Mergesort

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

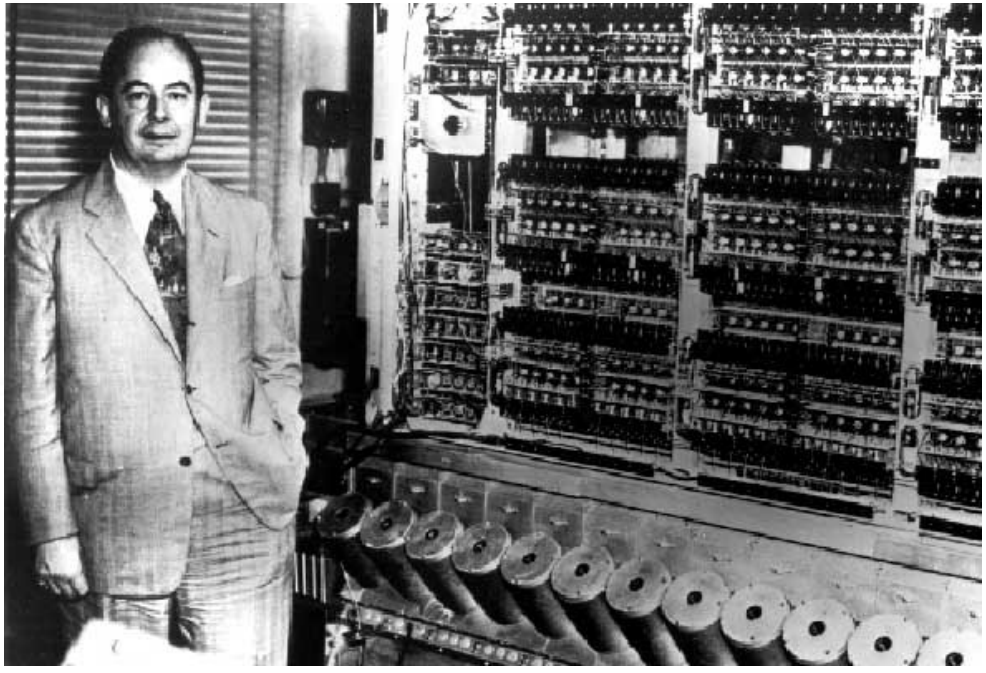
Háskóli Íslands

2012-02-22

EDVAC

First Draft of a Report on the EDVAC

John von Neumann



John von Neumann

- Manhattan verkefnið
- EDVAC: Stored program
- von Neumann architecture
- **Mergesort** árið 1945
- Monte Carlo hermun
- Minimax í leikjafræði
- Slembitölugjafar
- Reiknifræði
- Cellular automata
- Tölvulíkon fyrir veðurspá
- en aðallega þekktur fyrir vinnu í eðlisfræði og stærðfræði

Mergesort

Hugmyndin:

