Data Structures and Algorithms

Lab 1

TA bonus points

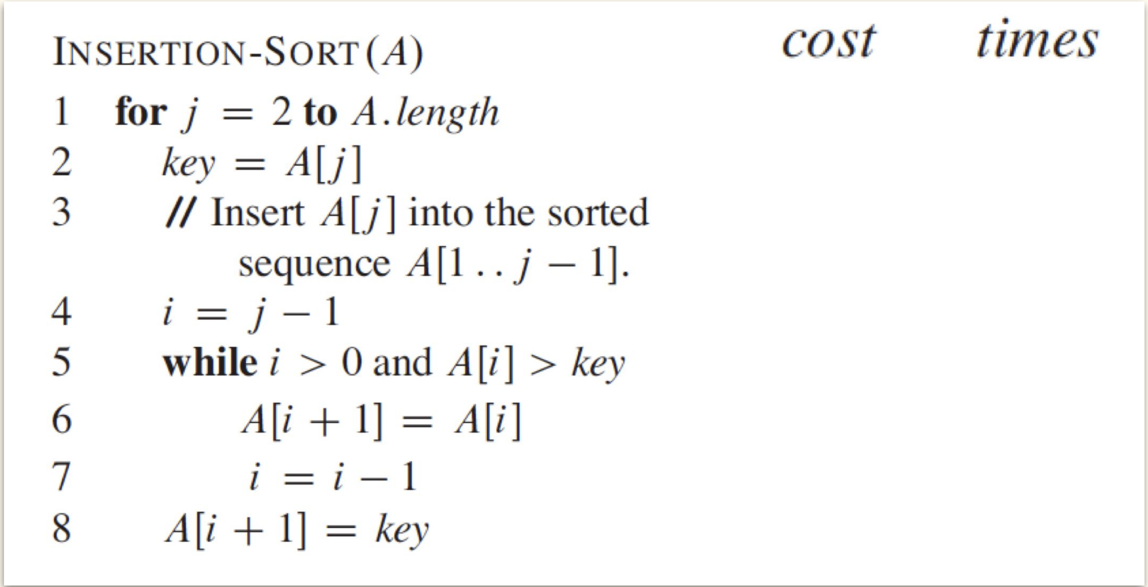
1. Each TA has **bonus 5%** to spend **for each group**
2. You get awarded **participation points** for activities in labs of your group
3. Participation points are converted to **course points** after the Final Exam:
    a. Only **borderline** students are considered
    (i.e. whose grade can be improved by bonus points)
    b. Students are considered in decreasing order of participation points
    c. Each student receives **at most 3%** in bonus points
    (or less if enough to improve the grade)
    d. Bonus points are awarded until all 5% are spent
    (or until there are no more borderline students)

Warm-up exercise: sorting

1. Write down an unsorted sequence of 5 integers
2. Rotate the papers
3. Apply any sorting algorithm you know, write down intermediate state at every step
4. Rotate the papers
5. Try to guess the algorithm and write down its name
6. Rotate papers back and mark the guess
7. Rotate again
8. Did you guess correctly?

Insertion sort

Exercise 1.1. Compute worst-case time complexity of insertion sort.



```
INSERTION-SORT(A)                                      cost    times
1  for $j = 2$ to $A.length$
2      $key = A[j]$
3      // Insert $A[j]$ into the sorted
           sequence $A[1\,..\,j-1]$.
4      $i = j - 1$
5      while $i > 0$ and $A[i] > key$
6          $A[i+1] = A[i]$
7          $i = i - 1$
8      $A[i+1] = key$
```

Exercise 1.2. Compute best-case time complexity of insertion sort.

Selection sort

Consider sorting N elements stored in an array A by finding the smallest element of A and exchanging it with the element in A[1]. Then finding the second smallest element and exchanging it with the element in A[2]. Continue similarly for the first N – 1 elements of A.
This algorithm is known as **selection sort**.

Exercise 1.3. Write pseudocode for selection sort.

Exercise 1.4. Specify the loop invariant for the main loop in the algorithm.

