Leit

Leit er annað klassískt vandamál. Gefið fylki af heiltölum `int[] a` er `x` í fylkinu, þ.e. `a[i]==x` fyrir eitthvað `i` ?

Einföld lausn, leitum í fylkinu

```
// Notkun: i = linulegleit(a,x)
// Fyrir: ekkert
// Eftir: i er -1 ef x er ekki í fylkinu a
//        annars er i þ.a. a[i] == x
int linulegleit(int[] a, int x) {
    for (int i = 0; i < a.length; i++) {
        if (a[i] == x) {
            return i;
        }
    }
    return -1;
}
```

Leit

Forritið er rétt en hversu langan tíma tekur að finna x? Ef fylkið er af lengd N

- Ef x er fremst: 1
- Ef x er aftast: N
- Ef x er í a, sett af handahófi: N/2 að meðaltali
- Ef x er ekki í a: N

Hvað ef við erum með 1 milljón heiltölur?

- Er hægt að anna 10 þúsund fyrirspurnum á sek?

Ég er að hugsa um tölu milli 1 og 100.

Þið megið spurja já/nei spurninga

Þið fáið 8 spurningar

Hver er talan

?

Helmingunarleit

Helmingunarleit byggir á sömu hugmynd.

Við gerum ráð fyrir að fylki a sé raðað

- Skoðum stakið í miðjunni á a
- Ef miðjan passar þá fundum við stakið
- Annars vitum við í hvorum helmingnum við eigum að leita

Eins og við leitum í (pappírs) símaskrá

Útfærsla á helmingunarleit

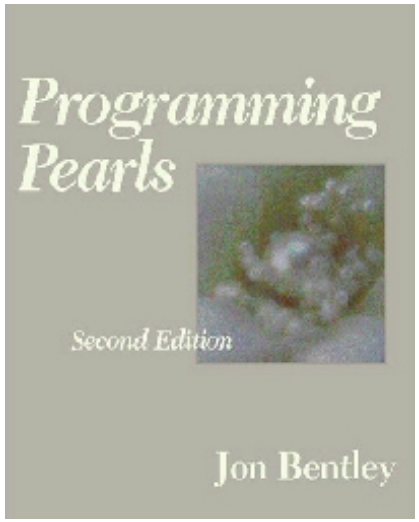
```
// Notkun: i = helmingunarleit(a,x)
// Fyrir: a er raðað í vaxandi röð
// Eftir: i er -1 ef x er ekki í fylkinu a
//        annars er i þ.a. a[i] == x
int helmingunarleit(int[] a, int x) {
    int low = 0, high = a.length-1;
    while ( … ) { // á meðan við erum enn að leita
        … // kíkjum á miðgildin og helmingana
    }
    … // skilum gildi
}
```

Helmingunarleit

Helmingunarleit var fyrst lýst 1946 (óformlega), það liðu 20 ár þar til rétt forrit kom fram.

Jon Bentley í Programming Pearls

- Lagði fyrir helmingunarleit í námskeiði fyrir 50 forritara
- 90% gátu ekki skrifað villulaust forrit eftir marga klukkutíma



Helmingunarleit

```
// Notkun: i = helmingunarleit(a,x)
// Fyrir: a er raðað í vaxandi röð
// Eftir: i er -1 ef x er ekki í fylkinu a
//        annars er i þ.a. a[i] == x
int helmingunarleit(int[] a, int x) {
    int low = 0, high = a.length-1;
    while (low <= high) {
        // Fastayrðing: 0<= low, high <= a.length-1
        //   x getur aðeins verið í a[low] .. a[high]
        int mid = ???

        if (a[mid] < x) {
            low = ???
        } else if (a[mid] > x) {
            high = ???
        } else {
            return mid;
        }
    }
    return -1;
}
```

