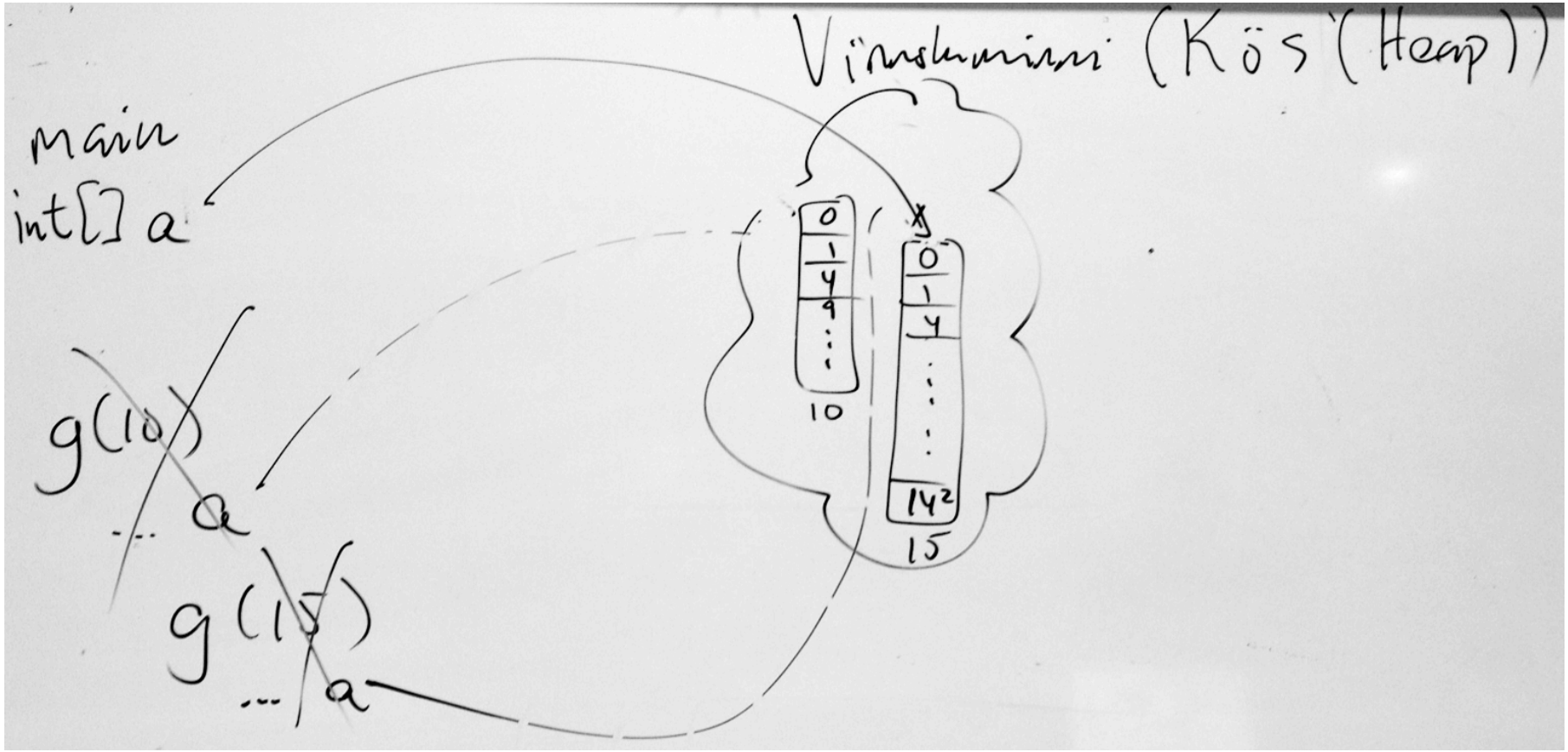
Föll og fylki

Fall getur skilað nýju fylki sem skilagildi

```
public static int[] g(int N) {
    int[] a = new int[N];
    for (int i = 0; i < a.length; i++) {
        a[i] = i*i;
    }
    return a;
}
```

Í hvert skipti sem við köllum á g er tekið frá minni fyrir nýtt fylki.

```
int[] a = g(10); // ATH ekkert new hér
…
a = g(15); // glænýtt fylki!
```


Viinaheinäsirkka (Kös (Heap))
main
int[] a
0
1
4
9
...
10
0
1
4
...
14²
15
g(10)
... a
g(15)
... a

Minni í Java er skipt í tvennt

- Stafli (Stack) er notaður fyrir þær breytur sem eru í gildissviði þess falls sem verið er að keyra (og þeirra falla sem kölluðu á það o.s.frv.)
  - int, double, boolean eru geymdar á stafla
  - Breytur á stafla detta út þegar við hættum í fallinu
- Kös (Heap) er notuð fyrir breytur sem lifa lengur
  - Fylki eru geymd í kös
  - Strengir eru geymdir í kös
  - Í hvert skipti sem við notum `new` þá geymum við eitthvað á kös
- Java notar ruslasöfnun (garbage collection) til að hreinsa þá hluti sem við erum hætt að nota

Röðun fylkja

Verkefni: Skrifum fall sem raðar fylki af heiltölum í rétta röð.

