

CS375 Notes

9/16/26

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

for $i = 1, \dots, n$:

$j = i$

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

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

$j--$

Induction:

① Base case

② Inductive step:

- if it works for n , show

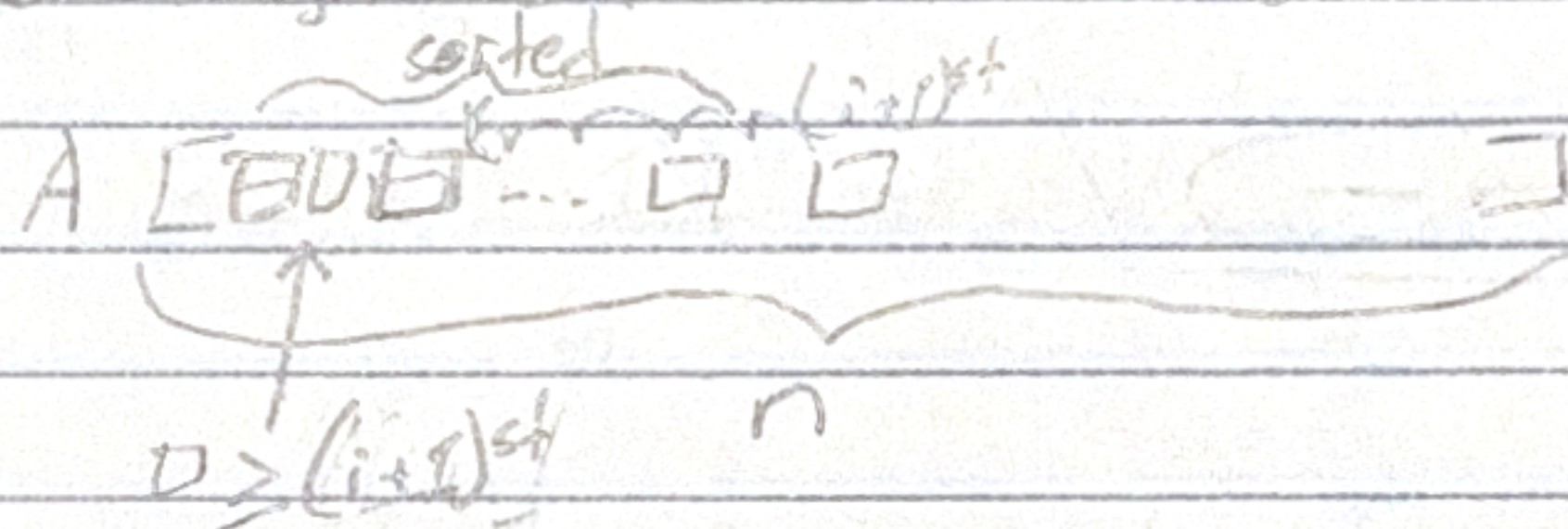
it works for $n+1$

③ Conclude

Claim: after i iterations, the first i elements of a have been sorted

Base Case ($i=1$): Nothing changes, but first element is sorted anyway

Inductive Step: Show that if the above statement is true for a particular i , it is true for $(i+1)^{st}$.



After moving the $(i+1)^{st}$ item:

- everything to the left is already sorted and smaller than it (ending condition of while loop)
- everything to the right is bigger and sorted
- So now the first $i+1$ items are sorted

Best Case:

- min amt. of work as a func. of the input size

What we need:

- Input on which it achieves
- Show the algorithm takes at least that much time

upper bound on best case

lower bound on best case

Worst Case:

- Max amt. of work as a function of the input size

What we need:

- Input that achieves
- Show the algorithm can't take more than that time

upper bound on worst case

Greedy Algorithms

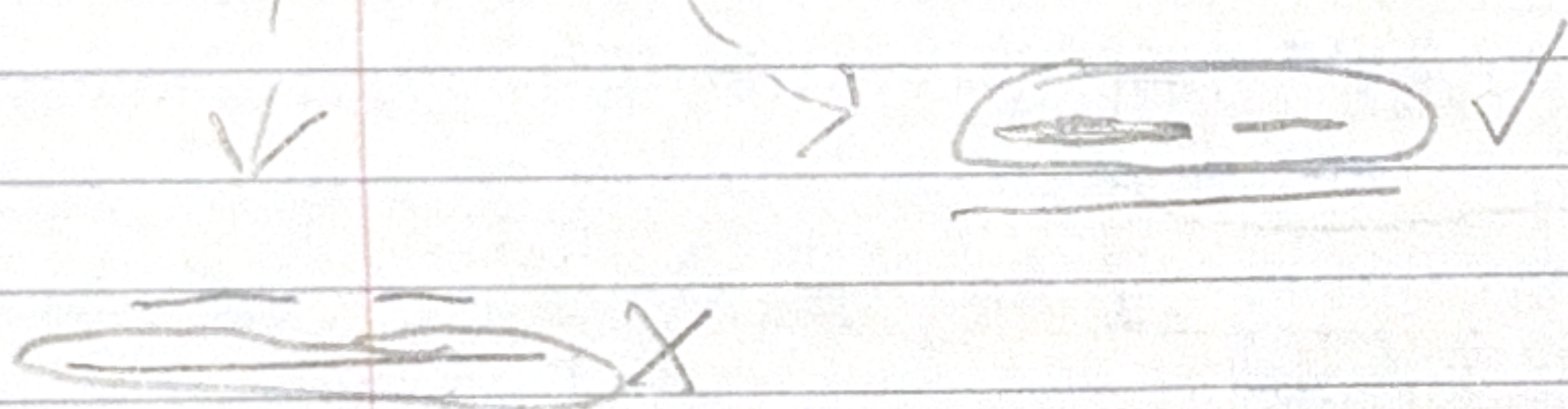
Maximal Interval Scheduling

Input: A collection of intervals - each with a given start + end time.

Goal: Compute the max # of non-overlapping intervals.

Ideas:- Prioritize smallest intervals

- Sort based off of start time, collect and delete X
- works → X - Go over every possible combination (Brute force) 2^n X
- Delete classes fully containing other classes
- Sort by end time, collect & delete ✓



Exchange Argument:

We will show that the solution is at least as good as any other solution

1. Take any arbitrary solution (S)
2. Find 1st source of disagreement btwn. algorithm and S
3. Tweak S a little bit to produce S' st:
 - i) We don't make any earlier disagreements
 - ii) S' fixes that first disagreement
 - iii) S' is no worse as a solution than S