UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Analysis and Design of Algorithms

CS2102
Analysis of Algorithms Practice
2020-II

15

Submission deadline: 22 Oct, 20:05
Number of questions: 5

- Attach your answers in order to keep a single PDF file. Locate your images and source code in the proper directory according to the problem number.
- Read the questions carefully and write your answers clearly. Answers that are not legible will not have any score.
- Notes are allowed. To compile this file you can use the command latexmk -pdf -shell-escape main.tex
- Consider edge cases properly, any file that doesn't compile or doesn't met the requirements will not have any grade (clang++ or g++ are preferred).

Outcomes:

a. Apply appropriate mathematical and related knowledge to computer science.

b. Analyze problems and identify the appropriate computational requirements for its solution.

Problem 1 (Outcomes a) - 5 points

Consider the amortized analysis of the dynamic array problem.

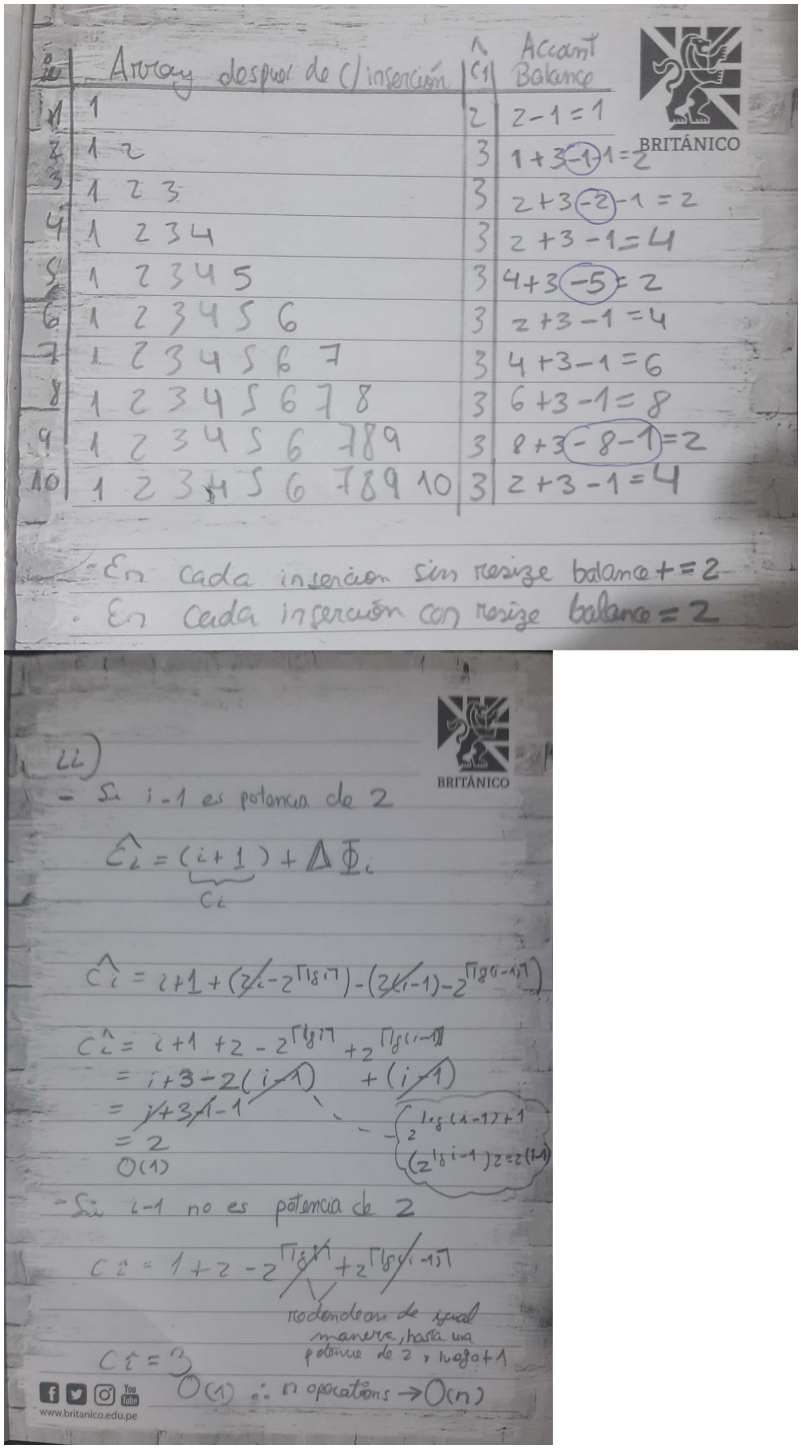
(i) (1pt) Apply the accounting method to simulate the bank account balance over 10 operations considering that:

$$\hat{c}_i = \begin{cases} \$2.00, & i = 1. \\ \$3.00, & \text{otherwise.} \end{cases}$$