Hvað er rétt röð? vaxandi

```
// Notkun: f(a)
// Fyrir:  ekkert, a er af lengd N
// Eftir:  a er breytt, a[i] <= a[i+1]
//         fyrir 0<= i < N-2
public static void f(int[] a) {
    …
}
```

Hvernig röðum við í rétta röð?

Selection sort:

- Minnsta stakið á að vera fyrst
- Finnum hvar í fylkinu það er
- Skiptum á fyrsta stakinu og minnsta stakinu
- Endurtökum eftir þörfum

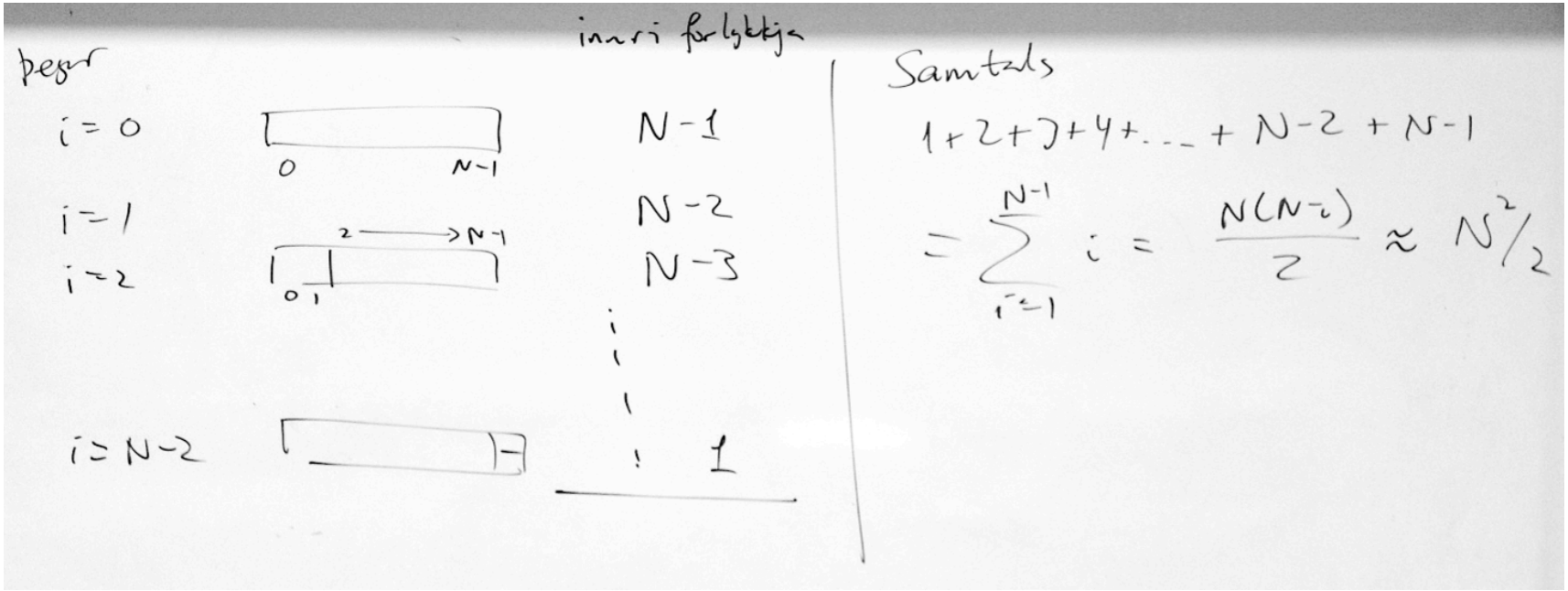
```
// Notkun: f(a)
// Fyrir:  ekkert, a er af lengd N
// Eftir:  a er breytt, a[i] <= a[i+1]
//         fyrir 0<= i < N-2
public static void f(int[] a) {
    for (int i = 0; i < a.length; i++) {
        // a[0] .. a[i-1] inniheldur i minnstu stök í vaxandi röð
        int k = i;
        for (int j = i+1; j < a.length; j++) {
            // a[k] er minnst af stökunum í a[i] .. a[j-1]
            if (a[j] < a[k]) {
                k = j;
            }
        }
        // a[k] er minnsta stakið í a[i] .. a[N-1]
        // skiptum á i og k í fylkinu a
        int t = a[i];
        a[i] = a[k];
        a[k] = t;
    }
    // a[0] .. a[N-1] er raðað í vaxandi röð
}
```

Kostir við selection sort

- Það raðar í rétta röð
- ?

