Lecture 22

CSE 331

Oct 21, 2019

Mid-term-I grading

Hopefully done by tonight

Problem 2 Coding project out

note

stop following 59 views

Problem 2 on Coding Project is now live!

Apologies again for the delay in getting this done but Autolab is now accepting submissions for Problem 2 for the coding project. The coding project webpage has been updated with the required coding details:

http://www-student.cse.buffalo.edu/~atri/cse331/fall19/coding-project/index.html

Few things to keep in mind:

- As with problem 1 (@801) This is group submission-- please see the webpage for instructions on how to do so. *Please follow the instructions EXACTLY. Not following the instructions might make the group submission on Autolab not behave as intended.*
- You have to form your group **AGAIN** for Problem 2 on Autolab-- it unfortunately does not carry over from Problem 1.
- **Please download the zip for Problem 2 and use that for Problem 2 submission.** In particular, do NOT use the zip for Problem 1 for Problem 2.
  - The C++ template is not slow anymore (though Problem 1 will remain slow-- we decided not to change it since Problem 1 is due soon).
- I hope to have Problem 3 released in a few days.

If you have questions, please post on piazza! Or go to office hours. Have fun :-)
#pin

coding_mini_project

edit · good note | 1

Updated 10 hours ago by Atri Rudra

Problem 1 due this Friday!

This is an easy problem

Once you understand the problem and get familiar with the template, to get full points you only will need to add a couple of lines of code to `Solution.py`.

Some reminders about MP

Submitting as a group on Autolab

As mentioned above, you will be working on this project as a group. This mean y'all will also **submit your solution file as a group**. Here is a quick primer on how to form a group on Autolab and to submit as a group (opens in a new page).

The same group for ALL problems

You are expected to work in the **same** group for **ALL** problems in the coding project. **Not doing so will be considered to be an academic integrity violation.** Please note that you will have to sign up your group separately for each problem on Autolab and we will check for group consistency at the *end of the semester*. So please make sure you follow this rule.

There is ONE exception to the above rule: It is OK if the group becomes smaller (e.g. if a group member resigns) but in that case another student cannot be added to group. If you need clarification on this, please get in touch with Atri.

There are no optimal algorithms known!

Other than the first problem, we do not know of optimal algorithms to solve the rest of the problems (and we suspect that doing so is not possible (definitely not within a semester). Note that this is unlike the HW Q3s where your code is supposed to always output the optimal/correct solution: i.e. you will have to think of algorithms where you might not be able to prove any guarantee on how good your output is.

Problem 2 specific reminders

Make sure you use skeleton code for Problem 2!

While the skeleton code for this problem is similar to that of Problem 1, there are one major and some small differences. The major one is that `Solutions.cpp` now runs in a reasonable time and is much faster than the same functionality in Problem 1. This also needed the addition of another file `Global.h` (which you can safely ignore).

The above change led to some other minor changes (e.g. the function that you need to write is called `outputPaths()` instead of `outputDistances()`). **We VERY STRONGLY encourage you to download the zip for this problem and work on it separately from your work on Problem 1! In particular, we do not take responsibility for any weird behavior if you use the wrong zip :-)**

Grading Guidelines

The grading works a little differently for this project.

Each testcase is worth 5 points. The number of testcases for each problem depends on the maximum points (max_points) achievable, and is equal to $\frac{max\_points}{5}$. For eg. Problem 1 has one testcase, since it is worth 5 points, Problem 2 has two testcases, since it is worth 10 points and so on.

For Problem 1, you get the full 5 points if your revenue matches ours and 0 otherwise.

Except for Problem 1, there is partial grading for each testcase. The number of points awarded to you depend on how well your solution's revenue compares with our revenue.

For other problems, the thresholds are outlined below, the numbers on the left indicate the ratio of (your solution's revenue - revenue of optimal Solution for Problem 1) and (our revenue - revenue from optimal Solution for Problem 1) in percentage, and the right half indicates the points achieving that ratio will award you.

- $[100, 80]$ -> 5 points
- $[80, 60)$ -> 4 points
- $[60, 40)$ -> 3 points
- $[40, 20)$ -> 2 points
- $[20, 5)$ -> 1 points
- $[0, 5]$ -> 0 points

Video project submission open

note

0 views

Video mini-project submission open

Sorry for the delay but now Autolab is accepting submissions on your video as part of your video mini-project. The mini-project page has all the details on what is needed and what is expected:

http://www-student.cse.buffalo.edu/~atri/cse331/fall19/mini-project/index.html

I made one change from the original instructions: each of the three must submit the PDF with the link **individually** (even though this is a group project) and the PDF must be the same for all the group members.

mini_project autolab

edit · good note | 0

Updated Just now by Atri Rudra

! ALL group members submit!

