Data Structures:
Active Learning 2: Introduction

YoungWoon Cha

Spring 2022

Active Learning 2

Active Learning 2

- Comparison of Sorting Algorithms
  - ** The purpose of this exercise is to
    - Help you understand the sorting algorithms you've learned in full detail.
    - Help you experience performance analysis of algorithms.
  - Part 1~5.
    - 20 points X 5 = 100 points.
- Project Presentation
  - Present your analysis results & thoughts
- Project Report
  - Source code and test result screen shots
  - Submit to CyberCampus

Comparison of Sorting Algorithms

- Part1: Implement the following random input generator
  - Given, the number of elements n,
  - The generator should generate n-sized list with the following random values:
    - Level1: positive numbers
    - Level2: integers (including positives, 0, and negatives)
    - Level3: double precision numbers (e.g. -1000.000 < number < +1000.000)
    - Level4: fixed length strings (e.g. 5-sized characters: "abcde", "vwxyz")
  - Save the generated lists as separate files.
- Notes
  - The list should include some of duplicate values so that you could check the stability.
  - Each element should keep its initial index of the order in the list
    - e.g. using a struct: struct INPUT_INT{int val, int index;};
    - Stability check: after sorting, if val1 == val2, then check id of va1 < id of val2
- Score Criterion
  - up to Level4: 20 points
  - up to Level3: 16 points
  - up to Level2: 10 points
  - only Level1: 6 points

Comparison of Sorting Algorithms

- Part2: Implement the following sorting algorithms:
  - Category1: Insertion sort, Selection sort, Bubble sort
  - Category2: Merge sort, Quick sort
  - Category3: Heap sort, Radix sort

- Recommendation
  - Reference the sample codes shared in the class.

- Score Criterion
  - Level4: 6+ methods: 20 points
  - Level3: 5 methods: 16 points
  - Level2: 4 methods: 10 points
  - Level1: 3 methods: 6 points
    - The methods should be chosen from different categories.
  - Level0: 2 methods: 2 points

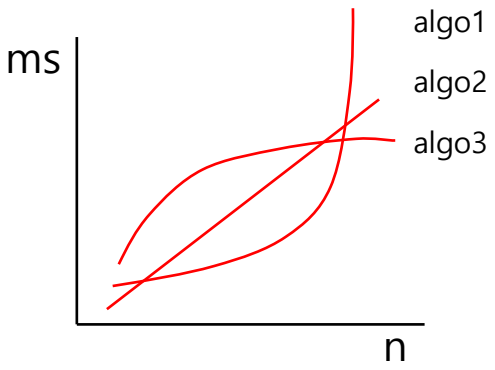
Comparison of Sorting Algorithms

- Part3: Implement the following sorting evaluator:
  - Level1: Load the generated lists from files / Save the evaluation results as separate files.
  - Level2: The sorting correctness
  - Level3: The running time
  - Level4: The stability correctness

- Score Criterion
  - Level4: 20 points
  - Level3: 16 points
  - Level2: 10 points
    - If Level2 is wrong, you will get zero points in the total score.
    - If Level2 shows the sorting algorithm you implemented is wrong, this should not be counted in Part2.
  - Level1: 6 points

Comparison of Sorting Algorithms

- Part4: Plot the following performance comparison graphs
  - Run each sorting algorithm on different n, measure its running time.
    - You should evaluate the running time multiple times for each n, then take an average of the records for fair experiments.
  - Plot the graph for all sorting algorithms that shows the running time over different sized n.
    - Include the result, only if the sorting correctness is passed.
    - If the sorting algorithm theoretically does not work for that specific data type, you don't need to add it to the graph.
    - Level1: the evaluation on positive numbers
    - Level2: the evaluation on integers
    - Level3: the evaluation on double precision numbers
    - Level4: the evaluation on fixed length strings
- Score Criterion
  - up to Level4: 20 points
  - up to Level3: 16 points
  - up to Level2: 10 points
  - only Level1: 6 points

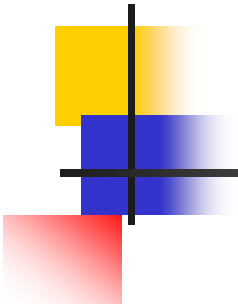


Project Presentation

- Part5: Submit 10-minutes Video Presentation Link
  - Submission Due on June 6th (Mon)
    - Team introduction
      - list all members & roles
      - Contribution percentage (Kim: 25%, Park: 30% …)
    - Specify your Levels of Achievements for part 1 ~ 4.
    - Specify your Experimental Environments
      - e.g. cpu, memory, n …
    - Performance Comparison Analysis
      - Including your thoughts
- Score Criterion
  - Level4: 20 points
  - Level3: 16 points
  - Level2: 10 points
  - Level1: 6 points

Project Report

- Submit [Team-A].Zip file
  - Submission Due on June 13th (Mon)
  - /code/
    - Copy the entire visual studio project here.
    - comment the authors of each function in codes
  - /data/
    - Leave your dataset here.
  - /eval/
    - Leave the performance evaluation files here.
  - Team-A.pdf (Documentation)
    - Team introduction (list all members & roles)
    - Contribution percentage (Kim: 25%, Park: 30% …)
    - Achievement table (self-evaluation, leave the Part5 blank)
    - Attach the Codes & Result screenshots for each problem
      - You can reuse the presentation slides



End of Lecture