Helmingunarleit

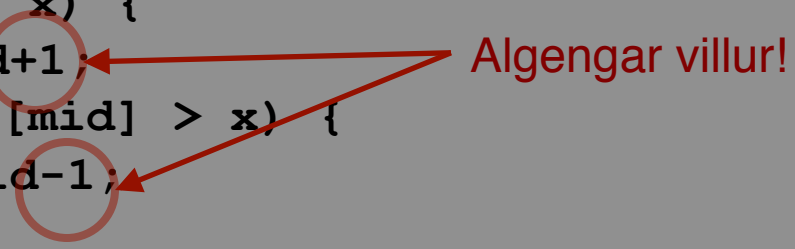
```
// Notkun: i = helmingunarleit(a,x)
// Fyrir: a er raðað í vaxandi röð
// Eftir: i er -1 ef x er ekki í fylkinu a
//        annars er i þ.a. a[i] == x
int helmingunarleit(int[] a, int x) {
    int low = 0, high = a.length-1;
    while (low <= high) {
        // Fastayrðing: 0<= low, high <= a.length-1
        //   x getur aðeins verið í a[low] .. a[high]
        int mid = (low+high)/2;

        if (a[mid] < x) {
            low = mid+1;
        } else if (a[mid] > x) {
            high = mid-1;
        } else {
            return mid;
        }
    }
    return -1;
}
```

Helmingunarleit

```
// Notkun: i = helmingunarleit(a,x)
// Fyrir: a er raðað í vaxandi röð
// Eftir: i er -1 ef x er ekki í fylkinu a
//        annars er i þ.a. a[i] == x
int helmingunarleit(int[] a, int x) {
    int low = 0, high = a.length-1;
    while (low <= high) {
        // Fastayrðing: 0<= low, high <= a.length-1
        //   x getur aðeins verið í a[low] .. a[high]
        int mid = (low+high)/2;

        if (a[mid] < x) {
            low = mid+1;
        } else if (a[mid] > x) {
            high = mid-1;
        } else {
            return mid;
        }
    }
    return -1;
}
```

