1  #include <stdio.h>
2
3  int bs(int* a, int t, int l, int r)
4  {
5      if (r - l == 1) return a[l] == t ? l : -1;
6      int m = (l + r)/2;
7      if (a[m] <= t) return bs(a, t, m, r);
8      else return bs(a, t, l, m);
9  }
10
11 int main()
12 {
13     int i, n, q, x;
14     scanf("%d%d", &n, &q);
15     int a[n];
16     for (i = 0; i < n; i++) scanf("%d", &a[i]);
17     while (q-- != 0)
18     {
19         scanf("%d", &x);
20         printf("%d\n", bs(a, x, 0, n));
21     }
22     return 0;
23 }

```

- ▶ Helmingunarleit er $\mathcal{O}(\quad)$, þar sem við helmingum stærð listans í hverri ítrun.

- ▶ Helmingunarleit er $\mathcal{O}(\log n)$, þar sem við helmingum stærð listans í hverri ítrun.

- ▶ Helmingunarleit er $\mathcal{O}(\log n)$, þar sem við helmingum stærð listans í hverri ítrun.
- ▶ Góð æfing í helmingurleit er að útfæra leitina þannig að hún skili vísi á fyrstu (eða síðustu) endurtekningu staksins.

- ▶ Helmingunarleit er $\mathcal{O}(\log n)$, þar sem við helmingum stærð listans í hverri ítrun.
- ▶ Góð æfing í helmingurleit er að útfæra leitina þannig að hún skili vísi á fyrstu (eða síðustu) endurtekningu staksins.
- ▶ Slíkar útgáfur að helmingunarleit nýtast þegar við förum að nota helmingurleit í almennari mynd.

- Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.

- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?

- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.

- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.
- ▶ En hvernig finnum við slíkt x ?

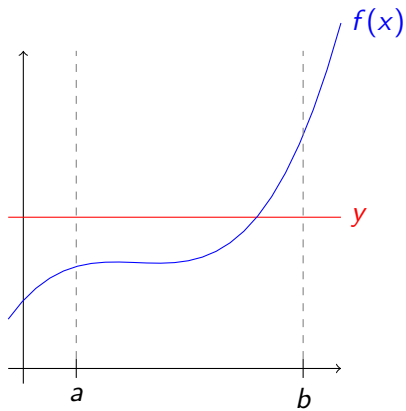
- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.
- ▶ En hvernig finnum við slíkt x ?
- ▶ Við getum notað nákvæmlega sömu hugmynd og í helmingurleit í lista.

- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.
- ▶ En hvernig finnum við slíkt x ?
- ▶ Við getum notað nákvæmlega sömu hugmynd og í helmingurleit í lista.
- ▶ Látum $m = (a + b)/2$.

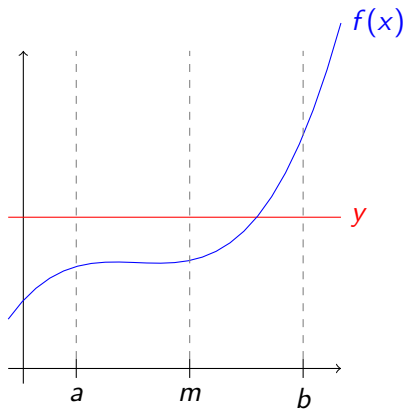
- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.
- ▶ En hvernig finnum við slíkt x ?
- ▶ Við getum notað nákvæmlega sömu hugmynd og í helmingurleit í lista.
- ▶ Látum $m = (a + b)/2$.
- ▶ Ef $f(m) > t$ þá þarf $x \in [a, m]$.

- ▶ Gerum ráð fyrir að við séum með vaxandi fall $f: [a, b] \rightarrow \mathbb{R}$ og $y \in \mathbb{R}$.
- ▶ Getum við fundið $x \in [a, b]$ þannig að $f(x) = y$?
- ▶ Slíkt x þarf ekki að vera til, en ef f er samfelld er það til ef $y \in [f(a), f(b)]$.
- ▶ En hvernig finnum við slíkt x ?
- ▶ Við getum notað nákvæmlega sömu hugmynd og í helmingurleit í lista.
- ▶ Látum $m = (a + b)/2$.
- ▶ Ef $f(m) > t$ þá þarf $x \in [a, m]$.
- ▶ Ef $f(m) < t$ þá þarf $x \in [m, b]$.

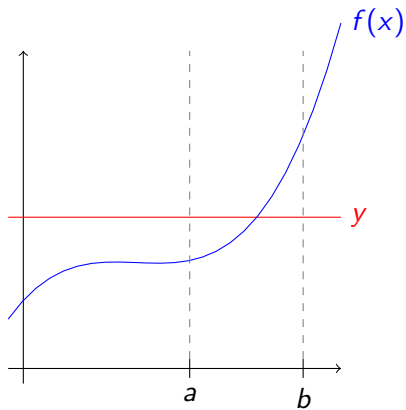
Sýnidæmi



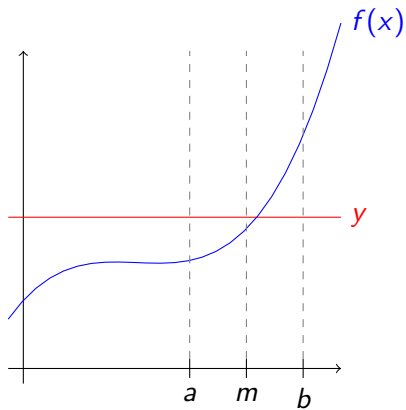
Sýnidæmi



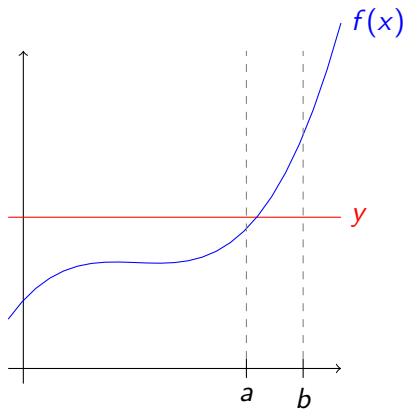
Sýnidæmi



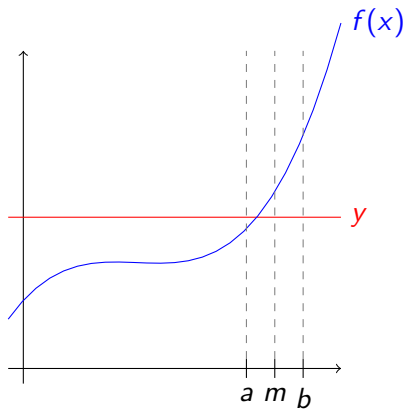
Sýnidæmi



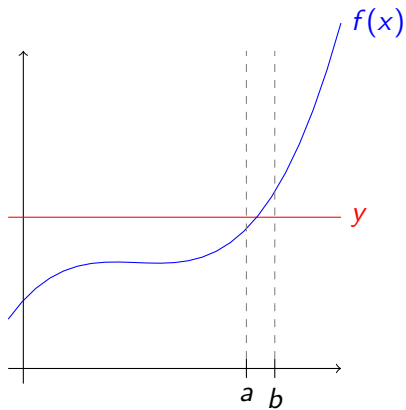
Sýnidæmi



Sýnidæmi



Sýnidæmi



- ▶ Takið eftir við munum ekki beint finna $x \in [a, b]$ þannig að $f(x) = y$.

- ▶ Takið eftir við munum ekki beint finna $x \in [a, b]$ þannig að $f(x) = y$.
- ▶ Það sem við finnum er $x \in [a, b]$ þannig að $|f(x) - y| < \varepsilon$, fyrir hvaða ε sem vera skal.

- ▶ Takið eftir við munum ekki beint finna $x \in [a, b]$ þannig að $f(x) = y$.
- ▶ Það sem við finnum er $x \in [a, b]$ þannig að $|f(x) - y| < \varepsilon$, fyrir hvaða ε sem vera skal.
- ▶ Það er þó aldrei ætlast til annars í keppnisforritun og skekkjan er alltaf gefin í úttakslýsingu dæma.

- ▶ Við getum alhæft frekar.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.