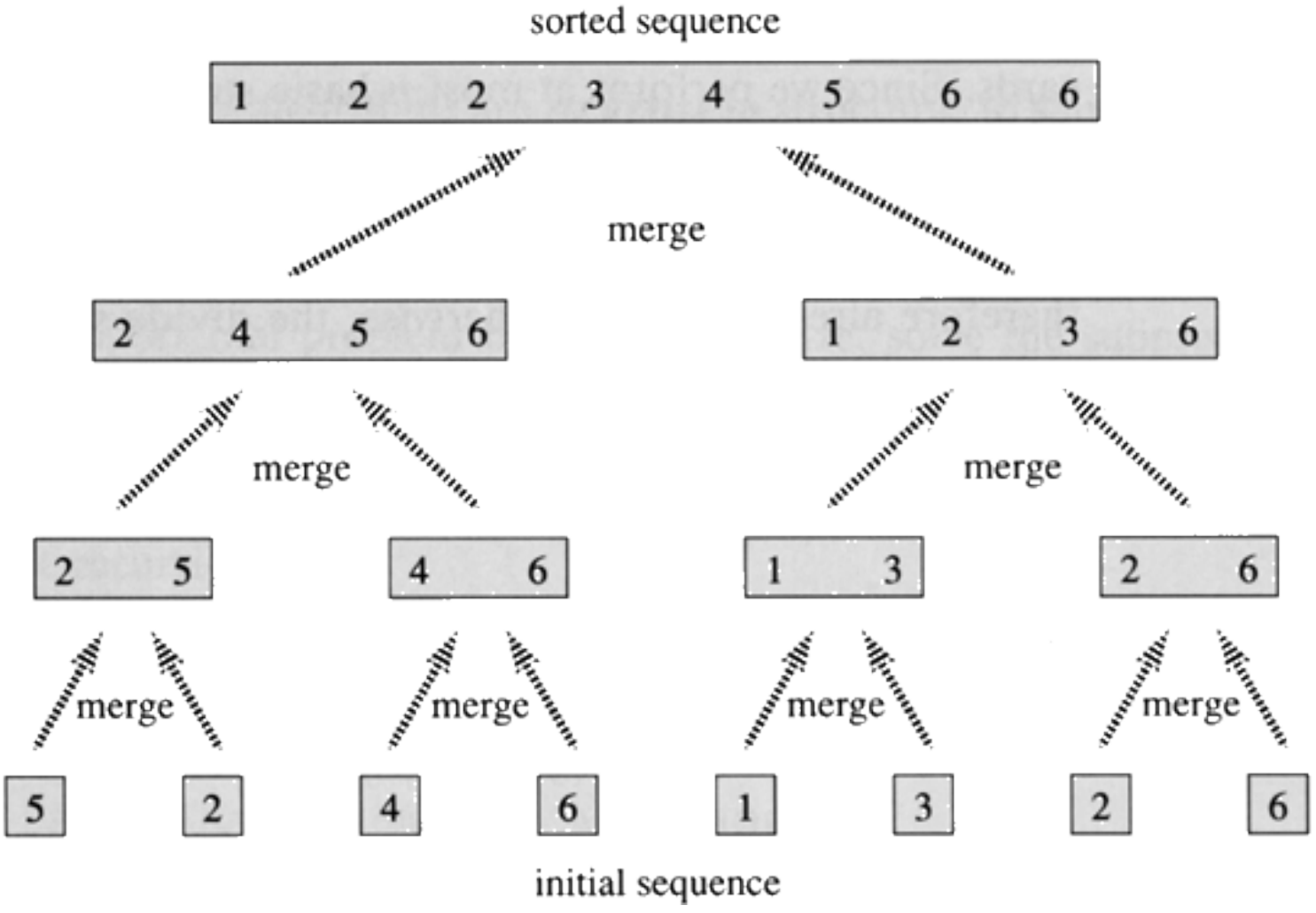
- split: Skiptum fylkinu upp í tvo hluta
- sort: Endurkvæmt röðum sitthvorum hlutanum
- merge: Sameinum hlutana upp á nýtt

Mergesort overview

Merging demo

http://algs4.cs.princeton.edu/lectures/22DemoMerge.mov

Merging


sorted sequence
1 2 2 3 4 5 6 6
merge
2 4 5 6
1 2 3 6
merge
merge
2 5
4 6
1 3
2 6
merge
merge
merge
merge
5
2
4
6
1
3
2
6
initial sequence

Merging

Hvernig sameinum við tvö röðuð svæði í eitt raðað svæði?