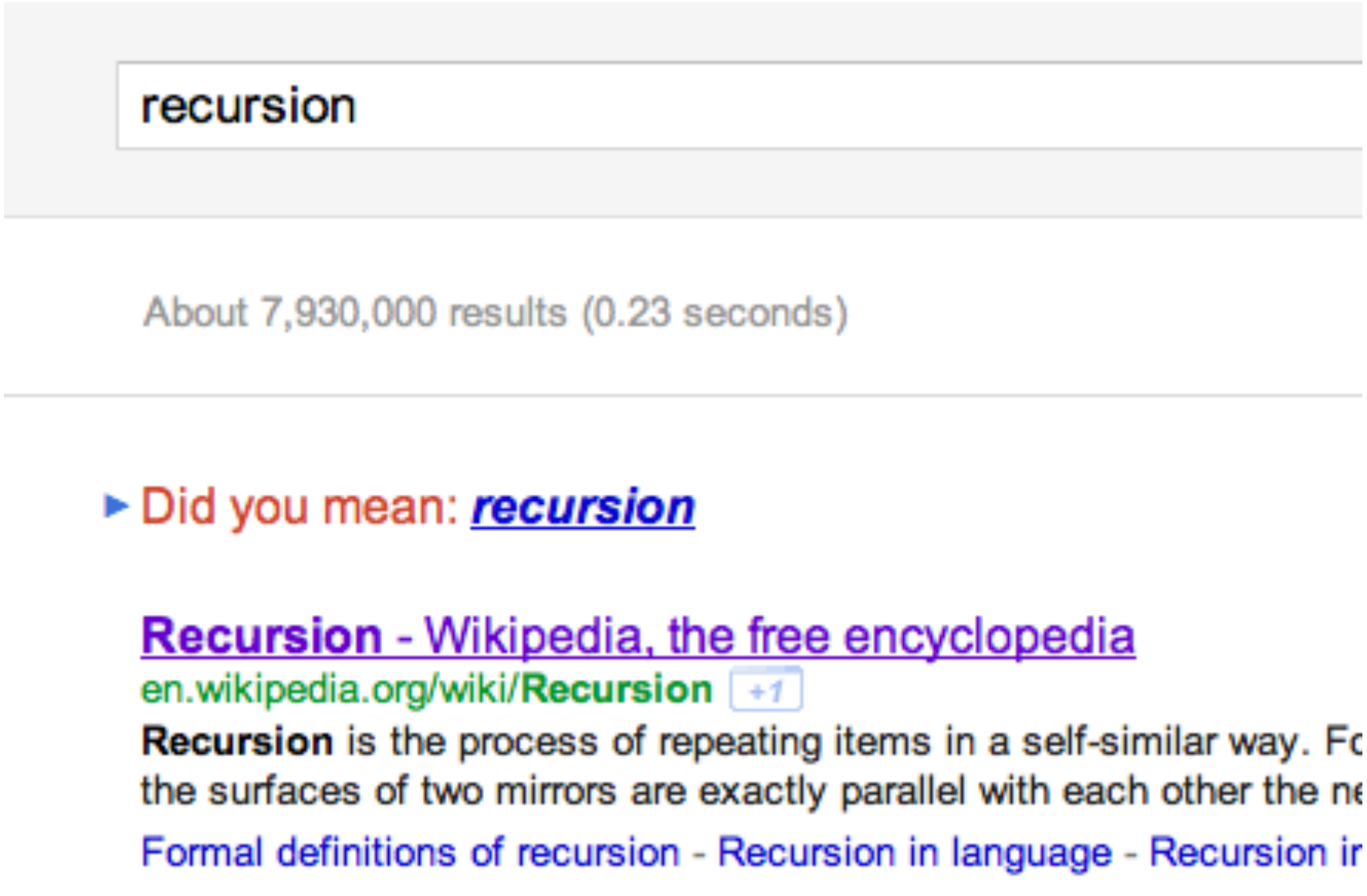


Helmingunarleit er mjög hraðvirk

Dæmi: fylki með 1 milljón ~ 2^{20} stök

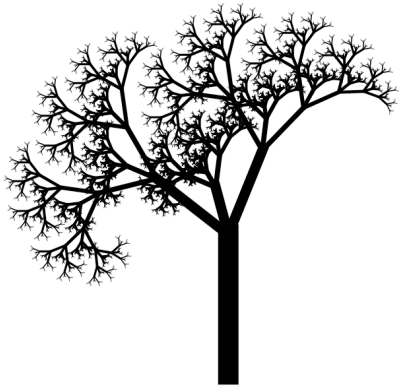
- Línuleg leit: 1M aðgerðir
- Helmingunarleit: 20 aðgerðir!

Almennt fyrir N staka fylki, tekur helmingunarleit $\log_2(N)$ aðgerðir


recursion
About 7,930,000 results (0.23 seconds)
Did you mean: recursion
Recursion - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Recursion +1
Recursion is the process of repeating items in a self-similar way. Fo
the surfaces of two mirrors are exactly parallel with each other the ne
Formal definitions of recursion - Recursion in language - Recursion in

Endurkvæmni

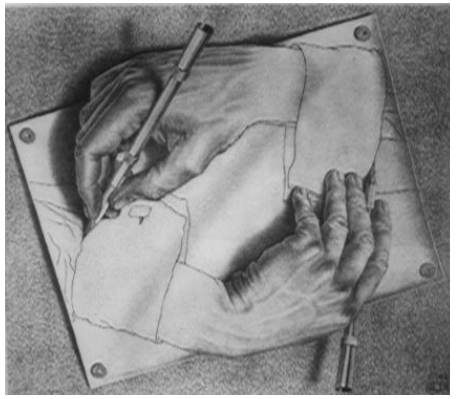
Endurkvæmt fall er fall sem kallar á sjálft sig, beint eða óbeint



Sum verkefni eru einfaldari þegar við notum endurkvæmni

Mörg verkefni eru sjálfvísandi

- GCD, FFT, DFS
- Helmingunarleit, Quicksort, Mergesort
- Tré og aðrar gagnagrindur
- Möppur og skrár í skráarkerfi



Reproductive Parts
M. C. Escher, 1948

GCD

GCD: stærsti sameiginlegi deilir

Dæmi: gcd(4032, 1272) = 24

$$4032 = 2^6 \times 3^2 \times 7^1$$
$$1272 = 2^3 \times 3^1 \times 53^1$$
$$\gcd = 2^3 \times 3^1 = 24$$

