

Today: Induction on Insertion Sort, Runtimes Intro to Greedy Algs.

Insertion Sort ($A[1:n+1]$):

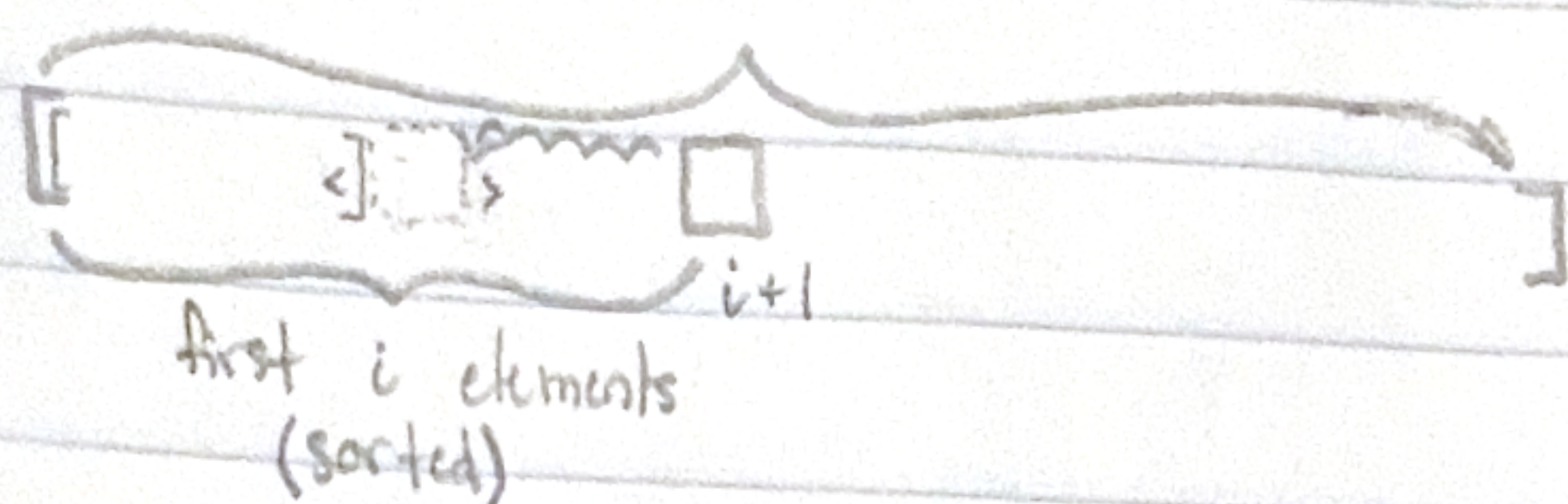
for i in $1, \dots, n$

$j = i$

while $j > 1$ and $A[j] < A[j-1]$

swap $A[j]$ and $A[j-1]$

$j--$



How to show...

Best case:

The minimum amount of work required as a function of input size

Worst case:

The maximum amount of work required as a function of input size

What we need:

- the input on which it achieves a small amount of work

- show the algorithm takes at least that time always

Upper Bound

Lower Bound

What we need:

- the input on which it takes a long amount of time

- show the algorithm never takes more than that time

Lower Bound

Upper Bound

Claim: After the first i iterations, the first i elements of A are sorted

Base Case ($i=1$): Nothing happens, but first element is sorted by default

Induction Step ($i+1$): Show that if above is true for particular i it is true for ($i+1$).

After moving this ($i+1$) item, everything to the left is sorted from before and smaller as per while loop condition, everything to the right is larger and sorted as per while loop condition, so the first ($i+1$) items are in sorted order.

Greedy Algorithms

Maximal interval scheduling Example

Input: Collection of intervals, each with start/stop times

Goal: Compute maximum # of non-overlapping intervals

Idea: - Prioritize smallest intervals

- Sort by start time, collect and delete

- Go over every possible combination

- Sort by end time, collect and delete

by $\checkmark$

by $\times$

$\sim n^2$

Exchange Argument:

We will show that this solution is at least as good as any other algorithms output solution.

1. Take any arbitrary solution S

2. Look at the first point of disagreement between S and our solution

3. Tweak S to produce S' such that

i) we don't introduce any other disagreements

ii) S' "fixes" the disagreement

iii) S' is no worse a solution than S