- Notum samröðun (merge) með aðstoð aukafylkis

		a[]												aux[]									
	k	0	1	2	3	4	5	6	7	8	9	i	j	0	1	2	3	4	5	6	7	8	9
input		E	E	G	M	R	A	C	E	R	T			–	–	–	–	–	–	–	–	–	–
copy		E	E	G	M	R	A	C	E	R	T			E	E	G	M	R	A	C	E	R	T
												0	5										
	0	A										0	6	E	E	G	M	R	A	C	E	R	T
	1	A	C									0	7	E	E	G	M	R		C	E	R	T
	2	A	C	E								1	7	E	E	G	M	R			E	R	T
	3	A	C	E	E							2	7		E	G	M	R			E	R	T
	4	A	C	E	E	E						2	8			G	M	R			E	R	T
	5	A	C	E	E	E	G					3	8			G	M	R				R	T
	6	A	C	E	E	E	G	M				4	8				M	R				R	T
	7	A	C	E	E	E	G	M	R			5	8					R				R	T
	8	A	C	E	E	E	G	M	R	R		5	9									R	T
	9	A	C	E	E	E	G	M	R	R	T	6	10										T
merged result		A	C	E	E	E	G	M	R	R	T												

Abstract in-place merge trace

Merge með aðstoð hjálparfylkis

```
// Notkun: merge(a, aux, lo, mid, hi);
// Fyrir:  a[lo..mid] er í vaxandi röð
//         a[mid+1..hi] er í vaxandi röð
// Eftir:  a[lo..hi] er í vaxandi röð
void merge(String[] a, String[] aux, int lo, int mid, int hi)
{
    for (int k = lo; k <= hi; k++)
        aux[k] = a[k];

    int i = lo, j = mid+1;

    for (int k = lo; k <= hi; k++) {
        if      (i > mid)                  a[k] = aux[j++];
        else if (j > hi)                   a[k] = aux[i++];
        else if (less(aux[j], aux[i])) a[k] = aux[j++];
        else                               a[k] = aux[i++];
    }
}
```

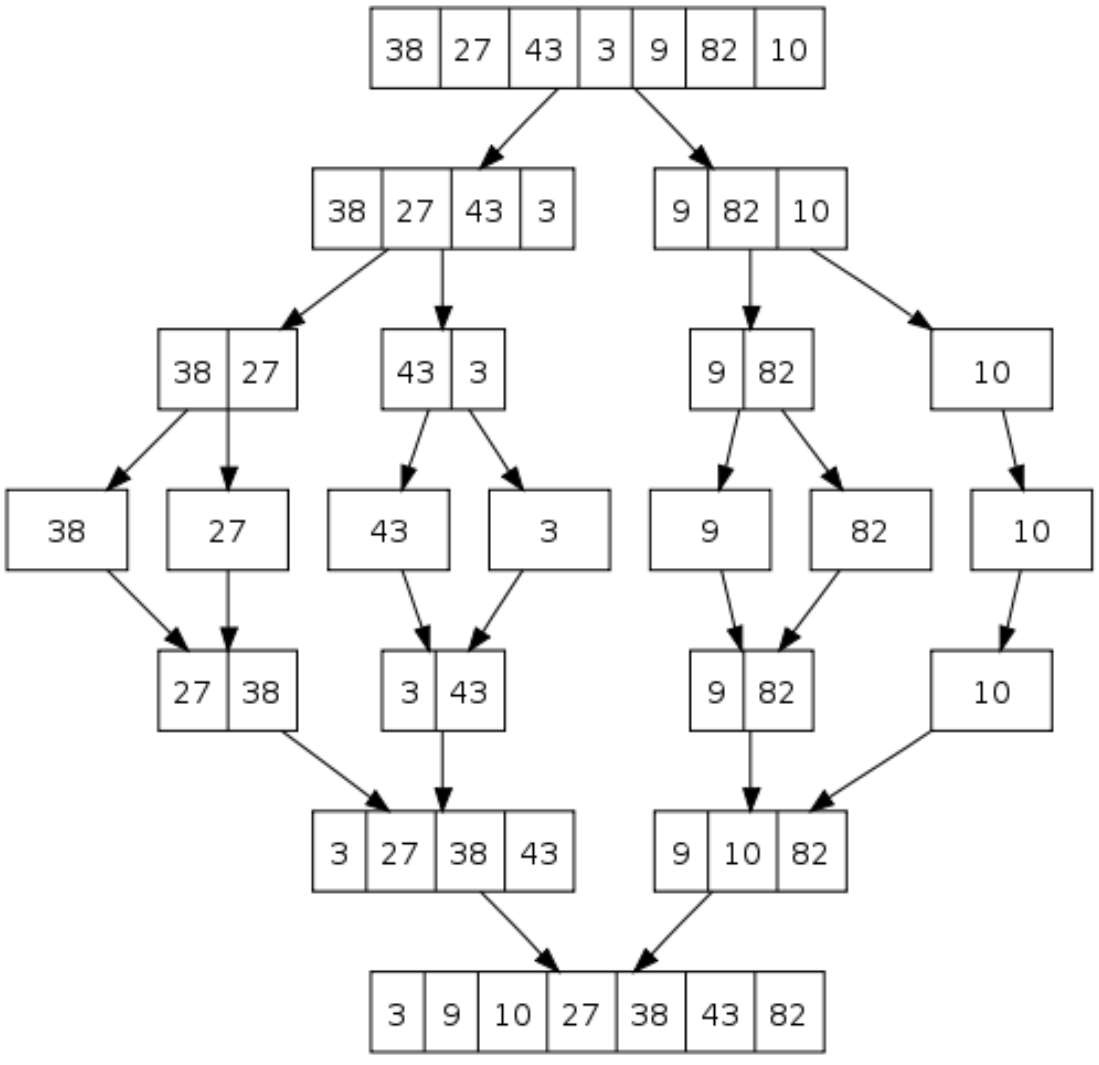
Merge með fastayrðingu

```
    // ...

    for (int k = lo; k <= hi; k++) {
        // a[lo..k] inniheldur tölurnar úr
        // aux[lo..i-1] og aux[mid+1..j-1]
        // í vaxandi röð
        //
        // aux invariant:
        // | done | todo  | done | todo  |
        //  ^      ^      ^      ^      ^
        //  lo     i      mid    j      hi

        if      (i > mid)                  a[k] = aux[j++];
        else if (j > hi)                   a[k] = aux[i++];
        else if (less(aux[j], aux[i])) a[k] = aux[j++];
        else                               a[k] = aux[i++];
    }
}
```