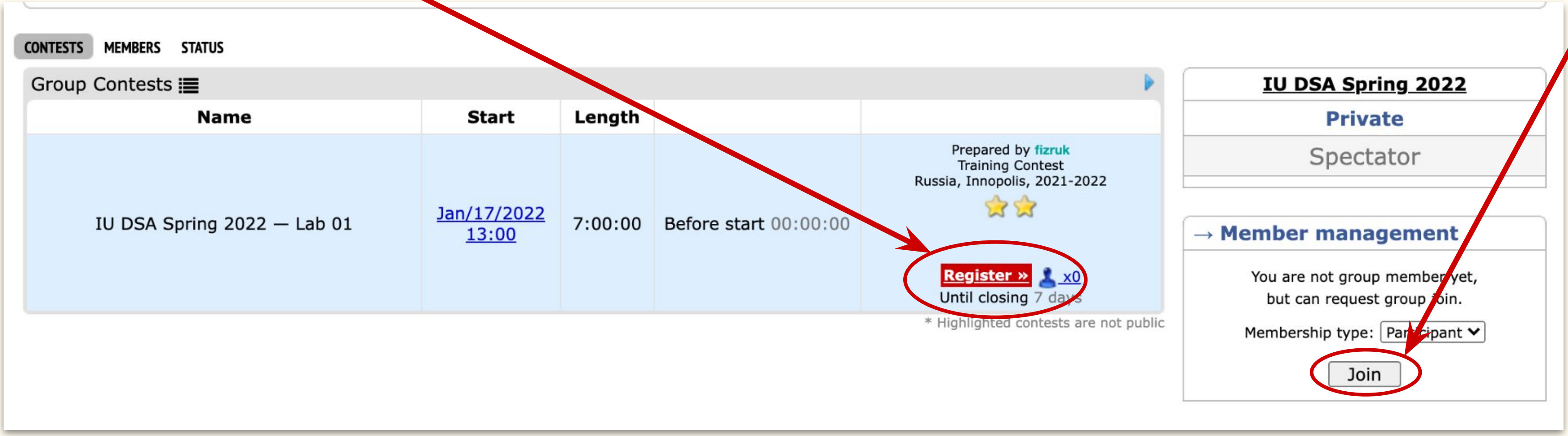
Exercise 1.5. Prove that it is enough to run the main loop for only N – 1 iterations.

Exercise 1.6. Find best- and worst-case time complexity of selection sort in Θ-notation.

Exercise 1.7. Find best- and worst-case time complexity of selection sort given that writing/modifying array takes significantly longer than reading or comparing elements.

CodeForces (for coding exercises)

1. https://codeforces.com/group/M5kRwzPJlU/contests
2. **Join** the group «**IU DSA Spring 2022**»
3. **Register** for the contest «**IU DSA Spring 2022 — Lab 01**»



Name	Start	Length		
IU DSA Spring 2022 — Lab 01	Jan/17/2022 13:00	7:00:00	Before start 00:00:00	Prepared by fizruk Training Contest Russia, Innopolis, 2021-2022 Register » x0 Until closing 7 days

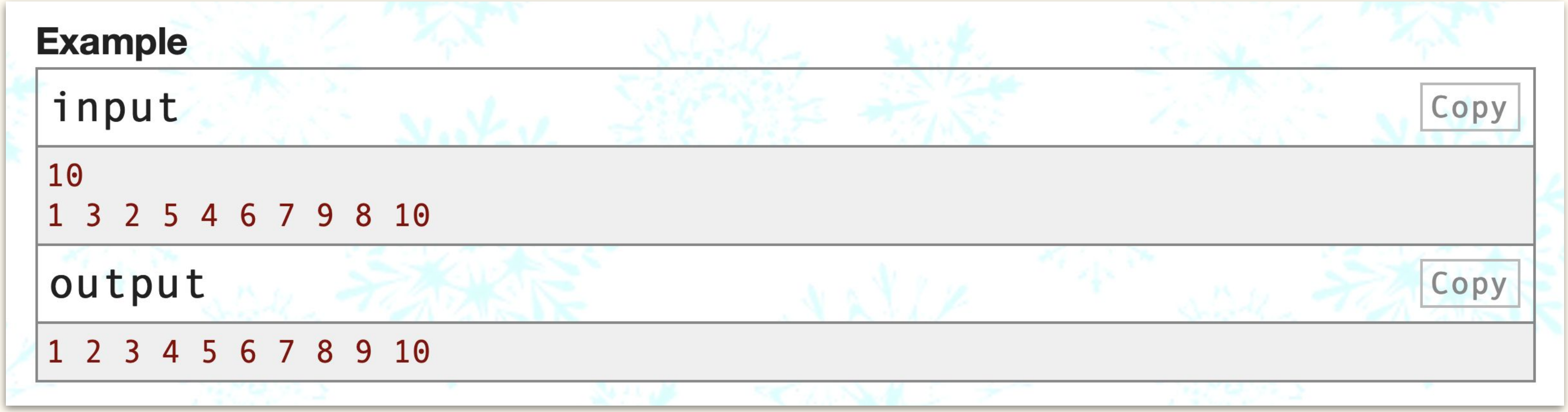
* Highlighted contests are not public

IU DSA Spring 2022
Private
Spectator

→ Member management
You are not group member yet, but can request group join.
Membership type: Participant
Join

Nearly sorted sequence

Exercise 1.8. Implement both insertion sort and selection sort.
Make a prediction of how well these algorithms might perform.
Submit both solutions to CodeForces.
Compare your prediction with results on CodeForces.
Can you explain what happens?


Example

input

```
10
1 3 2 5 4 6 7 9 8 10
```

output

```
1 2 3 4 5 6 7 8 9 10
```

Recap

- What is the best/worst-case time complexity of insertion/selection sort?
- In what situations is insertion/selection sort good?
- Can you implement insertion/selection sort?

See you next week!