CS1550 Algorithmic design

Tom Bitterman, David Mestel

Assignment 1. Deadline: 23:59, Tuesday 16 April

1 Introduction

The written exercises are optional and will not form part of your course grade. However, you are strongly encouraged to do them, and you should submit solutions through Canvas (by midnight on Tuesday) to get feedback from your TA.

For the programming problems ('The important machine' parts A and B, and 'Two's company'), you need to submit your solutions via Codegrade, which can be reached from the assignment page on Canvas. Your program needs to read a filename from the terminal, read the input from the named file, and print the output to the terminal. The Canvas page for each task contains an example `.java` file demonstrating the I/O. Note that **the file you upload must have the same name and main class as the example, otherwise the grading will fail**. An example input file is also included, and its expected output.

The deadline is 23:59 on Tuesday 16 April, and this deadline is strict. **You are therefore strongly encouraged to submit solutions well before the deadline, in case of technical problems. Technical problems with submitting to Codegrade will not be considered a reason to extend the deadline.** The runtime time limit for each test case is 20 seconds. You must not try to exploit the automatic grader in any way, or attempt to exfiltrate the test input data—doing so will be considered cheating.

2 The important machine

Consider a single machine scheduling problem where we are given a set, T, of tasks specified by their start and finish times. All tasks run on only one, important, machine, and the important machine can only perform one task at a time. If a task cannot be scheduled at its assigned starting time it cannot be performed at all. We wish to maximize the number of tasks that this single machine performs.

The rules of scheduling are as follows (writing $[a, b]$ for the start and end times of a task):

- Times $[a, b]$ means the task must start at time step a (or not at all).

- Times $[a, b]$ means that the task ends before time step b starts. This agrees with the real-world convention for scheduling meetings (e.g., if a meeting is from 1-3, it starts at 1 o'clock and ends at 3 o'clock. It takes up the entire hours of 1 o'clock and 2 o'clock, and none of 3 o'clock).
- Therefore, the sequence $[a, b], [b, c]$ is allowable.

Note that in the input the tasks are not necessarily ordered according to start time, finish time, duration, or any other criteria. There may be multiple tasks with the same start and finish times.

(A) One greedy algorithm to schedule jobs on this machine is as follows:

- start from the first time slot and go chronologically
- if the machine is unused, schedule a task $t \in T$ that starts at that time slot (if one exists)
- if more than one task starts at that time slot, schedule the shortest task

Write a program that will accept a set of tasks and use the above algorithm to schedule them. Your program will output the tasks that the important machine can accomplish.

The first line of the input will be an integer $k < 10^7$. Each of the following k lines will consist of a pair of integers ab, with $a \leq b \leq 10^7$, the start and finish times of the kth task.

The output of your program should be similar: a sequence of lines, each containing a pair of integers ab, the start and finish times of a scheduled task. You must list the scheduled tasks in chronological order.

Name your program `MachineA.java` for submission to Codegrade.

(B) The algorithm described in part 2 is not optimal (can you think of an example where it fails?). Your assignment is to design an algorithm which is guaranteed to always schedule the maximum possible number of jobs. The input will be in the same format as before, but now you should output only a single integer m, the maximum number of jobs that can be scheduled.

Name your program `MachineB.java` for submission to Codegrade.

3 Two's company, three's a crowd?

In the lectures, we saw how to encode text with the Huffman coding, assigning a code word to each letter. However, we could choose to look at our letters in pairs (so that we now have an alphabet of size up to $26^2 = 676$), and assign a codeword to each digraph. This is better in some circumstances: for example the text `aaaaaa` encodes to `000000` under the ordinary Huffman coding, but `000` if we encode by digraphs. Of course we can do the same thing with 'trigraphs' (so now our text encodes to `00`) and so on.

Your assignment is to write a program to help decide whether to use monographs, digraphs or trigraphs to encode a given piece of text. The input will be up to 10^8 characters of ASCII text, and your program should output integers n_1 n_2 n_3, the length of the text encoded by monographs, digraphs and trigraphs respectively. Treat the text as case-insensitive, and ignore non-alphabet characters like whitespace, punctuation, etc. If the number of characters is not a multiple of 6, add up to 5 `z`s at the end so that it is.

Name your program `Codelength.java` for submission to Codegrade.

4 Written exercises

For each question, briefly justify your answer.

1. Suppose we have recursive algorithms whose running times are given by each of the following definitions (together with $T(1) = 1$ in each case). Calculate the aysmptotic runtime for each.
   (a) $T(n) = 5T(n/4) + n^3$.
   (b) $T(n) = 2T(n/8) + \sqrt[3]{n}$.
   (c) $T(n) = 6T(n/3) + n \log n$.
2. I am hoping to improve on Karatsuba's algorithm for integer multiplication, by spliting each n-bit integer into three parts rather than two. What is the largest number of multiplies (of $n/3$-bit integers) I can use in order to beat Karatsuba's algorithm (which uses 3 multiplies of $n/2$-bit integers)?
3. For which of the following does the Master Theorem apply? For those where it does apply, calculate asymptotic runtime. For the first case where it does not, apply iterated substitution twice to obtain $T(n)$ in terms of $T(n/4)$.
   (a) $T(n) = 4T(n/2) + n^2 \log \log n$.
   (b) $T(n) = 3T(n/2) + n^2 \log \log n$.
   (c) $T(n) = T(n/2) + n(2 - \sin n)$.
   (d) $T(n) = T(n/2) + n(2 - \sin n)$.