2ip65 week 7
Sorting

Kees Huizing

Analysis of algorithms: example

		n=10	n=100	n=1000
InsertionSort:	$15\ n^2 + 7n - 2$	1568	150698	1.5×10^7
MergeSort:	$300\ n \lg n + 50\ n$	10466	204316	3.0×10^6

InsertionSort
6 x faster

InsertionSort
1.35 x faster

MergeSort
5 x faster

The rate of growth of the running time as a function of the input is essential!

n = 1,000,000 InsertionSort 1.5×10^{13}

MergeSort 6×10^9 2500 x faster !

O-notation

Let $g(n) : N \mapsto N$ be a function. Then we have

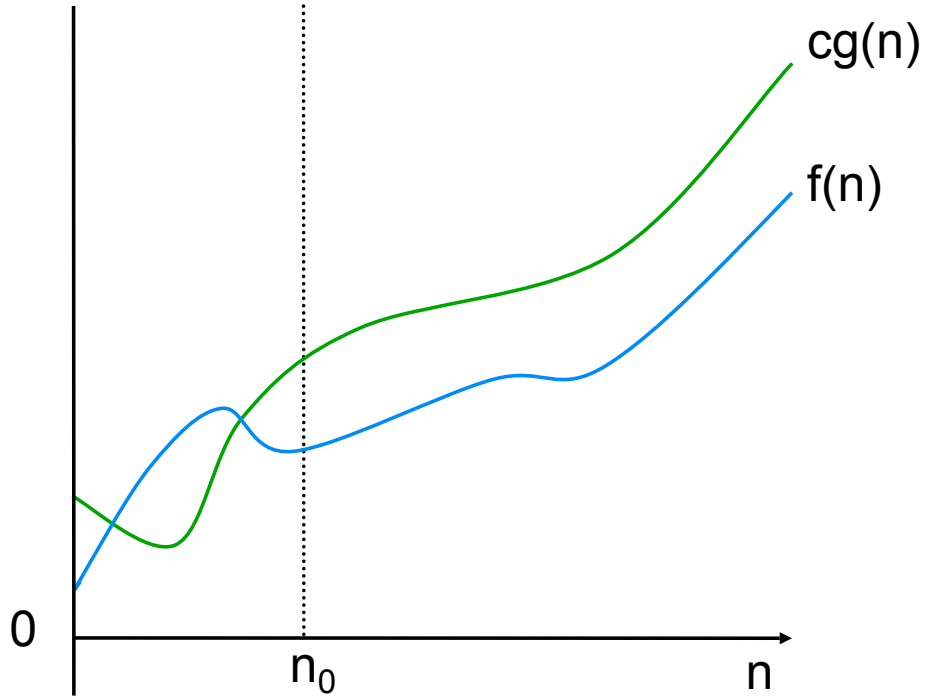
$O(g(n))$ = { $f(n)$: there exist positive constants c and n_0
such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$ }

"$O(g(n))$ is the set of functions that grow
at most as fast as $g(n)$"

O-notation

Let $g(n) : N \mapsto N$ be a function. Then we have

$O(g(n))$ = { $f(n)$: there exist positive constants c and n_0
such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$ }



Notation: $f(n) = O(g(n))$

Insertion sort

```
class InsertionSort extends SortAlgorithm {
    void sort(int a[]) {
        for (int i = 1; i < a.length; i++) {
            int j = i;
            int B = a[i];
            while ((j > 0) && (a[j-1] > B)) {
                if (stopRequested) { return; } //for animation
                a[j] = a[j-1];
                j--;
                pause(i,j); //for animation
            }
            a[j] = B;
            pause(i,j); //for animation
        }
    }
}
```

Quicksort

```
class Sorter {
    int[] a;

    void quicksort(int[] a) {
        this.a = a;
        quicksort(0, a.length);
    }

    void quicksort(int lo, int hi) {
        if (hi - lo < 2) {
            return;
        }

        int pivin = partition(lo, hi);
        quicksort(lo, pivin);
        quicksort(pivin+1, hi);
    }
```

```
    int partition(int lo0, int hi0) {
        int pivot = a[lo0];
        int lo = lo0+1;
        int hi = hi0-1;

        while (lo<=hi) {
            if (a[lo] <= pivot) {
                lo++;
            } else if (pivot<=a[hi]) {
                hi--;
            } else {
                swap(lo, hi);
                lo++;
                hi--;
            }
        }

        swap(lo0, hi);
        return hi;
    }

    void swap(int hier, int daar) {
        int opzij = a[hier];
        a[hier] = a[daar];
        a[daar] = opzij;
    }
}
```