- ▶ Þú átt k ketti og n bæli fyrir kettina þína, með $k \leq n$.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.
- ▶ Þú átt k ketti og n bæli fyrir kettina þína, með $k \leq n$.
- ▶ Öll bælin eru staðsett á gangi í íbúðinni þinni.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.
- ▶ Þú átt k ketti og n bæli fyrir kettina þína, með $k \leq n$.
- ▶ Öll bælin eru staðsett á gangi í íbúðinni þinni.
- ▶ Ganginum má lýsa sem talnalínu og staðsetningar kattabælanna eru þá tölurnar $0 < x_1 < x_2 < \dots < x_n < 10^9$ á talnalínunni.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.
- ▶ Þú átt k ketti og n bæli fyrir kettina þína, með $k \leq n$.
- ▶ Öll bælin eru staðsett á gangi í íbúðinni þinni.
- ▶ Ganginum má lýsa sem talnalínu og staðsetningar kattabælanna eru þá tölurnar $0 < x_1 < x_2 < \dots < x_n < 10^9$ á talnalínunni.
- ▶ En kettir eru einfatar svo þeir vilja hafa sem mesta fjarlægð í næsta kött.

- ▶ Við getum alhæft frekar.
- ▶ Tökum dæmi.
- ▶ Þú átt k ketti og n bæli fyrir kettina þína, með $k \leq n$.
- ▶ Öll bælin eru staðsett á gangi í íbúðinni þinni.
- ▶ Ganginum má lýsa sem talnalínu og staðsetningar kattabælanna eru þá tölurnar $0 < x_1 < x_2 < \dots < x_n < 10^9$ á talnalínunni.
- ▶ En kettir eru einfatar svo þeir vilja hafa sem mesta fjarlægð í næsta kött.
- ▶ Þú átt að raða köttum á bælin þannig að nálægustu kettirnir eru sem lengst frá hvorum öðrum.

- ▶ Þetta dæmi má leysa með helmingunarleit.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?

- ▶ Petta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Petta dæmi má leysa gráðugt.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Þetta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .

- ▶ Petta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Petta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .
- ▶ Við megum þá ekki setja kött á bæli sem liggja í $[x_1, x_1 + r]$.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Þetta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .
- ▶ Við megum þá ekki setja kött á bæli sem liggja í $[x_1, x_1 + r]$.
- ▶ Útilokum þau og veljum minnsta bælið sem er eftir, og endurtökum.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Þetta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .
- ▶ Við megum þá ekki setja kött á bæli sem liggja í $[x_1, x_1 + r]$.
- ▶ Útilokum þau og veljum minnsta bælið sem er eftir, og endurtökum.
- ▶ Ef við komum fyrir k , eða fleiri, köttum svona þá er svarið við nýju spurningunni “já”, en annars “nei”.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Þetta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .
- ▶ Við megum þá ekki setja kött á bæli sem liggja í $[x_1, x_1 + r]$.
- ▶ Útilokum þau og veljum minnsta bælið sem er eftir, og endurtökum.
- ▶ Ef við komum fyrir k , eða fleiri, köttum svona þá er svarið við nýju spurningunni “já”, en annars “nei”.
- ▶ Þetta tekur $\mathcal{O}(\quad)$.

- ▶ Þetta dæmi má leysa með helmingunarleit.
- ▶ Byrjum á að svara annari spurningu:
 - ▶ Er hægt að raða köttunum þannig að nálægustu kettirnir séu að minnsta kosti með fjarlægð r ?
- ▶ Þetta dæmi má leysa gráðugt.
- ▶ Við töpum aldrei á því að setja kött á bælið staðsett á x_1 .
- ▶ Við megum þá ekki setja kött á bæli sem liggja í $[x_1, x_1 + r]$.
- ▶ Útilokum þau og veljum minnsta bælið sem er eftir, og endurtökum.
- ▶ Ef við komum fyrir k , eða fleiri, köttum svona þá er svarið við nýju spurningunni “já”, en annars “nei”.
- ▶ Þetta tekur $\mathcal{O}(n + k)$.

- Tökum þó eftir að ef við komum fyrir öllum köttunum með fjarlægð r_0 þá gerum við það líka fyrir $r < r_0$.