(ii) (4pts) Use the potential function $\Phi_i = 2i - 2^{\lceil log(i) \rceil}$ to calculate the amortized operation cost $\hat{c}_i$ assuming that:

$$c_i = \begin{cases} i, & \text{if } i-1 \text{ is power of 2.} \\ 1, & \text{otherwise.} \end{cases}$$

Remember to explain with detail your procedure and that $\hat{c}_i = c_i + \Delta\Phi_i$



Problem 2 (Outcomes a) - 5 points

Evaluate the following expressions using Master Method. If the Master Method can't be applied explain the reason.

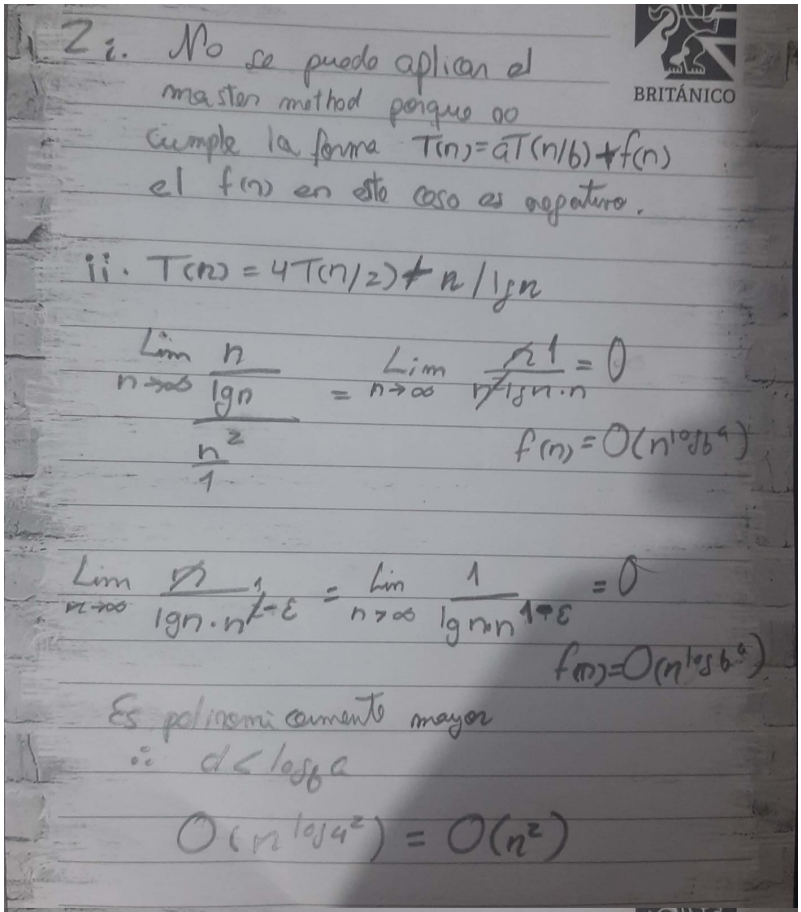
(i) $T(n) = 20T(n/6) - n^2 log(n)$

(ii) $T(n) = 4T(n/2) + \frac{n}{log(n)}$

(iii) $T(n) = 6T(n/3) + n^2 log(n)$

(iv) $T(n) = \sqrt{2}T(n/2) + log(n)$

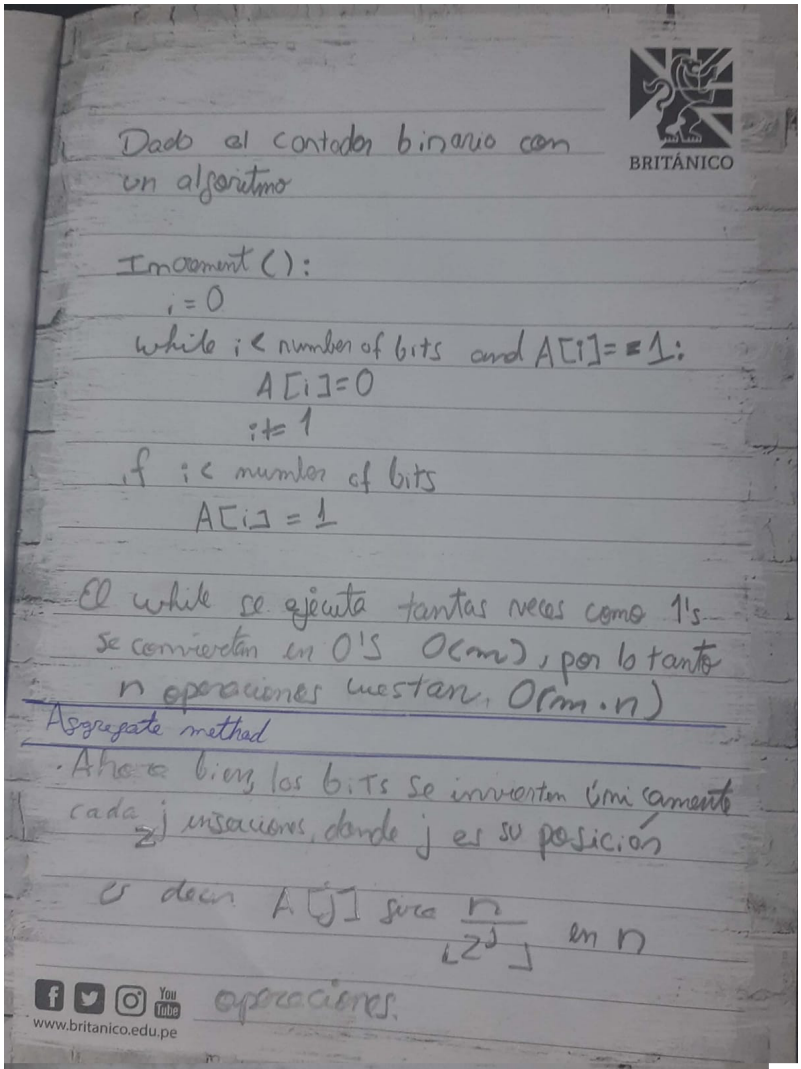
(v) $T(n) = 2T(n/2) + nlog(n)$

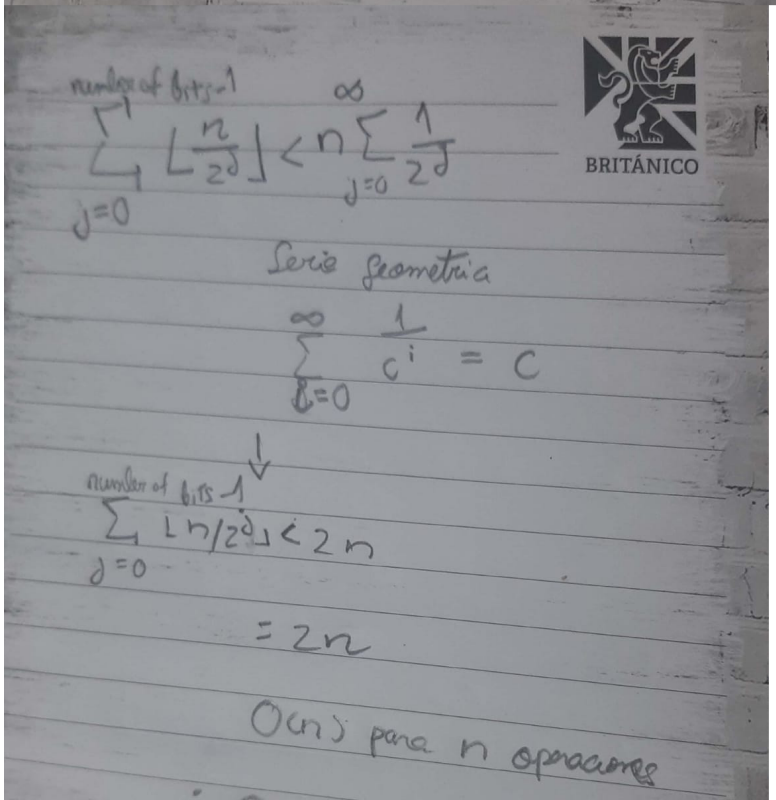


2 i. No se puede aplicar el master method porque no cumple la forma T(n)=aT(n/b)+f(n) el f(n) en este caso es negativo.
ii. T(n) = 4T(n/2) + n/lgn
Lim n / lgn / n²/1 = Lim 1 / lgn·n = 0