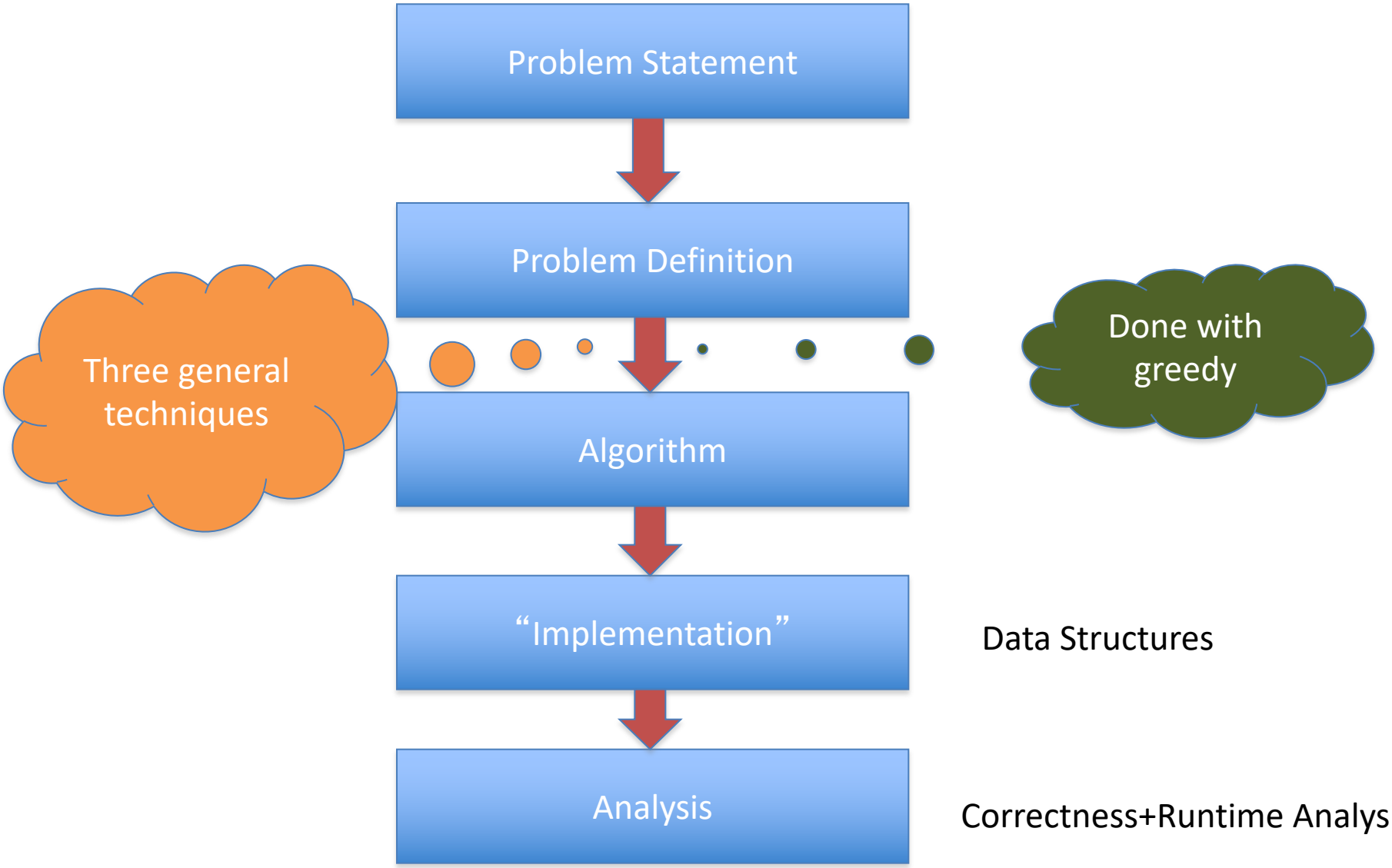
Each group member must submit the PDF. *Further, the PDF must EXACTLT be the same for all three group members.*

Questions?