- ▶ Tökum þó eftir að ef við komum fyrir öllum köttunum með fjarlægð r_0 þá gerum við það líka fyrir $r < r_0$.
- ▶ Skilgreinum fall

$$f(r) = \begin{cases} 1, & \text{ef koma má fyrir } k, \\ & \text{eða fleiri, köttum með fjarlægð } r \\ 0, & \text{annars.} \end{cases}$$

- ▶ Tökum þó eftir að ef við komum fyrir öllum köttunum með fjarlægð r_0 þá gerum við það líka fyrir $r < r_0$.
- ▶ Skilgreinum fall

$$f(r) = \begin{cases} 1, & \text{ef koma má fyrir } k, \\ & \text{eða fleiri, köttum með fjarlægð } r \\ 0, & \text{annars.} \end{cases}$$

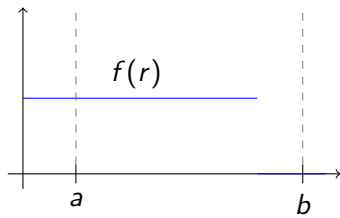
- ▶ Við getum nú umorðað upprunarlega dæmið sem: “Finnið stærsta r þannig að $f(r) = 1$ ”.

- ▶ Tökum þó eftir að ef við komum fyrir öllum köttunum með fjarlægð r_0 þá gerum við það líka fyrir $r < r_0$.
- ▶ Skilgreinum fall

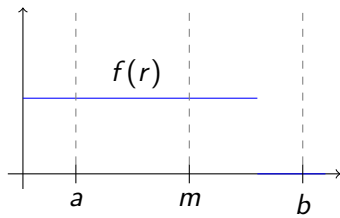
$$f(r) = \begin{cases} 1, & \text{ef koma má fyrir } k, \\ & \text{eða fleiri, köttum með fjarlægð } r \\ 0, & \text{annars.} \end{cases}$$

- ▶ Við getum nú umorðað upprunarlega dæmið sem: “Finnið stærsta r þannig að $f(r) = 1$ ”.
- ▶ En nú er fallið f minnkandi (samkvæmt efsta punktinum á glærunni), svo við getum fundið slíkt gildi með helmingunar leit.

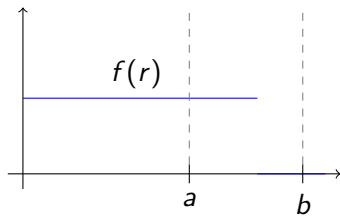
Sýnidæmi



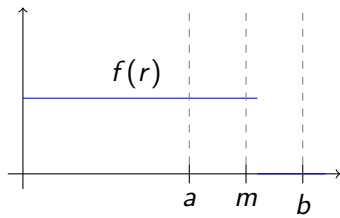
Sýnidæmi



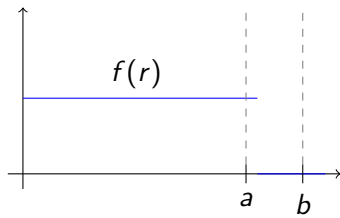
Sýnidæmi



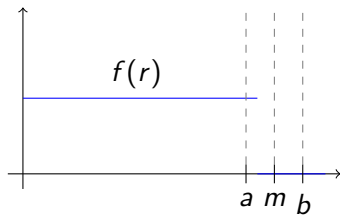
Sýnidæmi



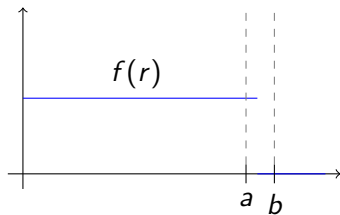
Sýnidæmi



Sýnidæmi



Sýnidæmi



- ▶ Ef við látum $M = 10^9/\varepsilon$, þar sem ε er leyfileg skekkja í dæminu, þá er lausnin $\mathcal{O}(\quad)$.

- ▶ Ef við látum $M = 10^9/\varepsilon$, þar sem ε er leyfileg skekkja í dæminu, þá er lausnin $\mathcal{O}((k + n) \log M)$.

- ▶ Ef við látum $M = 10^9/\varepsilon$, þar sem ε er leyfileg skekkja í dæminu, þá er lausnin $\mathcal{O}((k + n) \log M)$.
- ▶ Hér gerum við ekki ráð fyrir að svarið sé heiltala.

- ▶ Ef við látum $M = 10^9/\varepsilon$, þar sem ε er leyfileg skekkja í dæminu, þá er lausnin $\mathcal{O}((k + n) \log M)$.
- ▶ Hér gerum við ekki ráð fyrir að svarið sé heiltala.
- ▶ Ef við gerum ráð fyrir því verður tímaflækjan eins, nema með $M = 10^9$.

- ▶ Það sem við gerðum í raun var að breyta dæminu úr “finnið minnsta/stærsta gildið þannig að...” yfir í “tökum ákveðið gildi og athugum hvort að...”.

- ▶ Það sem við gerðum í raun var að breyta dæminu úr “finnið minnsta/stærsta gildið þannig að...” yfir í “tökum ákveðið gildi og athugum hvort að...”.
- ▶ Þetta er algengasta notkunin á helmingunarleit í keppnisforritun.

- ▶ Það sem við gerðum í raun var að breyta dæminu úr “finnið minnsta/stærsta gildið þannig að...” yfir í “tökum ákveðið gildi og athugum hvort að...”.
- ▶ Þetta er algengasta notkunin á helmingunarleit í keppnisforritun.
- ▶ Algengt er helmingurleit af þessum toga sé hluti af erfiðum dæmum.

Priðjungaleit

- Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.
- ▶ Hvernig finnum við útgildi (há- og lággildi) fallsins?

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.
- ▶ Hvernig finnum við útgildi (há- og lággildi) fallsins?
- ▶ Auðvelt er að finna hágildi.

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.
- ▶ Hvernig finnum við útgildi (há- og lággildi) fallsins?
- ▶ Auðvelt er að finna hágildi.
- ▶ Gerum ráð fyrir að $f(a) \leq f(b)$.

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.
- ▶ Hvernig finnum við útgildi (há- og lággildi) fallsins?
- ▶ Auðvelt er að finna hágildi.
- ▶ Gerum ráð fyrir að $f(a) \leq f(b)$.
- ▶ Fyrir öll $x \in [a, b]$ gildir þá að

