Algorithms Programming Project I
Greedy Algorithms

1 Problem Definition

You are a house painter that is available from day $1 \dots n$ (inclusive). You can only paint one house in a day. It also only takes one day to paint a house. You are given m houses. For each house i, you are also given $startDay_i$ and $endDay_i$ for $i = 1, \dots, m$. The house i can only be painted on a day between $startDay_i$ and $endDay_i$ (inclusive). The given houses are already sorted primarily on $startDay$ and secondarily on $endDay$ (in case of equality of the $startDay$). You are tasked to find the maximum number of houses that you can paint.

2 Greedy Strategies

For each of the following greedy strategies, you must i) design a $O(n + m\log(m))$ algorithm; ii) provide an instance where the strategy yields an optimal solution; iii) provide an instance where the strategy does not yield an optimal solution or prove that it is an optimal strategy. Your algorithm for Strat1 must have a $\Theta(n)$ running time. The remaining strategies must have a $\Theta(n + m\log(m))$ running time.

Strat1 Iterate over each day starting from day $1 \dots n$. For each day, among the unpainted houses that are available to be painted on that day, paint the house that started being available the earliest.

Strat2 Iterate over each day starting from day $1 \dots n$. For each day, among the unpainted houses that are available that day, paint the house that started being available the latest.

Strat3 Iterate over each day starting from day $1 \dots n$. For each day, among the unpainted houses that are available that day, paint the house that is available for the shortest duration.

Strat4 Iterate over each day starting from day $1 \dots n$. For each day, among the unpainted houses that are available that day, paint the house that will stop being available the earliest.

Hint:- While iterating over each day, maintain a priority queue that contains the unpainted houses that are available to be painted on the current day.

3 Programming Tasks

Once you complete the algorithm design tasks, you should have an implementation for each of the following programming procedures:

Task1 Give an implementation of Strat1.

Task2 Give an implementation of Strat2.

Task3 Give an implementation of Strat3.

Task4 Give an implementation of Strat4.

4 Language/Input/Output Specifications

You may use Java, Python, or C++. Your program must compile/run on the Thunder CISE server using gcc/g++, Python-3 interpreter, or standard JDK. You may access the server using SSH client on thunder.cise.ufl.edu. You must write a makefile document that executes first any compilation tasks then will run each task. First the command **make** will be run. Then each task should have its own command in the format **make run#** where the number corresponds to each task e.g. when **make run3** is called from the terminal, your program needs to execute the implementation of Task3.

Input. Your program will read input from standard input (stdin) in the following order:

- Line 1 consists of two integers n and m separated by a single space.
- next m lines consists of 2 integers with each line being the start and end date of when a house can be painted separated by a single space.

Output. print a set of indices that correspond to the houses you should paint to achieve the maximum number of houses in the order you paint them.

- Each house index should be outputted on a single line separated by a single space.

For convenience assume that $1 \leq n, m < 10^5$, and $\forall i \;\; 0 \leq startDay_i < endDay_i < 10^5$.

5 Experiments and Report

You should conduct an experimental study to test the performance and scalability of your algorithms/implementations. Your report should include at least the following components: i) Team members; ii) Greedy Strategies; iii) Experimental Comparative Study; iv) Conclusion.

5.1 Team Members

You are allowed to work as teams of (at most) two students on this programming assignment. If you decide to work as a team of two, clearly state the names of team members and describe the main contributions of each member.

5.2 Greedy Strategies

For each of the greedy strategies, you should clearly describe the algorithm and analysis. You must also provide examples that supports and contradicts each strategy. When a strategy is optimal, you must argue its correctness.

5.3 Experimental Comparative Study

You are expected to test your implementations for their optimality. For this purpose, you should create randomly generated input files of various sizes. The exact size of the experimental data sets that your program can handle depends on the quality of your implementation. For instance, you might want to choose $n = 1000, 2000, 3000, 4000, 5000$ Then, you compare the number of houses for each implementation for each input size. You should plot a grouped histogram with number of houses (y-axis) against input size (x-axis). You should group together all implementation for a given input size.

5.4 Conclusion

Summarize your learning experience on this project assignment. For each programming task, comment on ease of implementation and other potential technical challenges.

6 Submission

The following contents are required for submission:

1. **Makefile**: Your makefile must be directly under the zip folder. No nested directories. Do not locate the executable file in any directory either.
2. **Source code**: should include detailed comments next to each non-trivial block of code.
3. **Report**: The report must be in PDF format.
4. **Bundle**: Compress all your files together using a zip utility and submit through the Canvas system. Your submission should be identified with your last name and first name, i.e., **LNameFName.zip**. For teams of two, use **LName1FName1LName2FName2.zip** and make only one submission on canvas.

7 Grading Policy

Grades will be based on the correctness & efficiency of algorithms and the quality of the report:

- **Program 50%.** Correct/efficient design and implementation/execution. Also make sure to include comments with your code for clarity.
- **Report 50%.** Quality (clarity, details) of the write up on your design, analysis, programming experience, and experimental study.

8 Bonus [up to %20 points]

Design, analyze, and implement an efficient version of the Optimal strategy.

8.1 Algorithm Design

Design and analyze (time, space and correctness) of an $\Theta(m \log(m))$ algorithm based on the greedy strategy that you proved to be Optimal in Section 2.

8.2 Programming Task

This task has the same input and output specifications as the previous tasks.

Task5 Give an $\Theta(m \log(m))$ implementation of the Optimal strategy for the problem.

8.3 Experimental Study

You should conduct an experimental study to test the performance and scalability of your implementation. You should compare the implementations of the optimal solutions with running times of $\Theta(m \log(m))$ and $\Theta(n + m \log(m))$. You should use various values of m and (sufficiently large) n to show the disparity between the running times. For each m value, you should provide a plot of the running times of the algorithms (y-axis) against n (x-axis).