Guest Lecturer for rest of the week

Erdem Sariyuce

High Level view of the course



Trivia

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

“Patch up” the solutions to the sub-problems for the final solution

Sorting

Given n numbers order them from smallest to largest

Works for any set of elements on which there is a total order

Insertion Sort

Input: $a_1, a_2, \ldots, a_n$

Output: $b_1, b_2, \ldots, b_n$

$O(n^2)$ overall

Make sure that all the processed numbers are sorted

$b_1 = a_1$

for $i = 2 \ldots n$

Find $1 \le j \le i$ s.t. a_i lies between b_{j-1} and b_j

Move b_j to b_{i-1} one cell "down"

$b_j = a_i$

O(log n)

O(n)

a	b
4	4
3	3
2	2
1	1

Other $O(n^2)$ sorting algorithms

Selection Sort: In every round pick the min among remaining numbers

Bubble sort: The smallest number “bubbles” up

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

“Patch up” the solutions to the sub-problems for the final solution

Mergesort Algorithm

Divide up the numbers in the middle

Sort each half recursively

Merge the two sorted halves into one sorted output

Unless n=2

How fast can sorted arrays be merged?

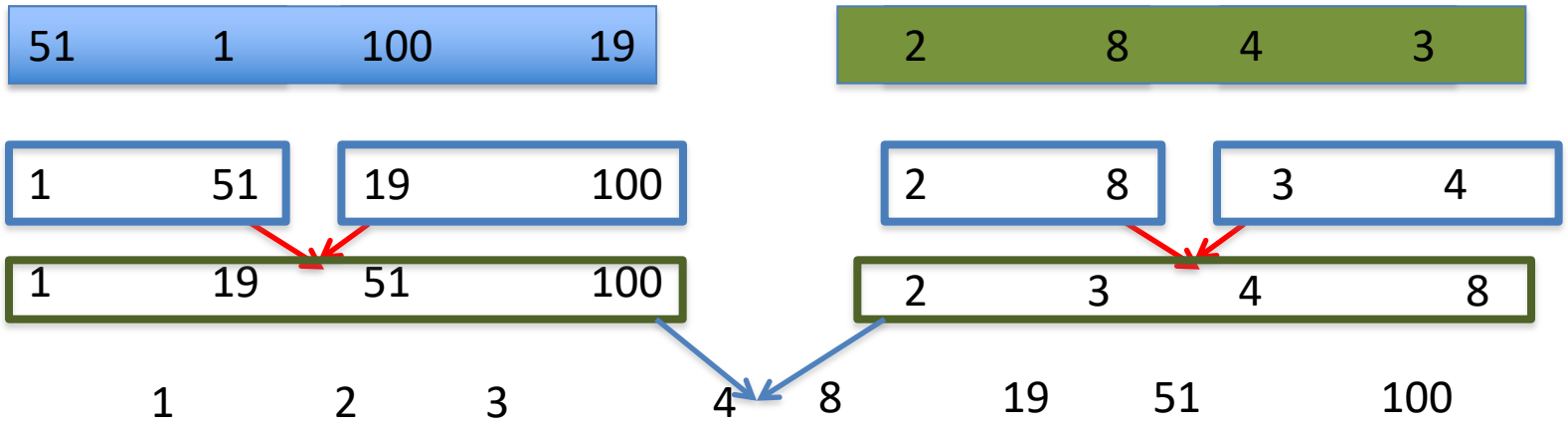
Mergesort algorithm

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

```
MergeSort( a, n )

    If n = 1 return the order a_1
    If n = 2 return the order min(a_1,a_2); max(a_1,a_2)

    a_L = a_1,..., a_n/2

    a_R = a_n/2+1,..., a_n

    return MERGE ( MergeSort(a_L, n/2), MergeSort(a_R, n/2) )
```

An example run



MergeSort(a, n)

If n = 1 **return** the order a_1

If n = 2 **return** the order min(a_1,a_2); max(a_1,a_2)

$a_L = a_1,\ldots, a_{n/2}$

$a_R = a_{n/2+1},\ldots, a_n$

return MERGE (MergeSort(a_L, n/2), MergeSort(a_R, n/2))

Correctness

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

```
MergeSort( a, n )

  If n = 1 return the order a_1
  If n = 2 return the order min(a_1,a_2); max(a_1,a_2)

  a_L = a_1,..., a_n/2

  a_R = a_n/2+1,..., a_n

  return MERGE ( MergeSort(a_L, n/2) , MergeSort(a_R, n/2) )
```

By induction on n

Inductive step follows from correctness of MERGE

Rest of today's agenda

Analyze runtime of mergesort algorithm