$$\begin{aligned} f(x) &= f(at + (1 - t)b) \leq tf(a) + (1 - t)f(b) \\ &\leq tf(b) + (1 - t)f(b) = f(b), \end{aligned}$$

með $t = (x - b)/(a - b)$.

Priðjungaleit

- ▶ Gerum ráð fyrir að við séum með kúpt fall $f: [a, b] \rightarrow \mathbb{R}$.
- ▶ Munið að fall er kúpt ef
$$f(tx_1 + (1 - t)x_2) \leq tf(x_1) + (1 - t)f(x_2),$$
fyrir öll $t \in [0, 1]$ og $x_1, x_2 \in [a, b]$.
- ▶ Hvernig finnum við útgildi (há- og lággildi) fallsins?
- ▶ Auðvelt er að finna hágildi.
- ▶ Gerum ráð fyrir að $f(a) \leq f(b)$.
- ▶ Fyrir öll $x \in [a, b]$ gildir þá að

$$\begin{aligned} f(x) &= f(at + (1 - t)b) \leq tf(a) + (1 - t)f(b) \\ &\leq tf(b) + (1 - t)f(b) = f(b), \end{aligned}$$

með $t = (x - b)/(a - b)$.

- ▶ Svo hágildi fæst í endapunktunum a eða b .

Priðjungaleit

- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.

Priðjungaleit

- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.

Priðjungaleit

- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.
- ▶ Við skoðum svo fallgildin $f(m_1)$ og $f(m_2)$.

Priðjungaleit

- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.
- ▶ Við skoðum svo fallgildin $f(m_1)$ og $f(m_2)$.
- ▶ Ef $f(m_1) < f(m_2)$ þá getur lággildið ekki legið á bilinu $[m_2, b]$, svo við getum útilokað það bil í leitinni.

Priðjungaleit

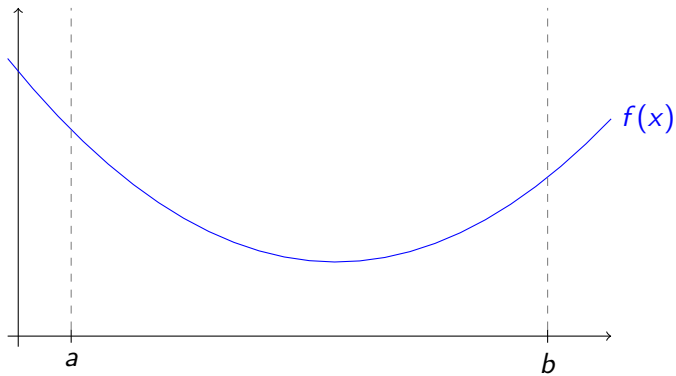
- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.
- ▶ Við skoðum svo fallgildin $f(m_1)$ og $f(m_2)$.
- ▶ Ef $f(m_1) < f(m_2)$ þá getur lággildið ekki legið á bilinu $[m_2, b]$, svo við getum útilokað það bil í leitinni.
- ▶ Ef $f(m_2) < f(m_1)$ þá getur lággildið ekki legið á bilinu $[a, m_1]$, svo við getum útilokað það bil í leitinni.

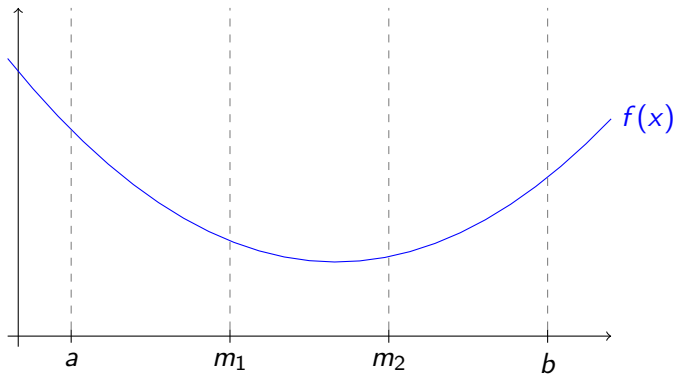
Priðjungaleit

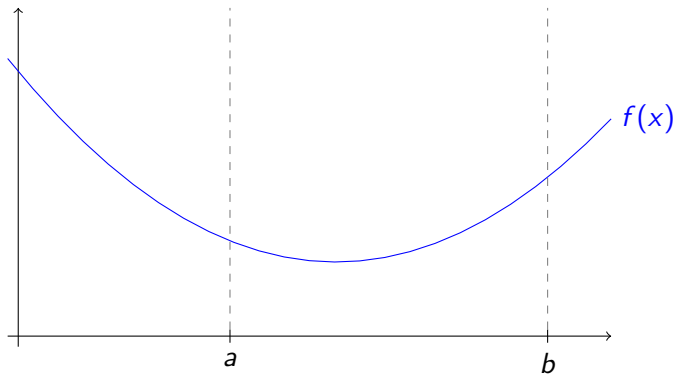
- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.
- ▶ Við skoðum svo fallgildin $f(m_1)$ og $f(m_2)$.
- ▶ Ef $f(m_1) < f(m_2)$ þá getur lággildið ekki legið á bilinu $[m_2, b]$, svo við getum útilokað það bil í leitinni.
- ▶ Ef $f(m_2) < f(m_1)$ þá getur lággildið ekki legið á bilinu $[a, m_1]$, svo við getum útilokað það bil í leitinni.
- ▶ Ef $f(m_2) = f(m_1)$ þá þarf lággildið að liggja á bilinu $[m_1, m_2]$.

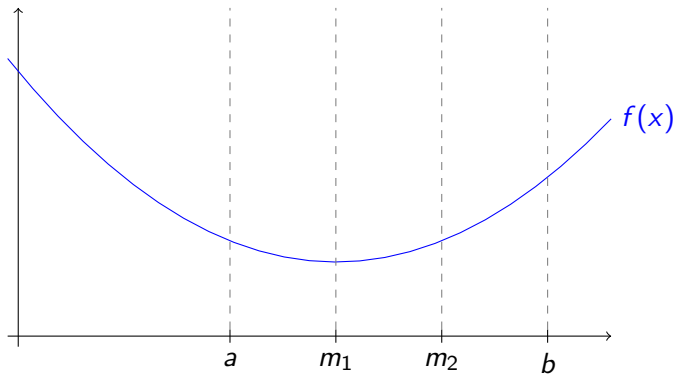
Priðjungaleit

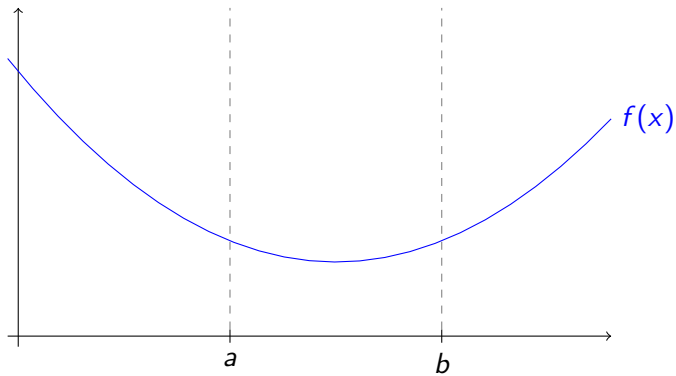
- ▶ Við getum fundið lággildið tölulega, svipað og með helmingunarleit.
- ▶ Við veljum punkta $m_1, m_2 \in [a, b]$ þannig að bilin $[a, m_1]$, $[m_1, m_2]$ og $[m_2, b]$ séu jafn löng.
- ▶ Við skoðum svo fallgildin $f(m_1)$ og $f(m_2)$.
- ▶ Ef $f(m_1) < f(m_2)$ þá getur lággildið ekki legið á bilinu $[m_2, b]$, svo við getum útilokað það bil í leitinni.
- ▶ Ef $f(m_2) < f(m_1)$ þá getur lággildið ekki legið á bilinu $[a, m_1]$, svo við getum útilokað það bil í leitinni.
- ▶ Ef $f(m_2) = f(m_1)$ þá þarf lággildið að liggja á bilinu $[m_1, m_2]$.
- ▶ Þetta stafar allt af því að kúpt föll taka hágildi í öðru hvorum endapunkta sinna.

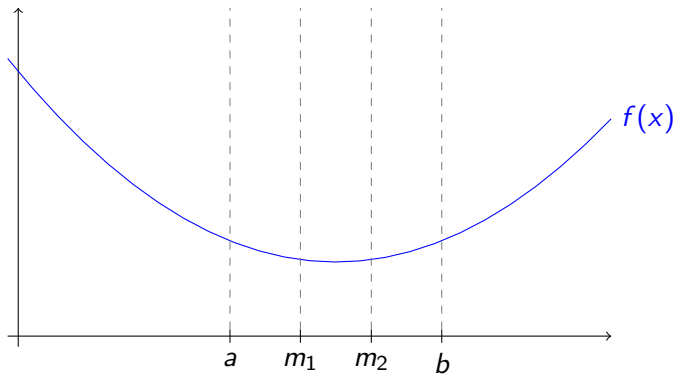


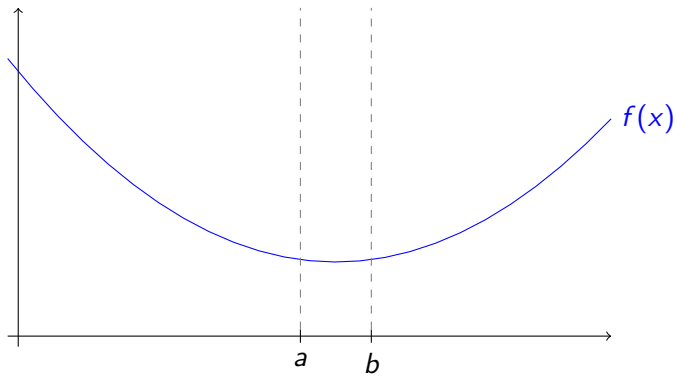


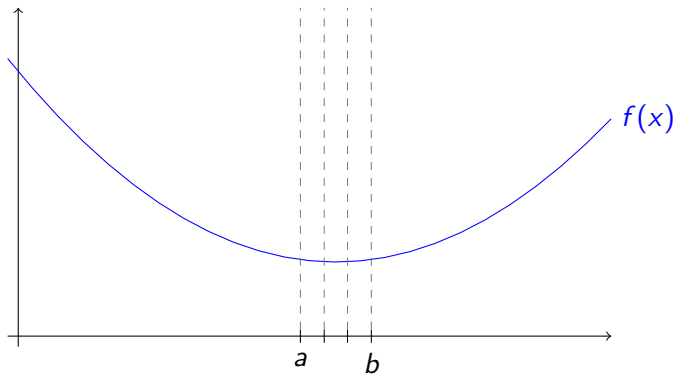


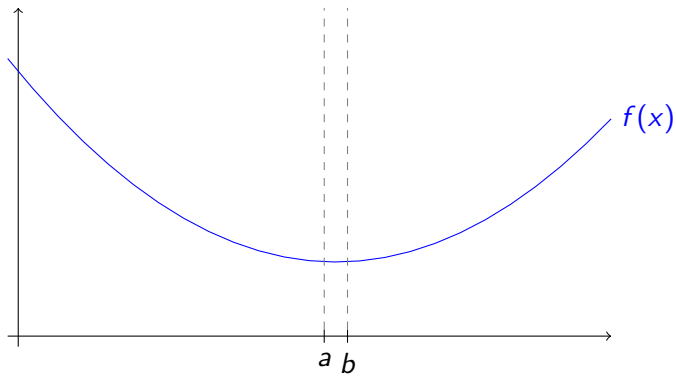












- ▶ Við notum okkur oft að tvídiffranlegt fall er kúpt ef og aðeins ef önnur afleiðan er jákvæð.

- ▶ Við notum okkur oft að tvídifffranlegt fall er kúpt ef og aðeins ef önnur afleiðan er jákvæð.
- ▶ Umfjöllunin okkar heimfærist á eðlilegan hátt yfir á hvelf föll.

- ▶ Við notum okkur oft að tvídifffranlegt fall er kúpt ef og aðeins ef önnur afleiðan er jákvæð.
- ▶ Umfjöllunin okkar heimfærist á eðlilegan hátt yfir á hvelf föll.
- ▶ Þriðjungaleit er algeng í rúmfræði því Evklíðska firðinni er kúpt.

