CSC 323-01 Algorithm Design and Analysis, Fall 2019
Instructor: Dr. Natarajan Meghanathan

Quiz 5: Greedy Algorithm for Selecting the Longest Sequence of Non-Overlapping Activities based on their Finish times

Due by: Oct. 24th, 11.59 PM (in Canvas)

In this quiz, you will implement the greedy algorithm to determine the longest sequence of non-overlapping activities from a given list of activities (their IDs) along with their start times and finish times.

Greedy Algorithm and Example: Recall, the greedy algorithm we looked in class first sorts the activities on the basis of their finish times. It then goes through the sorted activity list. The activity with the smallest finish time is the first activity chosen. Let *current finish time* be the variable used to keep track of the finish time of the latest activity chosen in the list. All activities whose start times are less than or equal to that of the current finish time are ruled out from consideration. The next activity chosen is the activity (from the sorted list based on finish times) whose start time is greater than the current finish time; the finish time of this activity is thereafter considered to be the current finish time and we proceed further. Below, I illustrate the working of the algorithm with an example.

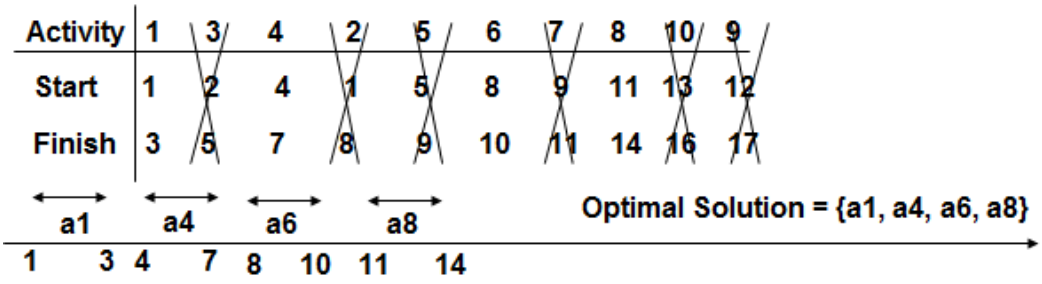
Given List

Activity	1	2	3	4	5	6	7	8	9	10
Start	1	1	2	4	5	8	9	11	12	13
Finish	3	8	5	7	9	10	11	14	17	16

Sorted List (based on Finish Times)

Activity	1	3	4	2	5	6	7	8	10	9
Start	1	2	4	1	5	8	9	11	13	12
Finish	3	5	7	8	9	10	11	14	16	17

Sorted List (Selected/ Discarded Activities)



Provided Code: You are given a startup code (ActivitySelection_StudentVersion.cpp) wherein the main function creates three arrays: the activity IDs, their start times and finish times. You need to just input the number of activities.

You need to first implement the Selection Sort algorithm (with the arguments passed as indicated) to sort the activities based on the increasing order of finish times. Note that at the end of the sorting process, the contents in the arrays representing the activity IDs and the start times should be also rearranged to correspond to the sorted order of the finish times (as also seen in the example above).
I have provided another cpp file called: SampleSelectionSort.cpp that demonstrates (given two arrays: IDs and timeValues) how to sort second array (timeValues) and rearrange the contents of the first array (IDs) based on the sorted order of the second array. You could use this code as a guideline to implement the selection sort algorithm in the ActivitySelection_StudentVersion.cpp file.

After the contents of the activity IDs, start times and finish times are rearranged according to the increasing order of finish times, the next step is to implement the greedy algorithm (explained above) to determine and print the longest sequence of non-overlapping activities. A sample screenshot is given below.

```
Enter the number of activities: 10
Activities and their start/finish times (before sorting)
1 2 3 4 5 6 7 8 9 10
1 4 6 9 12 14 15 17 18 19
4 5 10 13 16 18 20 21 22 22

After sorting in the increasing order of the finish times
Activity List: 1 2 3 4 5 6 7 8 9 10
StartTimes: 1 4 6 9 12 14 15 17 18 19
FinishTimes: 4 5 10 13 16 18 20 21 22 22

Activities Selected by the Greedy Algorithm
1 3 5 8
```

Submission (in Canvas)

1) Submit the entire ActivitySelection_StudentVersion.cpp file with the Selection Sort algorithm and the greedy algorithm for activity selection implemented.

2) Submit a screenshot (as a jpeg file) of the output of your code for the number of activities to be 15.