f(n) = O(n^log_b a)
Lim 1 / lgn·n^1-ε = Lim 1 / lgn·n^1+ε = 0
f(n) = O(n^log_b a)
Es polinomialmente mayor
∴ d < log_b a
O(n^log4 2) = O(n²)

Problem 3 (Outcomes a) - 4 points

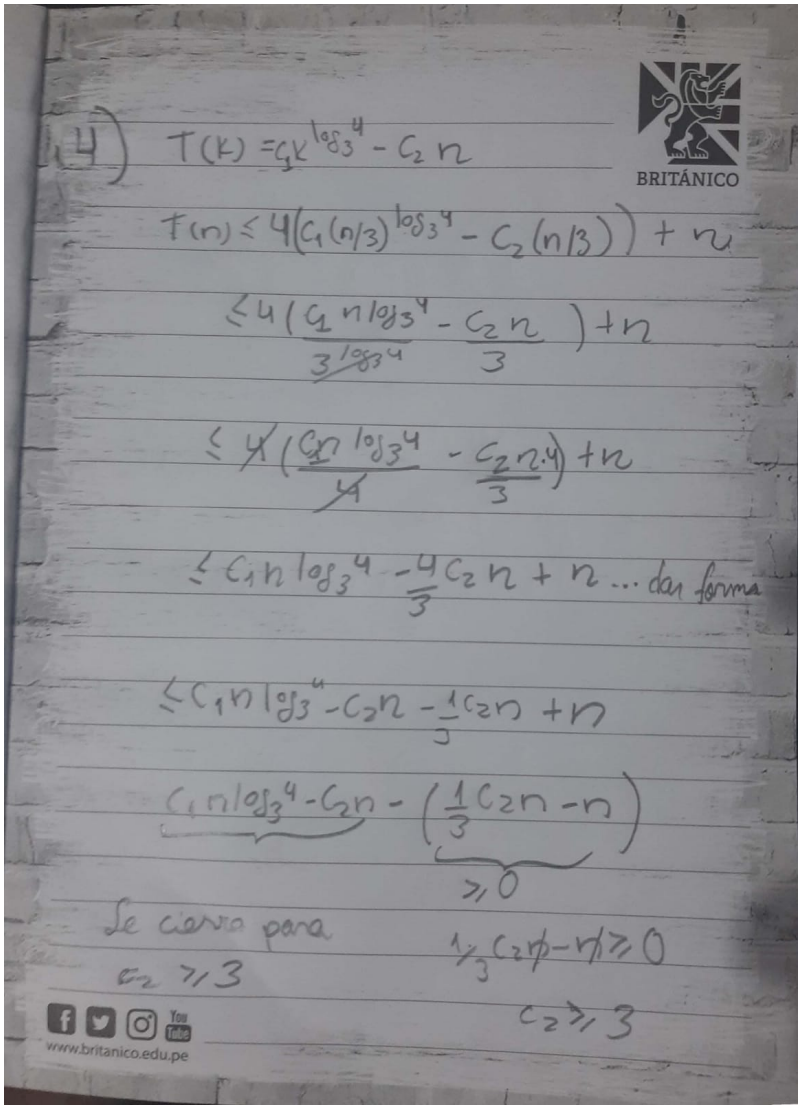
Consider a binary counter algorithm that counts from 0 to n using binary representation. Find an upper bound on the total worst-case running time (1pt), then calculate the amortized cost of each operation using one of the methods presented in class (3pts). Explain your procedure.