```
1 #include <stdio.h>
2 #define EPS 1e-9
3
4 double f(double x)
5 {
6     return 5.0 - 1.2*x + 0.1*x*x;
7 }
8
9 int main()
10 {
11     double a = 1.0, b = 10.0, m1, m2;
12     while (b - a > EPS)
13     {
14         m1 = (a + a + b)/3.0;
15         m2 = (a + b + b)/3.0;
16         if (f(m1) > f(m2)) a = m1;
17         else b = m2;
18     }
19     printf("f(%.2f) = %.2f\n", a, f(a));
20     return 0;
21 }
```


- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.
- ▶ Eftir einhvern fjölda daga þarft þú svo að kaupa 100 danskar krónur til að borga lánið, ásamt því að borga K íslenskar krónur á dag í lánakostnað.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.
- ▶ Eftir einhvern fjölda daga þarft þú svo að kaupa 100 danskar krónur til að borga lánið, ásamt því að borga K íslenskar krónur á dag í lánaðkostnað.
- ▶ Hver er mesti fjöldi íslenska króna sem þú getur grætt?

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.
- ▶ Eftir einhvern fjölda daga þarft þú svo að kaupa 100 danskar krónur til að borga lánið, ásamt því að borga K íslenskar krónur á dag í lánaðkostnað.
- ▶ Hver er mesti fjöldi íslenska króna sem þú getur grætt?
- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, $1 \leq n \leq 10^5$ og $1 \leq k \leq 100$.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.
- ▶ Eftir einhvern fjölda daga þarft þú svo að kaupa 100 danskar krónur til að borga lánið, ásamt því að borga K íslenskar krónur á dag í lánaðkostnað.
- ▶ Hver er mesti fjöldi íslenska króna sem þú getur grætt?
- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, $1 \leq n \leq 10^5$ og $1 \leq k \leq 100$.
- ▶ Næsta lína inniheldur n heiltölur $1 \leq x_1, x_2, \dots, x_n \leq 10^5$.

- ▶ Tökum eitt hefðbundið sýnidæmi í lokinn.
- ▶ Þú vilt skortselja gjaldmiðil, vitandi framtíðargengi, þannig að þú græðir sem mestan pening.
- ▶ Nánar, þú ætlar að fá lánaðar 100 danskar krónur á einhverjum degi og skipta þeim um leið í íslenskar krónur.
- ▶ Eftir einhvern fjölda daga þarft þú svo að kaupa 100 danskar krónur til að borga lánið, ásamt því að borga K íslenskar krónur á dag í lánaðkostnað.
- ▶ Hver er mesti fjöldi íslenska króna sem þú getur grætt?
- ▶ Fyrsta lína inntaksins inniheldur tvær tölur, $1 \leq n \leq 10^5$ og $1 \leq k \leq 100$.
- ▶ Næsta lína inniheldur n heiltölur $1 \leq x_1, x_2, \dots, x_n \leq 10^5$.
- ▶ Hér tákna x_i fjölda íslenska króna sem ein dönsk króna kostar á i -ta degi.

- ▶ Skoðum sýniinntök.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í fyrra tilfellinu viljum við taka lánið á fyrsta degi og borga það á síðasta degi.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í fyrra tilfellinu viljum við taka lánið á fyrsta degi og borga það á síðasta degi.
- Við fáum $100 \cdot 1000 = 10^5$ íslenskar krónur á fyrst degi og borgum $100 \cdot 10 = 10^3$ íslenskar krónur á síðasta degi.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í fyrra tilfellinu viljum við taka lánið á fyrsta degi og borga það á síðasta degi.
- Við fáum $100 \cdot 1000 = 10^5$ íslenskar krónur á fyrst degi og borgum $100 \cdot 10 = 10^3$ íslenskar krónur á síðasta degi.
- Við borgum svo $5 \cdot 10 = 50$ íslenskar krónur í lánakostnað.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í fyrra tilfellinu viljum við taka lánið á fyrsta degi og borga það á síðasta degi.
- Við fáum $100 \cdot 1000 = 10^5$ íslenskar krónur á fyrst degi og borgum $100 \cdot 10 = 10^3$ íslenskar krónur á síðasta degi.
- Við borgum svo $5 \cdot 10 = 50$ íslenskar krónur í lánakostnað.
- Svo við endum með $10^5 - 10^3 - 50 = 98950$ íslenskar krónur.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í seinna tilfellinu viljum við taka lánið á fjórða degi og borga það á síðasta degi.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í seinna tilfellinu viljum við taka lánið á fjórða degi og borga það á síðasta degi.
- Við fáum $100 \cdot 103 = 10300$ íslenskar krónur á fyrst degi og borgum $100 \cdot 100 = 10^4$ íslenskar krónur á síðasta degi.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í seinna tilfellinu viljum við taka lánið á fjórða degi og borga það á síðasta degi.
- Við fáum $100 \cdot 103 = 10300$ íslenskar krónur á fyrst degi og borgum $100 \cdot 100 = 10^4$ íslenskar krónur á síðasta degi.
- Við borgum svo $2 \cdot 100 = 200$ íslenskar krónur í lánakostnað.

► Skoðum sýniinntök.

