CS 2110: Simple sorting

The Preservation step in showing correctness of a loop is showing:

A: If the loop invariant holds at the beginning of the loop body then it holds at the end of the loop body.

Initialization

B: If the precondition holds before the loop, then the loop invariant does too.

C: If the guard is true and the loop invariant holds, then the loop invariant holds after running the loop body.

Postcondition

D: If the guard is false and the loop invariant holds, then the loop post condition holds.

Termination

E: There exists some function DF that decreases on each loop iteration and cannot decrease forever without violating the loop invariant or the loop guard.

Sorting

permute

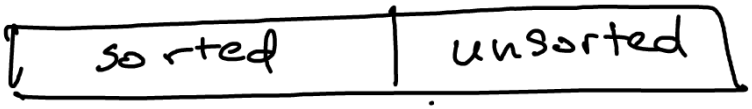
/** Effect: rearrange elements of a into ascending sorted order.

```
void sort(int[] a);
```

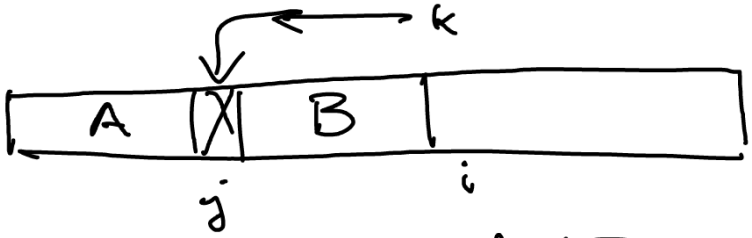
Best algorithm for small arrays: insertion sort

Idea: find location for each element

loop invariant:



inner loop invariant:



A, B sorted, $A \leq B$

$k < B$ $\quad 1 \leq i \leq a.length$

A, B, K are a permutation of elements

Running time

Best case: a already sorted

total time: $O(n)$

inner loop: $O(1)$

Worst case: sorted in descending order

total work: $1+2+3+4+\cdots+n$

$= \frac{n^2}{2} = O(n^2)$

A: $O(\log n)$

B: $O(n)$

C: $O(n^2)$

D: $O(2^n)$

E: ?

10^5 — 803 ms

$\frac{10^6}{10x}$ — $\frac{80{,}000 \text{ ms}}{100x}$

Stability

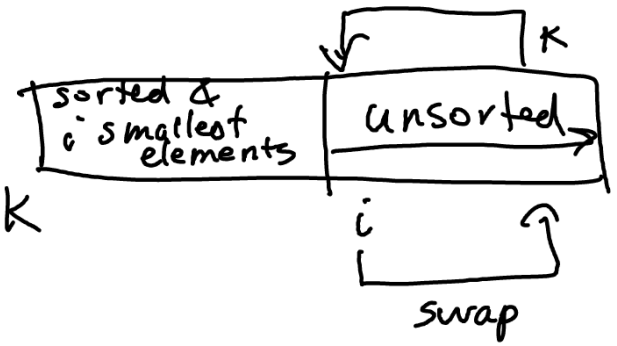
Sort is stable if it keeps "equal" elements in same order.

$5, 3, 4, \bar{3}, \bar{2}, 1 \rightsquigarrow 1\bar{2}3\bar{3}45$

Selection Sort

Find right element for location.

```
for (int i = 0; i < n; i++) {
    // inner loop: find smallest element k
    // in a[i..n-1]
    // swap a[i] with k
}
```



Running time? Best case = Worst case = $1+2+3+4+5\cdots$
$= O(n^2)$

minimum # moves $O(n)$ swaps

Stable? No, because of swapping.

How fast can a generic sort be?

only use $<, =, >$

```
<T> sort(T[] a, Comparator<T> c);
```

Sort receives a permutation of sorted array.

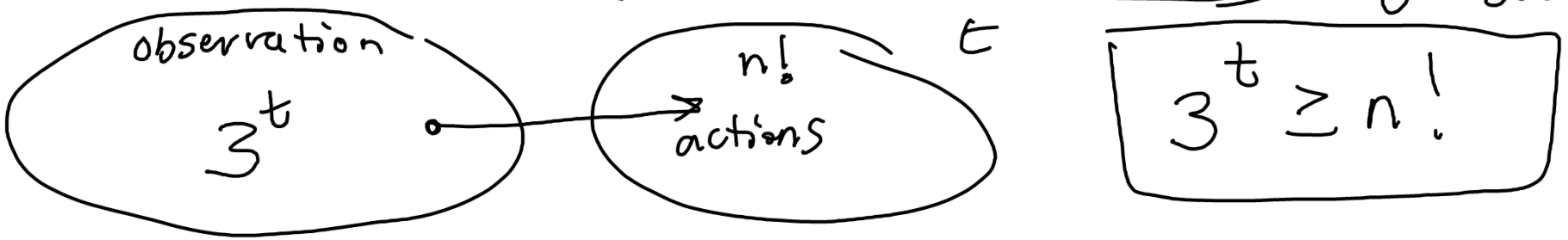
n element $\Rightarrow$ n! permutations: $n \times (n-1) \times (n-2) \times \ldots \times 1$

- Different action for each permutation
- Action chosen based comparisons $(<, =, >)$

t observation: $(<, =, <, >, <, =, \ldots)$ How many?

t Comparison

3^t observations



$3^t > n!$

$t > \log_3(n!)$

$t > k \ln(n!)$

t is $\Omega(\ln(n!))$ — lower bound

Stirling's formula: $\ln(n!) \approx n \ln n$

t is $\Omega(n \ln n)$

generic sort is at least $\Omega(n \lg n)$

$\Rightarrow$ no linear-time generic sort!