Mergesort


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Mergesort

```
static void sort(String[] a, String[] aux, int lo, int hi)
{
    if (hi <= lo) return;
    int mid = lo + (hi - lo) / 2;
    sort(a, aux, lo, mid);
    sort(a, aux, mid+1, hi);
    merge(a, aux, lo, mid, hi);
}

static void sort(String[] a)
{
    String[] aux = new String[a.length];
    sort(a, aux, 0, a.length-1);
}
```

Mergesort trace

	a[]															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 0, 0, 1)	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 2, 2, 3)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 0, 1, 3)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 4, 4, 5)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 6, 6, 7)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 4, 5, 7)	E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E
merge(a, 0, 3, 7)	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
merge(a, 8, 8, 9)	E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E
merge(a, 10, 10, 11)	E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E
merge(a, 8, 9, 11)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, 12, 12, 13)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, 14, 14, 15)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L
merge(a, 12, 13, 15)	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a, 8, 11, 15)	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge(a, 0, 7, 15)	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

lo

hi

Trace of merge results for top-down mergesort

Mergesort animation

http://www.sorting-algorithms.com/merge-sort

Praktísk atriði í útfærslu

Mergesort er óhagkvæmt fyrir lítil svæði. Oft skipt yfir í insertion sort neðst í endurkvæmninni (cutoff oft í kringum ≈ 7 gildi).

```
static void sort(String[] a, String[] aux, int lo, int hi)
{
    if (hi <= lo + CUTOFF - 1)
    {
        Insertion.sort(a, lo, hi);
        return;
    }
    int mid = lo + (hi - lo) / 2;
    sort(a, aux, lo, mid);
    sort(a, aux, mid+1, hi);
    merge(a, aux, lo, mid, hi);
}
```

Bottom-up mergesort

Hugmynd:

- merge: Samröðum öllum svæðum af stærð 1
- repeat: Endurtökum fyrir öll svæði af stærð 2, 4, 8, 16, ...