```
1 Sample input 1
2 5 10
3 1000 980 960 940 10
4
5
6 Sample input 2
7 5 100
8 100 100 100 103 100
```

```
Sample output 1
98950
```

```
Sample output 2
100
```

- Í seinna tilfellinu viljum við taka lánið á fjórða degi og borga það á síðasta degi.
- Við fáum $100 \cdot 103 = 10300$ íslenskar krónur á fyrst degi og borgum $100 \cdot 100 = 10^4$ íslenskar krónur á síðasta degi.
- Við borgum svo $2 \cdot 100 = 200$ íslenskar krónur í lánakostnað.
- Svo við endum með $10300 - 10^4 - 200 = 100$ íslenskar krónur.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.
- ▶ Gerum ráð fyrir að engin lánaðnaður sé greiddur síðasta.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.
- ▶ Gerum ráð fyrir að engin lánaðkostnaður sé greiddur síðasta.
- ▶ Það einfaldar reikninga og við getum alltaf bætt honum við eftir á.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.
- ▶ Gerum ráð fyrir að engin lánaðkostnaður sé greiddur síðasta.
- ▶ Það einfaldar reikninga og við getum alltaf bætt honum við eftir á.
- ▶ Táknum með $f(i, j)$ þann gróða (eða tap) sem fæst með því að taka lánið á i -ta degi og borga það á j -degi degi, og $g(i)$ vera gengið á i -ta degi.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.
- ▶ Gerum ráð fyrir að engin lánaðkostnaður sé greiddur síðasta.
- ▶ Það einfaldar reikninga og við getum alltaf bætt honum við eftir á.
- ▶ Táknum með $f(i, j)$ þann gróða (eða tap) sem fæst með því að taka lánið á i -ta degi og borga það á j -degi degi, og $g(i)$ vera gengið á i -ta degi.
- ▶ Við fáum nú að $f(i, j) = 100 \cdot g(i) - 100 \cdot g(j) - (j - i) \cdot k$.

- ▶ Til að nota deila og drottna þurfum við að taka eftir einu.
- ▶ Gerum ráð fyrir að engin lánaðkostnaður sé greiddur síðasta.
- ▶ Það einfaldar reikninga og við getum alltaf bætt honum við eftir á.
- ▶ Táknum með $f(i, j)$ þann gróða (eða tap) sem fæst með því að taka lánið á i -ta degi og borga það á j -degi degi, og $g(i)$ vera gengið á i -ta degi.
- ▶ Við fáum nú að $f(i, j) = 100 \cdot g(i) - 100 \cdot g(j) - (j - i) \cdot k$.
- ▶ Ef a , b og c eru heiltölur þannig að $1 \leq a < b < c \leq n$ þá fæst

$$\begin{aligned}
 f(a, b) + f(b, c) &= 100 \cdot (g(a) - g(b)) - (b - a) \cdot k \\
 &\quad + 100 \cdot (g(b) - g(c)) - (c - b) \cdot k \\
 &= 100 \cdot (g(a) - g(c)) - (c - a) \cdot k \\
 &= f(a, c).
 \end{aligned}$$

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.
 - ▶ Annar endapunktur bestu lausnarinnar liggur á $[1, m - 1]$ og hinn liggur á $[m, n]$.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.
 - ▶ Annar endapunktur bestu lausnarinnar liggur á $[1, m - 1]$ og hinn liggur á $[m, n]$.
- ▶ Fyrri tvö tilfelli má leysa með einfaldri endurkvæmni.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.
 - ▶ Annar endapunktur bestu lausnarinnar liggur á $[1, m - 1]$ og hinn liggur á $[m, n]$.
- ▶ Fyrri tvö tilfellin má leysa með einfaldri endurkvæmni.
- ▶ Fyrir síðasta tilfellið nýtum við gegnvirknina af fyrri glærunni.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.
 - ▶ Annar endapunktur bestu lausnarinnar liggur á $[1, m - 1]$ og hinn liggur á $[m, n]$.
- ▶ Fyrri tvö tilfellin má leysa með einfaldri endurkvæmni.
- ▶ Fyrir síðasta tilfellið nýtum við gegnvirknina af fyrri glærunni.
- ▶ Við viljum finna bestu lausnina sem liggur í gegnum m -ta stakið.

- ▶ Látum nú $m = \lfloor n/2 \rfloor$.
- ▶ Þá gildir eitt af þrennu, fyrir tiltekna lausn:
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[1, m - 1]$.
 - ▶ Báðir endapunktur bestu lausnarinnar liggja á $[m, n]$.
 - ▶ Annar endapunktur bestu lausnarinnar liggur á $[1, m - 1]$ og hinn liggur á $[m, n]$.
- ▶ Fyrri tvö tilfellin má leysa með einfaldri endurkvæmni.
- ▶ Fyrir síðasta tilfellið nýtum við gegnvirknina af fyrri glærunni.
- ▶ Við viljum finna bestu lausnina sem liggur í gegnum m -ta stakið.
- ▶ Gegnvirknin segir þó að okkur nægir að finna fyrst bestu lausnina sem endar í m -ta stakinu, finna svo bestu lausnina sem byrjar í m -ta stakinu og sameina svo lausnirnar.

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 typedef long long ll;
4 ll max(ll a, ll b) { if (a < b) return b; return a; }
5
6 ll foo(ll* a, ll l, ll r, ll k)
7 {
8     if (r - l < 5)
9     {
10         ll i, j, mx = 0;
11         for (i = l; i < r; i++) for (j = i + 1; j < r; j++)
12             mx = max(mx, 100*(a[i] - a[j]) - k*(j - i));
13         return mx;
14     }
15     ll m = (l + r)/2, i, j;
16     ll v1 = foo(a, l, m, k), v2 = foo(a, m, r, k), mx1 = 0, mx2 = 0;
17     for (i = l; i < m; i++) mx1 = max(mx1, 100*(a[i] - a[m]) - k*(m - i));
18     for (i = m; i < r; i++) mx2 = max(mx2, 100*(a[m] - a[i]) - k*(i - m));
19     return max(max(v1, v2), mx1 + mx2);
20 }
21
22 int main()
23 {
24     ll i, j;
25     int x, n, k;
26     scanf("%d%d", &n, &k);
27     ll a[n];
28     for (i = 0; i < n; i++)
29     {
30         scanf("%d", &x);
31         a[i] = x;
32     }
33     printf("%lld\n", max(0, foo(a, 0, n, k) - k));
34     return 0;
35 }

```

Rakningarvensl

- Talnarunan a_1, a_2, \dots kallast *k-ta stigs rakningarvensl* ef til er fall $f: \mathbb{R}^k \rightarrow \mathbb{R}$ þannig að

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

fyrir öll $n > k$.

Rakningarvensl

- ▶ Talnarunan a_1, a_2, \dots kallast *k-ta stigs rakningarvensl* ef til er fall $f: \mathbb{R}^k \rightarrow \mathbb{R}$ þannig að

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

fyrir öll $n > k$.

- ▶ Frægasta dæmið um rakningarvensl er *Fibonacci runan*.

Rakningarvensl

- ▶ Talnarunan a_1, a_2, \dots kallast k -ta stigs rakningarvensl ef til er fall $f: \mathbb{R}^k \rightarrow \mathbb{R}$ þannig að

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

fyrir öll $n > k$.

- ▶ Frægasta dæmið um rakningarvensl er *Fibonacci runan*.
- ▶ Hún er stigs rakningarvensl gefin með fallinu $f(x, y) = x + y$.

Rakningarvensl

- ▶ Talnarunan a_1, a_2, \dots kallast k -ta stigs rakningarvensl ef til er fall $f: \mathbb{R}^k \rightarrow \mathbb{R}$ þannig að

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

fyrir öll $n > k$.

- ▶ Frægasta dæmið um rakningarvensl er *Fibonacci runan*.
- ▶ Hún er annars stigs rakningarvensl gefin með fallinu $f(x, y) = x + y$.

Rakningarvensl

- ▶ Talnarunan a_1, a_2, \dots kallast k -ta stigs rakningarvensl ef til er fall $f: \mathbb{R}^k \rightarrow \mathbb{R}$ þannig að

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

fyrir öll $n > k$.

- ▶ Frægasta dæmið um rakningarvensl er *Fibonacci runan*.
- ▶ Hún er annars stigs rakningarvensl gefin með fallinu $f(x, y) = x + y$.
- ▶ Reikna má upp úr þessum venslum endurkvæmt.

```
1 #include <stdio.h>
2
3 int fib(int x)
4 {
5     if (x < 3) return 1;
6     return fib(x - 1) + fib(x - 2);
7 }
8
9 int main()
10 {
11     int n;
12     scanf("%d", &n);
13     printf("%d\n", fib(n));
14     return 0;
15 }
```


- ▶ Í hverju skrefi skiptist endurkvæmnin í tvennt svo þetta forrit hefur tímaflækju $\mathcal{O}(\quad)$.

- ▶ Í hverju skrefi skiptist endurkvæmnin í tvennt svo þetta forrit hefur tímaflækju $\mathcal{O}(2^n)$.

- ▶ Í hverju skrefi skiptist endurkvæmnin í tvennt svo þetta forrit hefur tímaflækju $\mathcal{O}(2^n)$.
- ▶ Við getum þó bætt þetta til muna með því að geyma niðurstöðuna úr hverju kalli.

- ▶ Í hverju skrefi skiptist endurkvæmnin í tvennt svo þetta forrit hefur tímaflækju $\mathcal{O}(2^n)$.
- ▶ Við getum þó bætt þetta til muna með því að geyma niðurstöðuna úr hverju kalli.
- ▶ Þá nægir að reikna hvert gildi einu sinni.

- ▶ Í hverju skrefi skiptist endurkvæmnin í tvennt svo þetta forrit hefur tímaflækju $\mathcal{O}(2^n)$.
- ▶ Við getum þó bætt þetta til muna með því að geyma niðurstöðuna úr hverju kalli.
- ▶ Þá nægir að reikna hvert gildi einu sinni.
- ▶ Þessi viðbót kallast *minnun* (e. *memoization*).

```
1 #include <stdio.h>
2 #define MAXN 1000000
3
4 int d[MAXN]; // Hér geymum við skilagildi fib(...).
5 // Ef d[i] = -1 þá eigum við eftir að reikna fib(i).
6 int fib(int x)
7 {
8     if (d[x] != -1) return d[x];
9     if (x < 2) return 1;
10    return d[x] = fib(x - 1) + fib(x - 2);
11 }
12
13 int main()
14 {
15     int n, i;
16     scanf("%d", &n);
17     for (i = 0; i < n; i++) d[i] = -1;
18     printf("%d\n", fib(n - 1));
19     return 0;
20 }
```

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.
- ▶ Við þurfum að reikna n gildi og hvert gildi má reikna í $\mathcal{O}(\quad)$ tíma, svo í heildina er forritið $\mathcal{O}(\quad)$.

