CMPE 300: Analysis of Algorithms Project 1

Mehmet Özer
mehmet.ozer1@bogazici.edu.tr
Asude Ebrar Kiziloglu
asude.kiziloglu@std.bogazici.edu.tr

Strict Deadline: Sunday 17 November 2024 23:59

1 Introduction

This project is designed to assess your understanding of the core concepts in algorithm analysis. You will be given the pseudocode of an algorithm. Your task is to thoroughly analyze this algorithm and determine its basic operation(s), calculate its best, worst, and average-case time complexities, implement it in code, execute it and compare the theoretical results with the actual execution times.

2 Details of the Algorithm

Consider the following algorithm which takes a list `arr`[0:n-1] as input. Each element in the list is 'c', 'm', 'p', or 'e'; i.e., `arr`$[i] \in \{$'c', 'm', 'p', 'e'$\}$, $0 \leq i \leq n-1$. For each i, the probability that `arr`$[i] =$ 'c' is $\frac{1}{8}$, the probability that `arr`$[i] =$ 'm' is $\frac{1}{4}$, the probability that `arr`$[i] =$ 'p' is $\frac{1}{8}$, and the probability that `arr`$[i] =$ 'e' is $\frac{1}{2}$, $0 \leq i \leq n-1$.

Algorithm 1 CmpE Algorithm

input: arr[0: n-1] (a list of size n)
output: res (an integer)

```
1: res ← 1
2: for i ← 0 to n − 1 do
3:     if arr[i] = 'c' then                                   ▷ (1)
4:         for j ← n down to 1 do
5:             k ← i + n
6:             for y ← 0 to n by y ← y + j do                 ▷ (2)
7:                 k ← k − 1                                  ▷ (4)
8:             res ← res + k                                  ▷ (3)
9:     else if arr[i] = 'm' then
10:        z ← 1
11:        while z < n do
12:            z ← 2 ∗ z                                      ▷ (4)
13:            res ← res + z                                  ▷ (3)
14:    else if arr[i] = 'p' then
15:        w ← n
16:        res ← res − 1
17:        while w > 0 do
18:            w ← ⌊w/5⌋                                      ▷ (4)
19:            res ← res + 1                                  ▷ (3)
20:    else if arr[i] = 'e' then
21:        for m ← 1 to i do                                  ▷ (2)
22:            p ← m
23:            for l ← m to n do
24:                for t ← n down to 1 do
25:                    p ← p + t                              ▷ (4)
26:                res ← res + p                              ▷ (3)
   return res
```

3 Theoretical Analysis

Consider each of the following four cases separately (one by one):

1. Basic operation is the comparison marked as (1)
2. Basic operations are the two loop increments marked as (2)
3. Basic operations are the four assignments marked as (3)
4. Basic operations are the four assignments marked as (4)

For each case:

1. Analyze B (n)
2. Analyze W (n)
3. Analyze A (n)

Your analyses must be exact, otherwise no points will be given. That is, for each analysis, first find the "exact" number of basic operations and then convert them to asymptotic notation. (Note that best-case input and worst-case input may be different for different cases.) So, you will obtain 12 analysis results.

4 Real Execution

Then, identify the "correct" basic operation(s) – i.e. the operation that characterizes this algorithm.

Code this algorithm in a programming language you wish. Execute it on a computer with 17 different input sizes n 1, 5, 10, 20, 30, 40, 50, 60, 70, 90, 100, 120, 130, 140, 150, 160, 170, for three types of inputs (best-case input, worst-case input, average-case input). For each, record the actual execution time (in milliseconds, seconds, etc.). At the end, you should have 51 different time records. Then, you will fill the table in the answer sheets according to the results you get. Also, your code should print out the results. For more information, see Section 6 Code.

As an average-case input, you can generate a random array arr[0:n-1] formed of 'c's, 'm's, 'p's, and 'e's according to the probability distribution explained at the beginning. Note that in order to observe the average behavior, for each input size, you must execute the algorithm with random inputs several times and take the average. For this project, run the algorithm at least 10 times for taking the average.

5 Comparison of Theoretical Analysis and Real Behavior

Compare the theoretical results and the actual execution times using graphs. Use graphs where the x-axis denotes the input size and the y-axis denotes the complexity (for the theoretical analysis) and time (for the actual execution). Your comparisons must include all the theoretical and actual analyses you derived above. For each of the 17 theoretical analyses, comment on the result of the analysis by comparing with the related actual execution time. No points will be given if any comment is missing or the comments are not satisfactory.

6 Code

You will also submit your code. You are free to select any language. (Python is preferred.) Yet, please provide a ReadMe file that simply explains how to run your code. Then you should print the time elapsed for best, worst and average cases, and for 17 different data sizes n as:

```
Case: <case> Size: <size> Elapsed Time (s): <time>
```

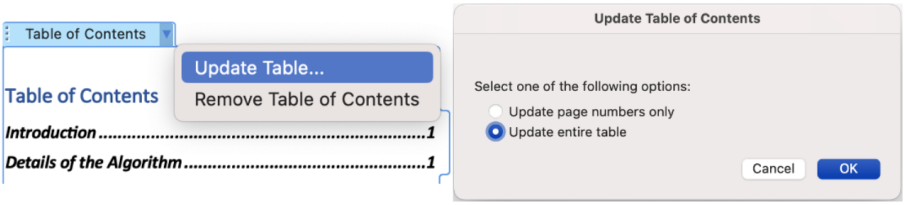
where,

- `<case>` denotes " `best`", " `worst`" or " `average`"
- `<size>` denotes the data size
- `<time>` denotes the time (in seconds)

For each average-case result, you should also print the execution times of all the runs from which you took the average.

7 Submission

- This project is designed to be completed by a pair of two students.
- The answers of all the questions must be collected into one of the answer sheets that is provided to you. You can use either the LATEX or Word answer sheet. Please follow the headings, and type your writings under the appropriate heading.
- If you decide to use the Word template, please update the table of contents after your report is ready (Click on the arrow on the top of the contents table, then select Update Entire Table).



- Prepare the answers using a word processor, not in handwritten.
- In addition to the project files, each student in a group must submit a document stating explicitly the parts of the project she/he has worked on. The document must begin with the line "I worked on the following parts in this project:" and all the parts the student has worked on must be explained very clearly and in not less than 10 lines. Do not write general comments such as "we worked on the project together", and the like. Write what you have done in the project explicitly. If both students write the same or similar explanations, their projects will not be graded.
- You should also submit the program code.
- Each student in the group should upload a single zip file on Moodle, which consists of the pdf file of the answers, a single file that is the program code and the document that explains the parts of the project you worked on. - Name the files as follows:
  - Answers: `StudentNumber.pdf`
  - Program code: `StudentNumber.xxx` (xxx: depending on the language)

  - Document: `StudentNumber_WorkDone.pdf`
  - Zip file that includes these three files: `StudentNumber.zip`

- Deadline is **Sunday, 17 November 2024, 23.59.** `Deadline is strict.`
- No late submission will be accepted.
- **Each group must answer the questions themselves, without any interactions with others. No materials from resources (internet, books, etc.) are allowed to be used.**