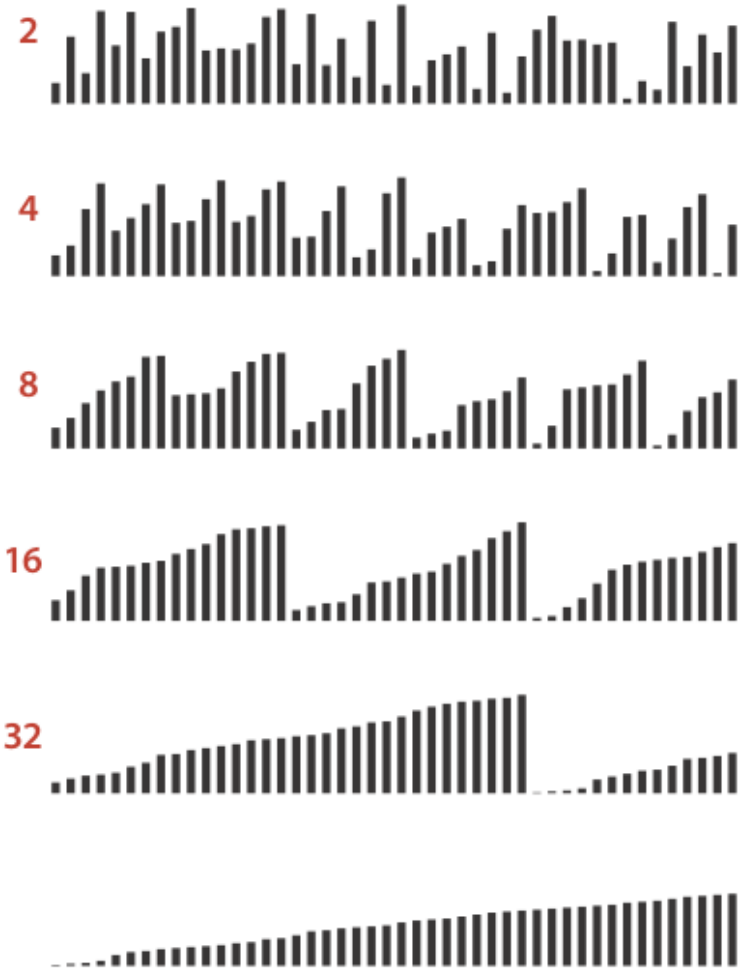
	a[i]															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
sz = 1																
merge(a, 0, 0, 1)	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 2, 2, 3)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 4, 4, 5)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 6, 6, 7)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 8, 8, 9)	E	M	G	R	E	S	O	R	E	T	X	A	M	P	L	E
merge(a, 10, 10, 11)	E	M	G	R	E	S	O	R	E	T	A	X	M	P	L	E
merge(a, 12, 12, 13)	E	M	G	R	E	S	O	R	E	T	A	X	M	P	L	E
merge(a, 14, 14, 15)	E	M	G	R	E	S	O	R	E	T	A	X	M	P	E	L
sz = 2																
merge(a, 0, 1, 3)	E	G	M	R	E	S	O	R	E	T	A	X	M	P	E	L
merge(a, 4, 5, 7)	E	G	M	R	E	O	R	S	E	T	A	X	M	P	E	L
merge(a, 8, 9, 11)	E	G	M	R	E	O	R	S	A	E	T	X	M	P	E	L
merge(a, 12, 13, 15)	E	G	M	R	E	O	R	S	A	E	T	X	E	L	M	P
sz = 4																
merge(a, 0, 3, 7)	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a, 8, 11, 15)	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
sz = 8																
merge(a, 0, 7, 15)	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Bottom-up mergesort kóði

```
static void sort(String[] a)
{
  int N = a.length;
  String[] aux = new String[N];

  for (int sz = 1; sz < N; sz = sz+sz)
      for (int lo = 0; lo < N-sz; lo += sz+sz)
          merge(a, aux, lo, lo+sz-1,
                Math.min(lo+sz+sz-1, N-1));
}
```