Keyrslutími

- Óháður gildunum, tekur alltaf sama tíma
- $\approx N^2/2$, fyrir fylki af lengd N
- Hægt að gera mun betur



Heildarkeyrslutími vex eins og N^2

Röðun

Af hverju röðun?

- Klassískt vandamál
- Mörg verkefni eru að hluta til leyst með röðun
- Röðum nemendum eftir einkunnum, o.s.frv
- Auðveldara að vinna með raðað gögn en óraðað

Hröð röðunarreiknirit eru mikilvæg

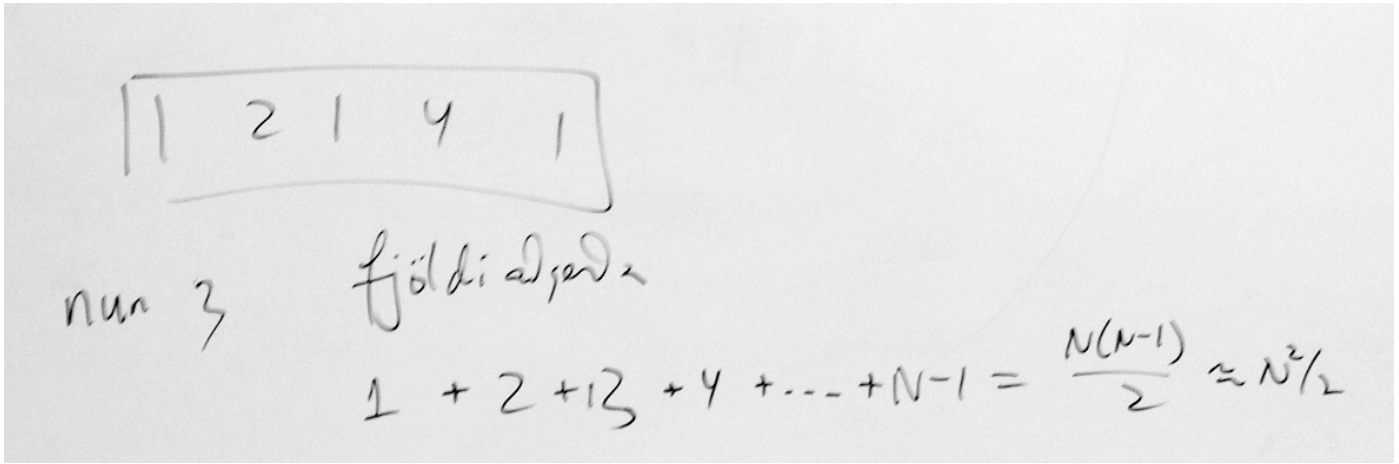
- Google raðar vefsíðum eftir PageRank röð
- Hraðvirk textaleit byggir á röðun
  - Lífupplýsingafræði: hvar kemur strengurinn
    GTGCCATATGGCATTAGGAAATAAGCTTGAGTT
    fyrir í erfðamengi mannsins?
    (chr4:74,262,231-74,262,263)

Röðun

Gefið fylki af heiltölum `int[] a`, hversu margar ólíkar tölur eru í a?

1. einföld lausn:

```
int num = 0;
for (int i = 0; i < a.length; i++) {
    boolean found = false;
    for (int j = 0; j < i; j++) {
        if (a[j] == a[i]) {
            found = true;
        }
    }
    if (!found) {
        num++;
    }
}
```