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.
- ▶ Við þurfum að reikna n gildi og hvert gildi má reikna í $\mathcal{O}(1)$ tíma, svo í heildina er forritið $\mathcal{O}(n)$.

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.
- ▶ Við þurfum að reikna n gildi og hvert gildi má reikna í $\mathcal{O}(1)$ tíma, svo í heildina er forritið $\mathcal{O}(n)$.

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.
- ▶ Við þurfum að reikna n gildi og hvert gildi má reikna í $\mathcal{O}(1)$ tíma, svo í heildina er forritið $\mathcal{O}(n)$.
- ▶ Án minnunar náum við með erfiðum að reikna fertugust Fibonacci töluna (því eframatið $\mathcal{O}(2^n)$ mætti bæta ögn) en með minnun náum við hæglega að reikna milljónustu Fibonacci töluna (hún mun þó ekki einu sinni passa í 64 bita).

Ofansækin kvik bestun

- ▶ Nú reiknum við hvert gildi aðeins einu sinni.
- ▶ Við þurfum að reikna n gildi og hvert gildi má reikna í $\mathcal{O}(1)$ tíma, svo í heildina er forritið $\mathcal{O}(n)$.
- ▶ Án minnunar náum við með erfiðum að reikna fertugust Fibonacci töluna (því eframatið $\mathcal{O}(2^n)$ mætti bæta ögn) en með minnun náum við hæglega að reikna milljónustu Fibonacci töluna (hún mun þó ekki einu sinni passa í 64 bita).
- ▶ Ef lausnin okkar er endurkvæm með minnun kallast hún *ofansækin kvik bestun* (e. *top down dynamic programming*).

Neðansækin kvik bestun

- ▶ Það er þó lítið mál að breyta endurkvæmnu lausninni okkar í ítraða lausn.

Neðansækin kvik bestun

- ▶ Það er þó lítið mál að breyta endurkvæmnu lausninni okkar í ítraða lausn.
- ▶ Eina sem við þurfum að passa er að reikna gildin í vaxandi röð.

Neðansækin kvik bestun

- ▶ Það er þó lítið mál að breyta endurkvæmnu lausninni okkar í ítraða lausn.
- ▶ Eina sem við þurfum að passa er að reikna gildin í vaxandi röð.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n, i;
6     scanf("%d", &n);
7     int a[n];
8     a[0] = a[1] = 1;
9     for (i = 2; i < n; i++) a[i] = a[i - 1] + a[i - 2];
10    printf("%d\n", a[n - 1]);
11    return 0;
12 }
```

Neðansækin kvik bestun

- ▶ Það er þó lítið mál að breyta endurkvæmnu lausninni okkar í ítraða lausn.
- ▶ Eina sem við þurfum að passa er að reikna gildin í vaxandi röð.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n, i;
6     scanf("%d", &n);
7     int a[n];
8     a[0] = a[1] = 1;
9     for (i = 2; i < n; i++) a[i] = a[i - 1] + a[i - 2];
10    printf("%d\n", a[n - 1]);
11    return 0;
12 }
```

- ▶ Þegar ofansækin kvik bestunar lausn er útfærð með ítrun köllum við það *neðansækin kvik bestun* (e. *bottom up dymanic programming*).

- ▶ Í neðansækinni kvikri bestun byrjum við með grunntilfellin og smíðum flóknari lausnirnar út frá þeim.

- ▶ Í neðansækinni kvikri bestun byrjum við með grunntilfellin og smíðum flóknari lausnirnar út frá þeim.
- ▶ Í ofansækinni kvikri bestun brjótum við fyrst flóknu dæmin niður í smærri dæmi sem við vitum svarið við og reiknum svo út úr því.

- ▶ Í neðansækinni kvikri bestun byrjum við með grunntilfellin og smíðum flóknari lausnirnar út frá þeim.
- ▶ Í ofansækinni kvikri bestun brjótum við fyrst flóknu dæmin niður í smærri dæmi sem við vitum svarið við og reiknum svo út úr því.
- ▶ Ef endurkvæmnafallið okkar er háð k breytum þá segjum við að lausnin okkar sé k *víð kvik bestun*.

- ▶ Í neðansækinni kvikri bestun byrjum við með grunntilfellin og smíðum flóknari lausnirnar út frá þeim.
- ▶ Í ofansækinni kvikri bestun brjótum við fyrst flóknu dæmin niður í smærri dæmi sem við vitum svarið við og reiknum svo út úr því.
- ▶ Ef endurkvæmnafallið okkar er háð k breytum þá segjum við að lausnin okkar sé k víð kvik bestun.
- ▶ Ofansækin kvik bestun hentar þegar við erum að vinna með fleiri en eina vídd.

- ▶ Í neðansækinni kvikri bestun byrjum við með grunntilfellin og smíðum flóknari lausnirnar út frá þeim.
- ▶ Í ofansækinni kvikri bestun brjótum við fyrst flóknu dæmin niður í smærri dæmi sem við vitum svarið við og reiknum svo út úr því.
- ▶ Ef endurkvæmnafallið okkar er háð k breytum þá segjum við að lausnin okkar sé k víð kvik bestun.
- ▶ Ofansækin kvik bestun hentar þegar við erum að vinna með fleiri en eina vídd.
- ▶ Þá getur verið erfitt að ítra í gegnum stöðurnar í “rétttri röð”.

- ▶ Annar kostur ofansækinnar kvikrar bestunar er að lausnirnar geta verið nokkuð einsleitar.

- ▶ Annar kostur ofansækinna kvikrar bestunar er að lausnirnar geta verið nokkuð einsleitar.

```
1 #include <stdio.h>
2 #define MAXN 1000000
3
4 int d[MAXN];
5 int dp_lookup(int x)
6 {
7     if (d[x] != -1) return d[x];
8     if (/* Er þetta grunntilfelli? */)
9     {
10         /* Skila tilheyrandi grunnsvari */
11     }
12     /* Reikna d[x] */
13     return d[x];
14 }
15
16 int main()
17 {
18     int n, i;
19     scanf("%d", &n);
20     for (i = 0; i < MAXN; i++) d[i] = -1;
21     printf("%d\n", dp_lookup(n));
22     return 0;
23 }
```

Lengsta sameiginlega hlutruna

- ▶ Tökum annað dæmi.

Lengsta sameiginlega hlutruna

- ▶ Tökum annað dæmi.
- ▶ Látum $S = s_1 s_2 \dots s_n$ og $T = t_1 t_2 \dots t_m$ vera strengi af lengd n og m , þannig að $1 \leq n, m \leq 10^4$.

Lengsta sameiginlega hlutruna

- ▶ Tökum annað dæmi.
- ▶ Látum $S = s_1 s_2 \dots s_n$ og $T = t_1 t_2 \dots t_m$ vera strengi af lengd n og m , þannig að $1 \leq n, m \leq 10^4$.
- ▶ Hver er lengd lengsta strengs X þannig að hann sé hlutruna í bæði S og T ?

Lengsta sameiginlega hlutruna

- ▶ Tökum annað dæmi.
- ▶ Látum $S = s_1s_2\dots s_n$ og $T = t_1t_2\dots t_m$ vera strengi af lengd n og m , þannig að $1 \leq n, m \leq 10^4$.
- ▶ Hver er lengd lengsta strengs X þannig að hann sé hlutruna í bæði S og T ?
- ▶ Takið eftir að "12" og "13" eru hlutrunur í "123" en "21" er það ekki.