Bottom-up mergesort trace



Mergesort fyrir biðraðir

```
// Notkun: sort(q);
// Eftir:  Búið er að raða gildunum í q í vaxandi röð
static public void sort(Queue<String> q)
{
    if (q.count() < 2)
        return;

    Queue<String> q1 = new Queue<String>();
    Queue<String> q2 = new Queue<String>();

    split(q,q1,q2);

    sort(q1);
    sort(q2);

    merge(q1,q2,q);
}
```

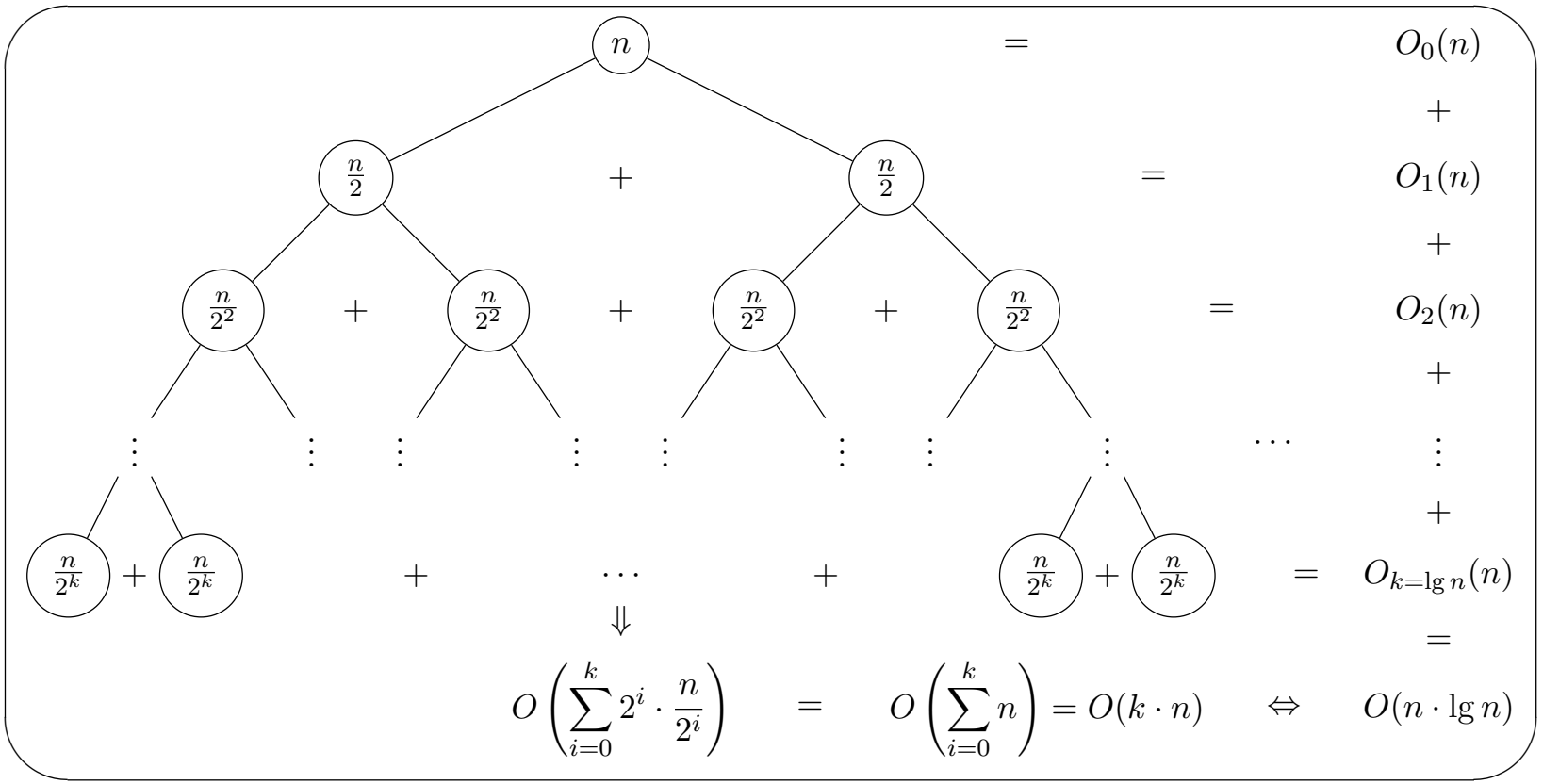
split fyrir biðraðir

```
// Notkun: split(p,q1,q2);
// Fyrir:  q1 og q2 eru tómar biðraðir, sem hver um sig
//         hefur pláss fyrir helming gilda úr p (þ.e.
//         fyrir (p.count()+1)/2 gildi).
// Eftir:  Búið er að fjarlægja öll gildi úr p og setja
//         helming þeirra í q1 og hinn helminginn í q2
//         (það gæti munað einum í fjölda staka, ef
//         fjöldinn er oddatala).
static public void split(Queue<String> p,
                         Queue<String> q1,
                         Queue<String> q2)
{
    // ...
}
```

