COMP 250, Winter 2017 - Homework 2

Mathieu Blanchette

February 18, 2017

This homework is due in whole on February 15th 2017, 23:59 Submit your solution to questions 1-8 as a PDF file. Submit your solution to question 9

Question 1 (20 points)

The algorithm below, called bubbleSort, sorts in increasing order the elements of an array.

```
Algorithm bubbleSort(A,n)
Input: An array $A[0...n-1]$ of $n$ elements
Output: The array $A$ is sorted
for $i \leftarrow 1$ to $n-1$ do
.        for $j \leftarrow 0$ to $n-1-i$ do
.                if ($A[j+1] < A[j]$) then /* swap A[j+1] and A[j] */
.                        $tmp \leftarrow A[j]$
.                        $A[j] \leftarrow A[j+1]$
.                        $A[j+1] \leftarrow tmp$
```

a) (5 points) What kind of input will yield the worst possible running time, for any fixed size n? Why?

Solution: If the array A is sorted in decreasing order, the first element will "bubble-up" to position n-1. After that, the new first element of A, which used to be the second one, will also bubble-up to position $n-2$, etc.

b) (10 points) Let $T(n)$ be the number of primitive operations performed in this worst case. Write, for each line of the pseudocode, the number of primitive operations executed and the type (assignment, condition, arithmetic...) of each. Sum up everything to obtain $T(n)$, assuming all primitive operations take the same time. If $T(n)$ contains summations, replace them by their explicit value. Show your work step by step like we did in class.

Solution: We count the number of primitive operations performed on each line of the code.
Initialization of for loop: 1
Repeat n times

```
... Check for loop condition: 3
... Initialization of second for loop: 1
... Repeat n-i times
...... Check for loop condition: 4
...... If (A[j+1] < A[j] ) then : 5
...... tmp ← A[j] : 2
...... A[j] ← A[j+1]: 4
...... A[j+1] ← tmp: 3
...... increment j: 2
... increment i: 2
```
Conclusion:

$$\begin{aligned} T(n) &= 1+\sum_{i=0}^{n-1}(4+(\sum_{j=0}^{n-1-i}4+5+2+4+3+2)+2) \\ &= 1+\sum_{i=0}^{n-1}6+(n-i)20 \\ &= 1+6n+20n^2-20\sum_{i=0}^{n-1}i \\ &= 1+6n+20n^2-20n(n-1)/2 \\ &= 1+16n+10n^2 \end{aligned}$$

c) (5 points) Give the simplest and most accurate big-Oh representation for $T(n)$.
Solution: $T(n) \in O(n^2)$.

Question 2. Running time analysis (8 points)

For each of the algorithms below, indicate the running time using the simplest and most accurate big-Oh notation. Assume that all arithmetic operations can be done in constant time. The first algorithm is an example. No justifications are needed.

Algorithm	**Running time in big-Oh notation**
Algorithm Example(n)	
$x \leftarrow 0$	$O(n)$
for $i \leftarrow 1$ **to** n **do**	
. $\quad x \leftarrow x + 1$	
Algorithm algo1(n)	
$i \leftarrow 1$	$O(n)$
while $i < n$ **do**	
. $\quad i \leftarrow i + 100$	
Algorithm algo2(n)	
$x \leftarrow 0$	
for $i \leftarrow 1$ **to** n **do**	$O(n^2)$
. $\quad$ **for** $j \leftarrow 1$ **to** i **do**	
. $\qquad x \leftarrow x + 1$	
Algorithm algo3(n)	
$i \leftarrow n$	$O(\log(n))$
while $(i > 1)$ **do**	
. $\quad i \leftarrow i/2$	
Algorithm algo4(n)	
$k \leftarrow 1$	
for $i \leftarrow 1$ **to** 1000	$O(1)$
. $\quad$ **for** $j \leftarrow 1$ to i	
. $\qquad k \leftarrow (k + i - j) * (2 + i + j)$	

Question 3 (10 points)

Prove by induction that $4^n < n!$ for any $n \geq 9$.
Note: $n! = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 2 \cdot 1$.

Solution: Let $P(n) : 4^n < n!$. We prove that $P(n)$ is true for all $n \geq 9$, using induction.
1) Base case. For $n = 9$, we have $4^n = 4^9 = 262144 < 362880 = 9! = n!$.
2) Induction step. The induction hypothesis is $P(k) : 4^k < k!$. We then show that for any $k \geq 9$, $P(k)$ implies $P(k+1) : 4^{k+1} < (k+1)!$.

$$\begin{aligned} 4^{k+1} &= 4 \cdot 4^k \\ &< 4 \cdot k! \ (\textit{by I.H.}) \\ &< (k+1) \cdot k! \ (\textit{since } k \geq 9) \\ &= (k+1)! \end{aligned}$$

Thus, by (1) and (2), $P(n)$ is true for all $n \geq 9$.

Question 4 (10 points)

Let

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3 \cdot T(n-1) + 2 & \text{if } n > 1 \end{cases}$$

Obtain an explicit formula for $T(n)$.

Solution: First, let us first obtain the first few values of $T(n)$, for future verifications. We have $T(1) = 1$, $T(2) = 3*T(1)+2 = 5$, $T(3) = 3*T(2)+2 = 17$, $T(4) = 3*T(3)+2 = 53$.

