CSE3081-1: Design and Analysis of Algorithms (Fall 2018)
Machine Problem 1: Maximum Subsequence Sum

Handed out: September 20, Due: October 9, 11:59PM (KST)

1. Goal

The goal of this MP is to understand how different algorithms perform differently while producing the same result for the same problem.

2. Problem Description

You are given a sequence of integers, such as the following.

3 -8 9 7 -30 22 -4 19 -6 5

Your goal is to find a **subsequence which has the maximum sum**.

By choosing a starting point and end point in the sequence, you create a subsequence. For example, the sequence indicated in red below is a subsequence.

3 -8 9 7 -30 22 -4 19 -6 5

The subsequence starts at **index 2**, and ends at **index 8**. Note that the index of the first number in the sequence is **0**. Now, the sum of the subsequence is:

9 + 7 -30 + 22 -4 + 19 - 6 +5 = **22**

There are several special cases which you need to consider when implementing algorithms.

(1) If all numbers are negative, the maximum sum subsequence will be a single largest number. Below is the example. The red subsequence is the answer.

-2 -5 -9 -1 -7 -8 -6 -5 -12 -3

(2) There could be multiple subsequences with the same sum. In this case, the answer should be the one which has the smallest index for both left and right end. In the below, the red sequences are the answers.

-10 -11 -12 3 4 5 -13 -14 -15 5 4 3 -16 -17 -18

-10 -11 -12 0 0 3 4 5 0 0 -13 -14 -15

3. Your task and requirements (Read Carefully!)

(1) You will write a **C/C++** program which takes an array of integers as input and finds the subsequence with the maximum sum. (We have seen this problem in class.)

(2) You should also write a **Makefile**. The TA will build your code by running 'make'. It should create the binary file.

(3) When created, your binary file should be named **mss20120001**. The numbers should be your student ID. There should be only a single binary file. It is up to you to make a single or multiple source code files.

(4) Your code should build and run on a Linux machine. Even if you write your code using Microsoft Visual Studio on Windows, your code should compile on Linux unless you use Windows-specific API. Still, you should check and make sure it runs on a Linux machine.

(5) Your program should take two command-line arguments: The first one is the **input file name**, and the second one is the **algorithm index**. An example run is:

```
$ ./mss20120001 input00001.txt 2
```

(6) There are three different types of algorithms. Their indices are:

1 – $O(n^2)$ algorithm
2 – O(nlogn) algorithm
3 – O(n) algorithm

You do not need to implement the $O(n^3)$ algorithm in this MP.

(7) The input file contains a single line of numbers. An example input file is the following:

```
10 3 -8 9 7 -30 22 -4 19 -6 5
```

The first number indicates the number of elements in the array. In the above example, there are 10 elements in the input sequence (3 -8 9 7 -30 22 -4 19 -6 5). Note that the first '10' is not a part of the input sequence.

(8) You program should produce an output file. The name of the output file must be "result_**inputfilename**". In the above example where the input file is "input00001.txt", the corresponding output file should be named "result_input00001.txt". The output file should have seven lines, containing the following items:

1st line: input file name
2nd line: algorithm index
3rd line: input size (the number of elements in the original sequence)
4th line: index of the leftmost number in the subsequence found
5th line: index of the rightmost number in the subsequence found
6th line: sum of the subsequence
7th line: running time in milliseconds

To measure running time, you can use functions such as `clock()`.

In the above example, the maximum sum subsequence is found as the following:

```
3 -8 9 7 -30 22 -4 19 -6 5
```

So, the output file should contain:

input00001.txt
2
10
5
7
37
1.4

Note that **the index of the first number is 0**.

3. Report

In addition to code files, you should also submit a report document. In the document, **you will describe how the running time of each algorithm grows as the input size becomes larger**.

You can use tables, graphs, or any other visualization methods so that the reader could clearly understand the performance difference of the algorithms.

Your document does not need to be long, but you should definitely include the following:

- Experiment environment: The hardware specification of the machine where you did your experiments, such as CPU speed, memory size, and OS type and version. Obviously, you should do all your experiments in one machine for fair comparison.

- Experiment setup: This is basically how you chose parameters for your experiment. What is the metric that you measured, and what is the range of input size, etc.

- Your comments on the experience: This is what you have observed from doing the experiments. (For example, you learned how the algorithms will perform when the input size becomes large. Can you see the trend for yourself? What if the input size is small?) Just write anything you observed, regardless of whether you can explain the phenomena or not.

4. Submission

You should submit the Makefile, the source code(s), and the report document. Make the file into a zip file named cse3081_mp1_**20120001**.zip. The numbers should be your student ID. Where to submit your file will be announced by TA at the cyber campus.

5. Evaluation Criteria

(1) Your implementation: 60%

- Does your implementation produce correct results for all three algorithms?
- Do you follow the requirements faithfully? (such as Makefile and output format)

(2) Comments and coding style: 10%

- Is the code organized and easy to read? (indentation, spaces, styles)
- Do variable names reflect what they really are?
- Are there enough comments that explains what the code blocks are doing?

(3) Your report: 30%

- Can the reader understand the object and result of your experiments?
- Can the reader see the result visually?

6. Notes

- You should write your own code. You can discuss ideas with other students, but definitely should not copy their work. For this particular MP, since you can see the example code on the lecture slides, you will lean towards following the code exactly. My advice is that you just catch the idea, and start writing your own code without looking at the example code.

- As announced in the first class, duplicates will receive zero grade.

- 10% of the evaluation goes to the practice of writing the code. Your code should not just work, but should also look good to other programmers. There are many books and materials on code writing practice. One example is a book called “Code Complete” by Steve McConnell.

- You may write your program in your own environment (Windows, Linux, MAC). But you should test your code on the Linux machine before submitting the file. Each of you will be given an account to the department Linux server.

- Remember that the TA will place your files in a directory, build your code using ‘make’, and run the code with the test inputs. Make sure everything works before submitting your work.

- Do NOT submit binary files. Submit only the files listed in the Submission section.