- ▶ Getum við sett upp dæmið með þægilegum rakningarvenslum?

- ▶ Getum við sett upp dæmið með þægilegum rakningarvenslum?
- ▶ Ef svo er þá getum við notað kvika bestun.

- ▶ Getum við sett upp dæmið með þægilegum rakningarvenslum?
- ▶ Ef svo er þá getum við notað kvika bestun.
- ▶ Það er yfirleitt þægilegast að hugsa um rakningarvenslin sem fall, frekar en runu.

- ▶ Getum við sett upp dæmið með þægilegum rakningarvenslum?
- ▶ Ef svo er þá getum við notað kvika bestun.
- ▶ Það er yfirleitt þægilegast að hugsa um rakningarvenslin sem fall, frekar en runu.
- ▶ Látum $f(i, j)$ tákna lengstu sameiginlegu hlutrunu strengjanna $s_1 s_2 \dots s_i$ og $t_1 t_2 \dots t_j$.

- ▶ Getum við sett upp dæmið með þægilegum rakningarvenslum?
- ▶ Ef svo er þá getum við notað kvika bestun.
- ▶ Það er yfirleitt þægilegast að hugsa um rakningarvenslin sem fall, frekar en runu.
- ▶ Látum $f(i, j)$ tákna lengstu sameiginlegu hlutrunu strengjanna $s_1 s_2 \dots s_i$ og $t_1 t_2 \dots t_j$.
- ▶ Okkur mun svo nægja að reikna $f(n, m)$.

- ▶ Viď vitum aď $f(0, i) = f(j, 0) = 0$.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.
- ▶ Almennt gildir að ef við erum að reikna $f(i, j)$ og $s_i = t_j$ þá getum við látið þann staf vera aftastan í sameiginlegu hlutrununni.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.
- ▶ Almennt gildir að ef við erum að reikna $f(i, j)$ og $s_i = t_j$ þá getum við látið þann staf vera aftastan í sameiginlegu hlutrununni.
- ▶ Svo $f(i, j) = f(i - 1, j - 1) + 1$ ef $s_i = t_j$.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.
- ▶ Almennt gildir að ef við erum að reikna $f(i, j)$ og $s_i = t_j$ þá getum við látið þann staf vera aftastan í sameiginlegu hlutrununni.
- ▶ Svo $f(i, j) = f(i - 1, j - 1) + 1$ ef $s_i = t_j$.
- ▶ Ef $s_i \neq t_j$ þá verður annað stakið (eða bæði stökin) að vera ekki í hlutrununni.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.
- ▶ Almennt gildir að ef við erum að reikna $f(i, j)$ og $s_i = t_j$ þá getum við látið þann staf vera aftastan í sameiginlegu hlutrununni.
- ▶ Svo $f(i, j) = f(i - 1, j - 1) + 1$ ef $s_i = t_j$.
- ▶ Ef $s_i \neq t_j$ þá verður annað stakið (eða bæði stökin) að vera ekki í hlutrununni.
- ▶ Við veljum að sjálfsgöðu að sleppa þeim sem gefur okkur betra svar, það er að segja $f(i, j) = \max(f(i - 1, j), f(i, j - 1))$.

- ▶ Við vitum að $f(0, i) = f(j, 0) = 0$.
- ▶ Þetta munu vera grunntilfellin okkar.
- ▶ Almennt gildir að ef við erum að reikna $f(i, j)$ og $s_i = t_j$ þá getum við látið þann staf vera aftastan í sameiginlegu hlutrununni.
- ▶ Svo $f(i, j) = f(i - 1, j - 1) + 1$ ef $s_i = t_j$.
- ▶ Ef $s_i \neq t_j$ þá verður annað stakið (eða bæði stökin) að vera ekki í hlutrununni.
- ▶ Við veljum að sjálfsögðu að sleppa þeim sem gefur okkur betra svar, það er að segja $f(i, j) = \max(f(i - 1, j), f(i, j - 1))$.
- ▶ Við getum svo sett allt saman og fengið

$$f(i, j) = \begin{cases} 0, & \text{ef } i = 0 \text{ eða } j = 0 \\ f(i - 1, j - 1) + 1, & \text{annars, og ef } s_i = t_j \\ \max(f(i - 1, j), f(i, j - 1)), & \text{annars.} \end{cases}$$

```

1 #include <stdio.h>
2 #include <string.h>
3 #define MAXN 10001
4 int max(int a, int b) { if (a < b) return b; return a; }
5
6 char s[10001], t[10001];
7 int d[MAXN][MAXN];
8 int dp_lookup(int x, int y)
9 {
10     if (d[x][y] != -1) return d[x][y];
11     if (x == 0 || y == 0) return 0;
12     if (s[x - 1] == t[y - 1]) return d[x][y] = dp_lookup(x - 1, y - 1) + 1;
13     return d[x][y] = max(dp_lookup(x - 1, y), dp_lookup(x, y - 1));
14 }
15
16 int main()
17 {
18     int n, m, i, j;
19     fgets(s, MAXN, stdin);
20     fgets(t, MAXN, stdin);
21     n = strlen(s) - 1;
22     m = strlen(t) - 1;
23     for (i = 0; i < n + 1; i++) for (j = 0; j < m + 1; j++) d[i][j] = -1;
24     printf("%d\n", dp_lookup(n, m));
25     return 0;
26 }

```


- Það er þessi virði að bera saman `dp_lookup(...)` fallið í forritinu og $f(i, j)$ af glærunni í framan.

$$f(i, j) = \begin{cases} 0, & \text{ef } i = 0 \text{ eða } j = 0 \\ f(i-1, j-1) + 1, & \text{annars, og ef } s_i = t_j \\ \max(f(i-1, j), f(i, j-1)), & \text{annars.} \end{cases}$$

```
8 int dp_lookup(int x, int y)
9 {
10     if (d[x][y] != -1) return d[x][y];
11     if (x == 0 || y == 0) return 0;
12     if (s[x-1] == t[y-1]) return d[x][y] = dp_lookup(x-1, y-1) + 1;
13     return d[x][y] = max(dp_lookup(x-1, y), dp_lookup(x, y-1));
14 }
```

- ▶ Forritið okkar þarf í versta falli að reikna öll möguleg gildi á $f(i, j)$, sem eru $(n + 1) \cdot (m + 1)$ talsins.

- ▶ Forritið okkar þarf í versta falli að reikna öll möguleg gildi á $f(i, j)$, sem eru $(n + 1) \cdot (m + 1)$ talsins.
- ▶ En hvert gildi má reikna í $\mathcal{O}(\quad)$ tíma.

- ▶ Forritið okkar þarf í versta falli að reikna öll möguleg gildi á $f(i, j)$, sem eru $(n + 1) \cdot (m + 1)$ talsins.
- ▶ En hvert gildi má reikna í $\mathcal{O}(1)$ tíma.

- ▶ Forritið okkar þarf í versta falli að reikna öll möguleg gildi á $f(i, j)$, sem eru $(n + 1) \cdot (m + 1)$ talsins.
- ▶ En hvert gildi má reikna í $\mathcal{O}(1)$ tíma.
- ▶ Svo forritið hefur tímaflækjuna $\mathcal{O}(\quad)$.

- ▶ Forritið okkar þarf í versta falli að reikna öll möguleg gildi á $f(i, j)$, sem eru $(n + 1) \cdot (m + 1)$ talsins.
- ▶ En hvert gildi má reikna í $\mathcal{O}(1)$ tíma.
- ▶ Svo forritið hefur tímaflækjuna $\mathcal{O}(n \cdot m)$.