Notað til

- Fullstytta brot: 1272/4032 = 53/168
- RSA dulkóðun – risastórar tölur

GCD: stærsti sameiginlegi deilir

Reiknirit Evklíðs (300 f.K)

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q, p \ \% \ q) & \text{otherwise} \end{cases}$$

Grunntilvik

Endurkvæmur reikningur

```
gcd(4032, 1272)  = gcd(1272, 216)
                 = gcd(216, 192)
                 = gcd(192, 24)
                 = gcd(24, 0)
                 = 24.
```

GCD

GCD: finnum stærsta d sem gengur upp í p og q

$$\gcd(p, q) = \begin{cases} p & \text{if } q = 0 \\ \gcd(q, p \ \% \ q) & \text{otherwise} \end{cases}$$

Java útfærsla

```
public static int gcd(int p, int q) {
   if (q == 0) return p;
   else return gcd(q, p % q);
}
```

Grunntilvik

Endurkvæmur reikningur

Hrópmerkt: n! = n * (n-1)! og 0! = 1

Endurkvæm skilgreining, hægt að forrita endurkvæmt:

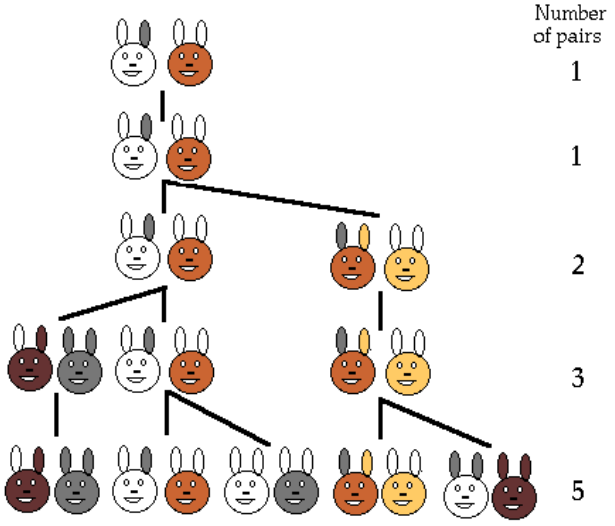
```
public static int factorial(int n) {
   if (n == 0) return 1;
   else return n * factorial(n-1);
}
```

En líka án endurkvæmni:

```
public static int factorial(int n) {
    int r = 1;
    for (int i = 1; i <= n; i++) {
        r *= i;
    }
    return r;
}
```

Fibonacci runan er skilgreind sem

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$



L. P. Fibonacci
(1170 - 1250)

Fibonacci skilgreining

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{otherwise} \end{cases}$$

Einfalt Java forrit

```
public static long F(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return F(n-1) + F(n-2);
}
```

Fibonacci

Er þetta góð leið til að reikna Fibonacci tölur?

```
public static long F(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return F(n-1) + F(n-2);
}
```

Fibonacci

Er þetta góð leið til að reikna Fibonacci tölur?

```
public static long F(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return F(n-1) + F(n-2);
}
```

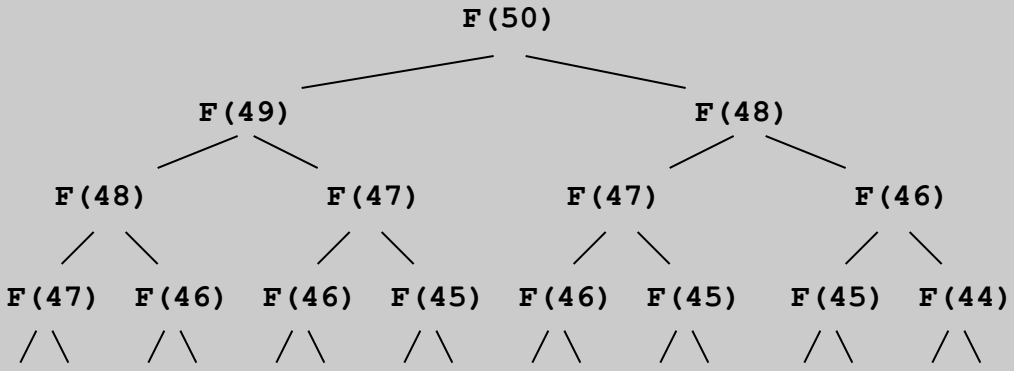
Nei! Þetta er **versta** leiðin

Fibonacci

Er þetta góð leið til að reikna Fibonacci tölur?

```
public static long F(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return F(n-1) + F(n-2);
}
```

