Programming Assignment 1

Maximum Subsequence Sum

In this programming assignment, you will implement four algorithms for the Maximum Subsequence Sum problem, which is described in the Lecture Note. Formally, it is

- Given (possibly negative) integers $A_1, A_2, \ldots, A_N$, find the maximum value of $\sum_{k=i}^{j} A_k$. For convenience, the maximum subsequence sum is 0 if all the integers are negative.
- Example: for input, -2, 11, -4, 13, -5, -2, the answer is 20 (A2 to A4).

The four algorithms are given in the Lecture Note. You need to implement them in the given `main.c` file. Note that `Max3()` in the algorithm 3 is not given. You need to implement it by yourself. Based on your implementation, please answer the following questions.

1. Explain the code of `GenerateNumbers()` line-by-line. Your answer should include the roles of `malloc()` and why the type casting is required. Also include why check if `A` is `NULL`. Please google if you need help message for a C function. (e.g. google "malloc in C")

2. Explain the role of `srand()`, `exit()` and `free()` in `main()`.

3. Explain how the code estimates the running time of a function. Please include the role of `clock()` function in `main()`.

4. When N=1000, run all four algorithms and check they resulted in the same answers. What is the answer when `Seed` is 0. Please generate a screen shot like the below. Attach the screen shot in your report.

```
Algorithm 1: The MaxSum is 12345 (0.5890 sec)
Algorithm 2: The MaxSum is 12345 (0.0010 sec)
Algorithm 3: The MaxSum is 12345 (0.0000 sec)
Algorithm 4: The MaxSum is 12345 (0.0000 sec)
```

5. Set `Seed` to the last four digits of your student Id, and repeat problem 4. Provide your code of the whole `main.c`.

6. Set `Seed` to 0. For N=100, 200, 500, 700, 1000, 1500, 2000, estimate the running time of algorithm 1. If the algorithm runs too fast to measure the running time, repeat the task many times and estimate the average running time. For example, run algorithm 1 100 times for N=100 and measure the whole running time. By dividing 100, estimate the running time of one run. Draw a plot N vs. running time (You may want to use log scale), and compare with theoretical running

time calculation.

7. Set `Seed` to 0. For N=100, 500, 1000, 5000, 10,000, 100,000, estimate the running time of algorithm 2. If the algorithm runs too fast to measure the running time, repeat the task many times and estimate the average running time. Draw a plot N vs. running time (You may want to use log scale), and compare with theoretical running time calculation.

8. Repeat problem 7 for algorithm 3 for N=100, 1000, 10,000, 100,000 and 1,000,000.

9. Repeat problem 7 for algorithm 4 for N=100, 1000, 10,000, 100,000 and 1,000,000.

10. The Minimum Subsequence Sum problem is similar, but finds the minimum sum. Please write an efficient function and, run when `Seed` is 0 and N=1,000,000. What is the minimum sum and how long does it take? Please provide your code of `MinSubseqeuenceSum()`.

Please submit a report including answers of the above questions, screen shots, source codes in "one single electronic file", which can be a format of doc, hwp or pdf. No paper report is accepted. Please upload your report to the Blackboard 'Assignments'.

If you have any questions for this project assignment, please contact me or TA.