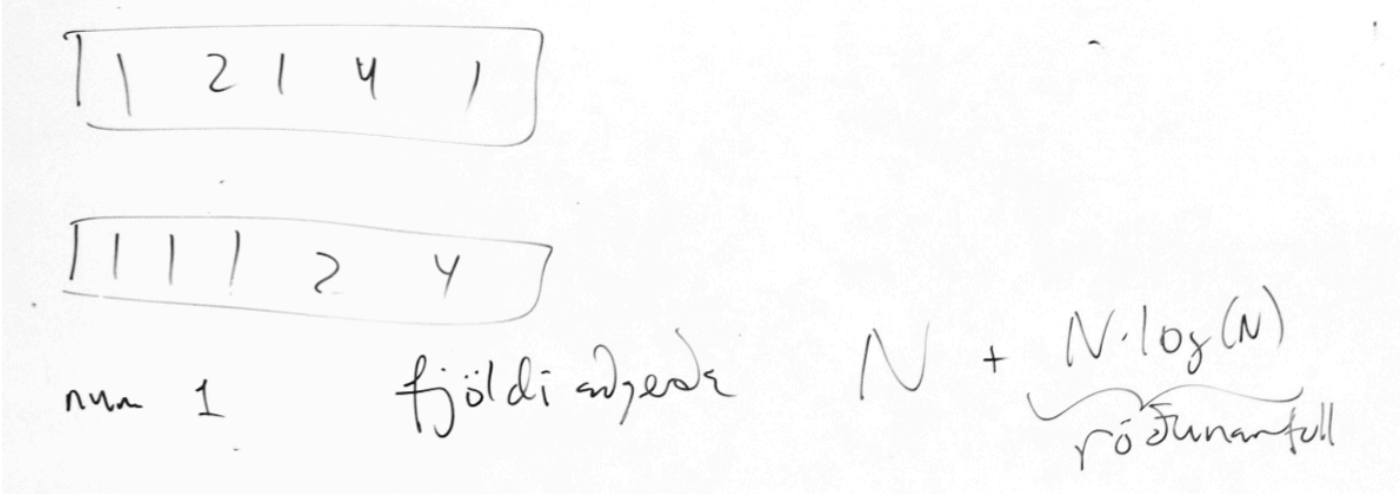
Eins og í selection sort, tíminn er N^2

Röðun

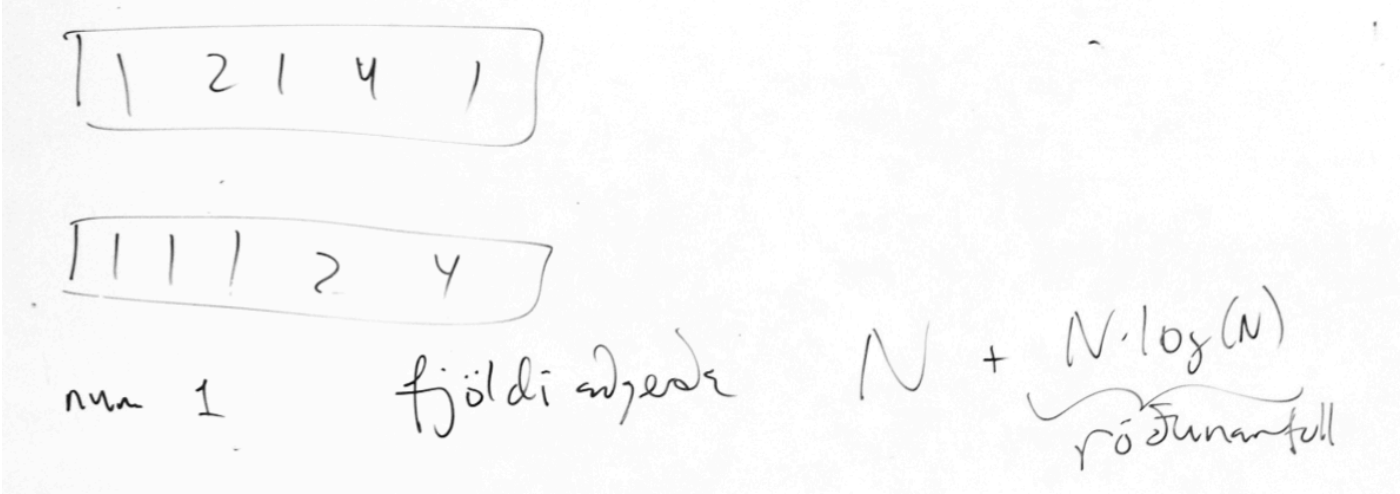
Gefið fylki af heiltölum `int[] a`, hversu margar ólíkar tölur eru í a?

2. betri lausn

```
rada(a); // köllum á eitthvað röðunarfall
int num = 1;
for (int i = 1; i < a.length; i++) {
    if (a[i-1] != a[i]) {
        num++;
    }
}
```



Keyrslutíminn verður háður röðunarfallinu, besta mögulega lausn tekur $N*\log_2(n)$ tíma



Keyrslutíminn verður háður röðunarfallinu, besta mögulega lausn tekur $N*\log_2(n)$ tíma