BRITÁNICO
Dado el contador binario con un algoritmo
Increment():
i = 0
while i < number of bits and A[i] == 1:
A[i] = 0
i += 1
if i < number of bits
A[i] = 1
El while se ejecuta tantas veces como 1's se conviertan en 0's O(m), por lo tanto n operaciones cuestan, O(m.n)
Aggregate method
Ahora bien, los bits se invierten únicamente cada 2^j iteraciones, donde j es su posición
O dea A[j] sera $\lfloor \frac{n}{2^j} \rfloor$ en n operaciones.
www.britanico.edu.pe


BRITÁNICO
$\sum_{j=0}^{\text{number of bits}-1} \lfloor \frac{n}{2^j} \rfloor < n \sum_{j=0}^{\infty} \frac{1}{2^j}$
Serie geometria
$\sum_{i=0}^{\infty} \frac{1}{c^i} = C$
$\sum_{j=0}^{\text{number of bits}-1} \lfloor n/2^j \rfloor < 2n$
$= 2n$
O(n) para n operaciones

Problem 4 (Outcomes a) - 2 points

Given $T(n) = 4T(n/3) + n$ prove using substitution method that the solution for this recurrence is $O(n^{log_3 4})$

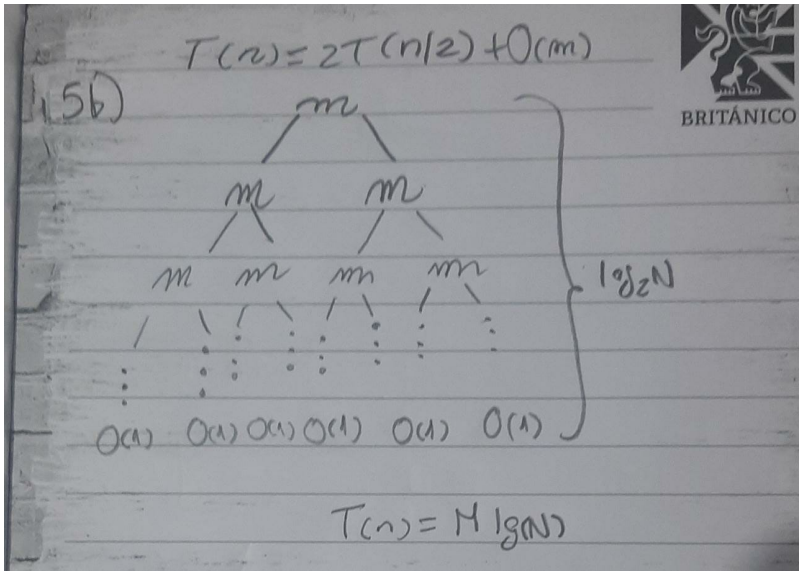


Problem 5 (Outcomes a) - 4 points

Write a divide-and-conquer c++ algorithm (3pts) that receives and array of n strings and returns the largest prefix of all strings in the array. For example if the array is {"dog", "doggie", "dogs"} then the largest prefix is "dog". If there isn't a common prefix between all words return an empty string. After that, write the execution time of your algorithm (1pt)

```
#include <iostream>
```

```cpp
#include <vector>

using namespace std;

string getCommonPrefix(string& word1, string &word2){
    auto size1 = word1.size(), size2 = word2.size();
    int i,j;
    string toBeReturned;
    for(i=0, j=0; i<size1 and j<size2; ++i,++j){
        if (word1[i] != word2[j])
            break;
        toBeReturned.push_back(word1[i]);
    }
    return toBeReturned;
}

string commonPrefix(vector<string>& words, int min, int max){
    if (min == max)
        return words[max];
    else{
        int mid = (max - min)/2;
        string dividedProblem1=commonPrefix(words, min, mid);
        string dividedProblem2=commonPrefix(words, mid+1, max);
        return getCommonPrefix(dividedProblem1, dividedProblem2);
    }
}

string problem5(vector<string>& words) {
    return commonPrefix(words,0,(int)words.size()-1);
}

int main() {
    std::cout << "Problem 5 - Prefix\n";
    std::vector<std::string> words = {"prefixHOLA", "prefixChau",
    ↪  "prefixPALABRA"};
    std::cout << problem5(words) << "\n";

    ///T(n) = 2T(n/2) + O(m), where m is length of the words
}
```


T(n) = 2T(n/2) + O(m)
1.5b)
BRITÁNICO
m
m m
m m m m
log2N
O(1) O(1) O(1) O(1) O(1) O(1)
T(n) = M lg(N)