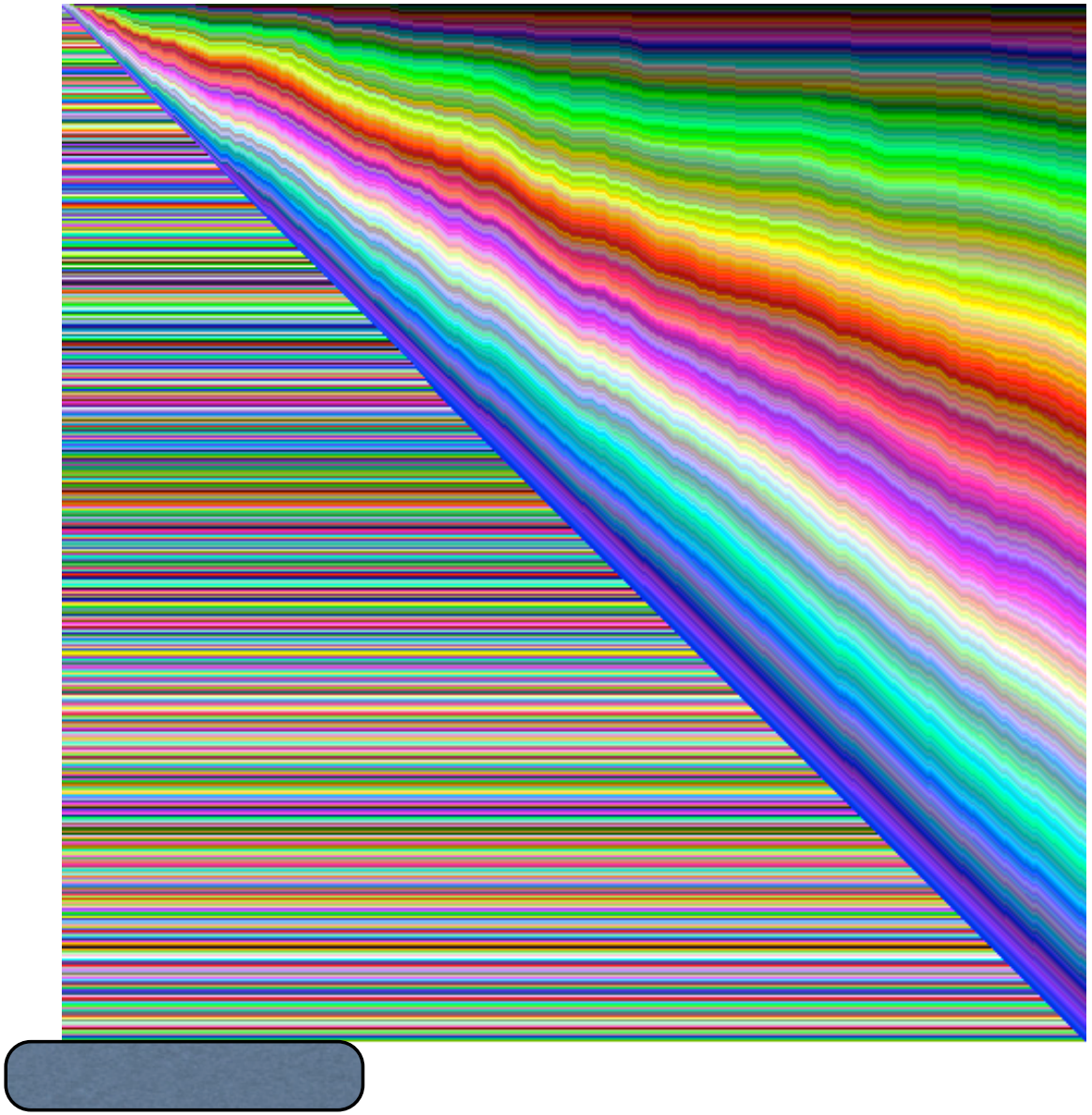
Quicksort Annotated 1

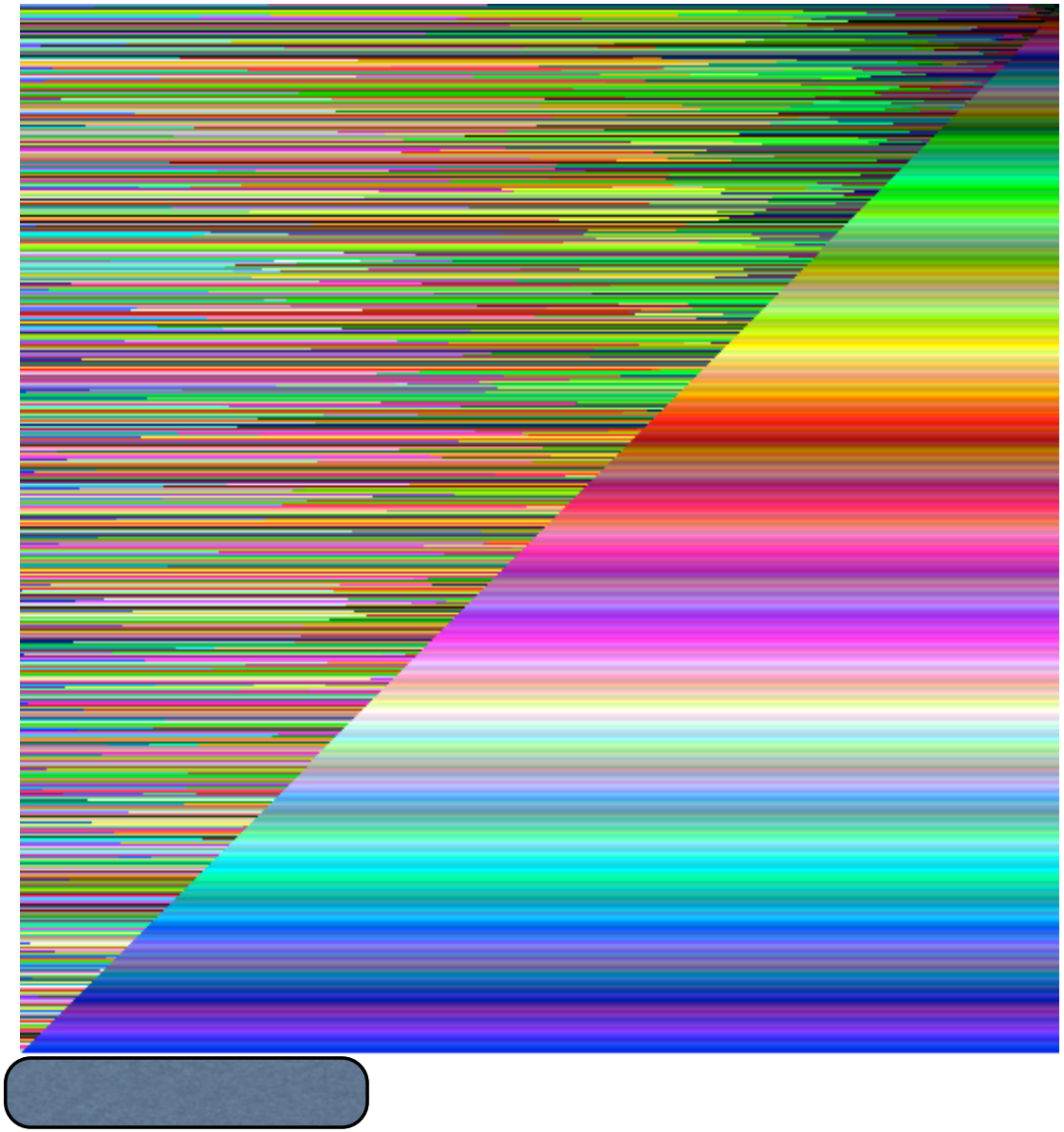
```
class SorterAnnotated {
    int[] a;

    void quicksort(int[] a) {
        this.a = a;
        quicksort(0, a.length);
    }

    //@pre 0 ≤ lo ≤ hi ≤ a.length
    //@post a[lo..hi) ascending
    //@post a[lo..hi) permutation of \old(a[lo..hi))
    void quicksort(int lo, int hi) {
        if (hi-lo < 2) { // nothing to sort
            return;
        }

        int pivin = partition(lo, hi);
        //assert lo+1 ≤ pivin < hi
        //assert a[lo+1..pivin) ≤ pivot ≤ a[pivin+1..hi)
        //assert pivot = a[lo]
        //P: assert a[lo..pivin) ≤ a[pivin] ≤ a[pivin+1..hi)

        quicksort(lo, pivin);
        //assert a[lo..pivin) ascending && P
        quicksort(pivin+1, hi);
        //assert a[pivin+1, hi) ascending
        //assert a[lo..pivin) ascending && P
    }
```