merge fyrir biðraðir

```
// Notkun: merge(q1,q2,q);
// Fyrir:  q er tóm biðröð, q1 og q2 eru biðraðir, sem
//         innihalda gildi, sem eru í vaxandi röð frá
//         fremsta til aftasta.
//         q hefur pláss fyrir öll gildin í q1 og q2.
// Eftir:  q1 og q2 eru tómar, gildin úr þeim eru í q í
//         vaxandi röð frá fremsta til aftasta.
static public void merge(Queue<String> q1,
                         Queue<String> q2,
                         Queue<String> q)
{
    // ...
}
```

Myndræn greining á tímaflækju



Greining á mergesort

Tímaflækja:

- Fræðilegt lágmark fyrir samanburðarröðun er $\Omega(n \log n)$
- Mergesort notar $O(n \log n)$ samanburði – optimal

Minnisnotkun:

- Mergesort notar $O(n)$ aukaminni – ekki space optimal
- Röðunarreiknirit er *in-place* ef það notar $\leqslant \log N$ aukaminni.
- Dæmi: insertion sort og selection sort eru in-place

Er til reiknirit fyrir röðun sem er optimal í tímaflækju og minnisnotkun?

Já – quicksort, sem við skoðum í næstu viku

Mælingar

Gróft áætlað:

- Fartölva getur framkvæmt 10^8 samanburði á sekúndu
- Ofurtölva getur framkvæmt 10^{12} samanburði á sekúndu

	insertion sort (N^2)			mergesort (N log N)		
computer	thousand	million	billion	thousand	million	billion
home	instant	2.8 hours	317 years	instant	1 second	18 min
super	instant	1 second	1 week	instant	instant	instant

$\Longrightarrow$ Gott reiknirit getur verið betra en góð tölva

Mergesort vs. quicksort

Mergesort:

- Java röðun fyrir hluti (objects)
- Perl, C++, Python stable sort, Firefox JavaScript, ...

Quicksort:

- Java röðun fyrir primitive tög
- C qsort, Unix, Visual C++, Python, Matlab, Chrome JavaScript, ...

Hvenær hentar mergesort sérstaklega vel?

- Mergesort þarf ekki random access, hentar til að raða listum
- Mergesort hentar vel sem external sort, þegar gögnin komast ekki öll fyrir í minni
- Mergesort hentar vel fyrir verkaskiptingu (parallelization)