Nei, nei og aftur nei! Þetta er **versta** leiðin



Kalltré fyrir F(50)

F(50) reiknað einu sinni.

F(49) reiknað einu sinni.

F(48) reiknað tvisvar.

F(47) reiknað 3 sinnum.

F(46) reiknað 5 sinnum.

F(45) reiknað 8 sinnum.

...

F(1) reiknað 12,586,269,025 sinnum.

Fibonacci

Er þetta skárri leið?

```
public static long F(int n) {
    long[] F = new long[n+1];
    F[0] = 0;
    F[1] = 1;
    for(int i = 2; i <= n; i++)
        F[i] = F[i-1] + F[i-2];
    return F[n];
}
```

Fibonacci

Er þetta skárri leið?

```
public static long F(int n) {
    long[] F = new long[n+1];
    F[0] = 0;
    F[1] = 1;
    for(int i = 2; i <= n; i++)
        F[i] = F[i-1] + F[i-2];
    return F[n];
}
```

Já, sjáum ennþá votta fyrir endurkvæmu skilgreiningunni.

Endurkvæmni er ekki alltaf lausnin

Endurkvæm forritun

Sum forrit verða mun einfaldari ef við notum endurkvæmni. Þurfum að passa okkur á villum

- Ekkert grunntilvik

```
public static int f(int n) {
    return n * f(n-1);
}
```

- Sömu viðföng
  óendanleg lykkja

```
public static int f(int n) {
    if (n == 0) return 1;
    else return n * f(n);
}
```

- Of mikið minni
  Hvert kall á g notar minni á stafla

```
public static int g(int n) {
    if (n == 0) return 1;
    else return n + g(n-1);
}
```

- Of miklir óþarfir/endurteknir útreikningar

Endurkvæm helmingunarleit

```
int helmingunarleit(int[] a, int x, int l, int h) {
    if (l > h) return -1;
    int mid = (l+h)/2;
    if (a[mid] < x) {
        return helmingunarleit(a,x,mid+1,h);
    } else if (a[mid] > x) {
        return helmingunarleit(a,x,l,mid-1);
    } else {
        return mid;
    }
}
```

Hjálparfall

```
int helmingunarleit(int[] a, int x) {
    return helmingunarleit(a,x,0,a.length-1);
}
```

Helmingunarleit

Með fastayrðingu:

```
// Notkun: k = helmingunarleit(a,x,i,j)
// Fyrir:  a[i],..,a[j] er í vaxandi röð
// Eftir:  skilar -1 er x er ekki í a[i],..,a[j]
//         annars gildir a[k] == x
int helmingunarleit(int[] a, int x, int i, int j) {
   if (i > j) return -1;
   int mid = (i+j)/2;
   if (a[mid] < x) {
       return helmingunarleit(a,x,mid+1,j);
   } else if (a[mid] > x) {
       return helmingunarleit(a,x,i,mid-1);
   } else {
       return mid;
   }
}
```

Hvað ef x kemur oft fyrir í a?

Helmingunarleit

Viljum finna minnsta k þannig að x <= a[k]

```
// Notkun: k = helmingunarleit(a,x,i,j)
// Fyrir:  a[i],..,a[j] er í vaxandi röð
// Eftir:  a[i],..,a[k-1] < x <= a[k],..,a[j]
int helmingunarleit(int[] a, int x, int i, int j) {
   if (i > j) return i;
   int mid = (i+j)/2;
   if (a[mid] < x) {
       return helmingunarleit(a,x,mid+1,j);
   } else {
       return helmingunarleit(a,x,i,mid-1);
   }
}
```