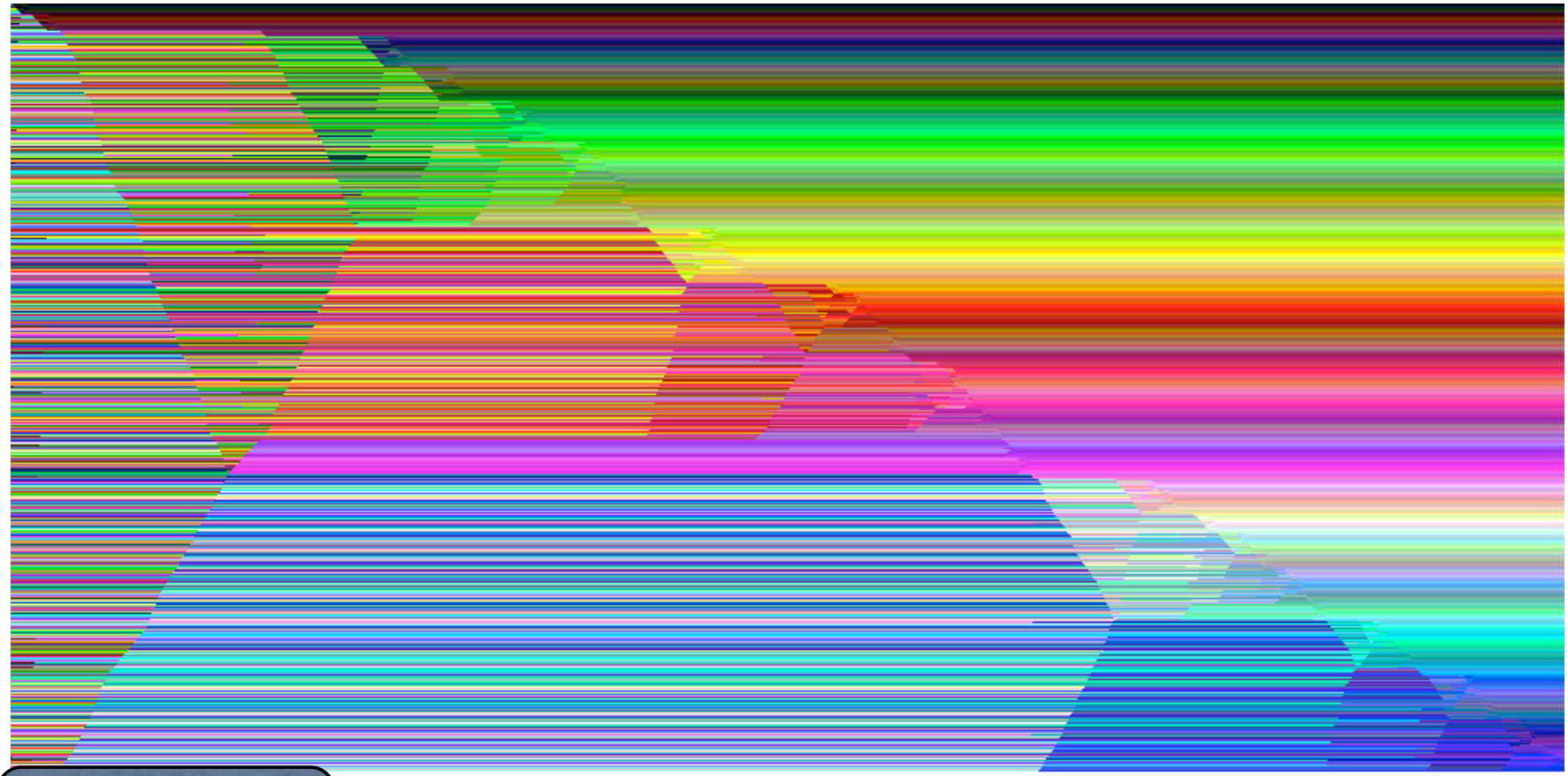
Quicksort Annotated 2

```
//@pre 0≤lo0<hi0≤a.length
//@post a[lo0..\result) ≤ a[\result] ≤ a[\result+1..hi0)
int partition(int lo0, int hi0) {
    int pivot = a[lo0];
    int lo = lo0+1;
    int hi = hi0-1;


    //@inv a[lo0..lo) ≤ pivot ≤ a[hi+1..hi0)
    //@inv lo0 < lo && hi < hi0 && lo ≤ hi+1
    while (lo<=hi) {
        if (a[lo] <= pivot) {
            lo++;
        } else if (pivot<=a[hi]) {
            hi--;
        } else { assert a[lo] > pivot && pivot > a[hi];
            swap(lo, hi);
            assert a[lo] < pivot && pivot < a[hi];
            lo++;
            hi--;
        }
    }
    //assert a[lo0..lo) ≤ pivot ≤ a[hi+1..hi0)
    assert lo == hi+1 && lo0 <= hi && hi < hi0 && a[lo0]==pivot;
    swap(lo0, hi);
    //assert a[lo0..lo) ≤ a[hi] ≤ a[hi+1..hi0)
    return hi;
}
```

```
//@pre 0≤i,j<a.length
//@post a[i] == \old(a[j] && a[j] == \old(a[i]
void swap(int i, int j) {
    int help = a[i];
    a[i] = a[j];
    a[j] = help;
}
```







From Aldo Cortesi – http://corte.si/posts/code/sortvis-fruitsalad/index.html