We use the substitution method to obtain an explicit formula for T.

$$\begin{aligned} T(n) &= 3T(n-1)+2 && (1) \\ &= 3(3T(n-2)+2)+2 = 9T(n-2)+2*3+2 && (2) \\ &= 9(3T(n-3)+2)+2*3+2 = 27T(n-3)+2*9+2*3+2 && (3) \\ &= 27(3T(n-4)+2)+2*9+2*3+2 = 81T(n-5)+2*27+2*9+2*3+2 && (4) \end{aligned}$$

We can now guess that the expression after k rounds of substitutions will be:

$$T(n) = 3^k T(n-k) + \sum_{i=0}^{k-1} 2 * 3^i$$

We reach the base case when $n-k=1$, i.e. when $k = n-1$. Replacing k by $n-1$ in the equation above, we get:

$$\begin{aligned} T(n) &= 3^{n-1}T(1) + \sum_{i=0}^{(n-1)-1} 2 * 3^i \\ &= 3^{n-1}1 + 2\sum_{i=0}^{n-2} 3^i \\ &= 3^{n-1} + 2\frac{3^{(n-2)+1}-1}{3-1} \\ &= 3^{n-1} + 3^{n-1} - 1 \\ &= 2 * 3^{n-1} - 1 \end{aligned}$$

Conclusion: $T(n) = 2 * 3^{n-1} - 1$.

We can now check that the values given by our explicit formula match those calculated by the recurrence for small values of n. This is not a proof that we got the correct formula, but for sure if they do not match, then we must have made a mistake somewhere. From the explicit formula, we get $T(1) = 2*3^0 - 1 = 1$, $T(2) = 2*3^1 - 1 = 5$, $T(3) = 2*3^2 - 1 = 17$, $T(4) = 2*3^3 - 1 = 53$. All those values matched those obtained from the recurrence, so it all looks good.

Question 5 (8 points)

Prove, using only the definition of $O()$, that $25n + 5$ is $O(n)$. Write your proof using proper mathematical formalism, as done in the lecture notes

Solution:

We need to find a $c \in \mathbf{R}$ and an $n_0 \in \mathbf{N}$ such that if $n \geq n_0$, then $25n + 5 \leq cn$. We have:

$$\begin{aligned} 25n + 5 &\leq 25n + 5n \text{ if } n \geq 1 \\ &= 30n \\ &= cn \text{ for } c = 30 \end{aligned}$$

Thus, if $c = 30$ and $n_0 = 1$, $25n + 5 \leq cn$ for all values of $n \geq n_0$. Conclusion: $25n + 5$ is $O(n)$.

Question 6 (8 points)

Prove using only the definition of $O()$ that $(n+10)^{2.5} + n^2 + 1$ is $O(n^{2.5})$. Write your proof using proper mathematical formalism, as done in the lecture notes.

Solution:

We need to find a $c \in \mathbf{R}$ and an $n_0 \in \mathbf{N}$ such that if $n \geq n_0$, then $(n + 10)^{2.5} + n^2 + 1 \leq cn^{2.5}$. We have:

$$\begin{aligned} (n + 10)^{2.5} + n^2 + 1 &\leq (n + n)^{2.5} + n^2 + 1 \text{ if } n \geq 10 \\ &\leq (2n)^{2.5} + n^{2.5} + n^{2.5} \text{ if } n \geq 1 \\ &= (2 + 2^{2.5})n^{2.5} \\ &= cn^{2.5} \text{ for } c = 2 + 2^{2.5} \end{aligned}$$

Thus, if $c = 2 + 2^{2.5}$ and $n_0 = 10$, $(n + 10)^{2.5} + n^2 + 1 \leq cn^{2.5}$ for all values of $n \geq n_0$. Conclusion: $(n + 10)^{2.5} + n^2 + 1$ is $O(n^{2.5})$.

Question 7 (8 points)

Prove, using only the definition of O(), that $(n + 1)^2$ is not $O(n)$. Write your proof using proper mathematical formalism, as done in the lecture notes.

Solution:

We must show that no matter what value of c and n_0 one chooses, there is value of $n \geq n_0$ such that $(n + 1)^2 > cn$. This can be done in many ways. One way is to observe that if we choose $n = c$, then $(n+1)^2 = (n+1)(n+1) > nn = cn$. Thus, choosing $n = max(n_0, c)$ will work.

Another way would be to use the quadratic equation formula to solve for n in $(n + 1)^2 = cn$, which we would rewrite as $n^2 + (2 - c)n + 1 = 0$. We get that $n = \frac{-(2-c) \pm \sqrt{(2-c)^2 - 4*1*1}}{2}$, which we simplify to $n = \frac{(c-2) \pm \sqrt{c^2-4c}}{2}$. Thus, choosing $n = max(n_0, \frac{(c-2)+\sqrt{c^2-4c}}{2}$ will work.

Question 8 (8 points)

There exists a remote island with a certain number of inhabitants, 10 of whom have a blue face. For reasons that are easily understandable, having a blue face is a topic that is taboo on the island. There are no mirrors on the island and nobody is allowed to tell another person that this person has a blue face. In fact, their law specifies that if a person was to learn that he/she has a blue face, he/she would have to commit suicide at midnight that day. Since the topic is taboo, there is no way for a person to realize that he/she have a blue face, so everybody lives happily. Until...

A foreigner comes to visit the island. Before leaving the island, he announces to everybody: "There are some people on this island who have a blue face! You should all start thinking about it now." Ten days later (that is, on the the tenth midnight after this declaration), all ten blue-faced inhabitants commit suicide.

Question: How did they realize that they had a blue face? Give your answer as concisely as possible. You shouldn't need more than 10 lines to write your answer. Hint: All inhabitants of the island took CS250 and got an 'A' in it.

Solution:
Consider what would happen if there was a single inhabitant with a blue face on the island. After the foreigner makes his announcement, the blue face person knows that they must be the one with a blue face because they see no one else with a blue face. Thus, on the first midnight after the announcement, they would commit suicide.

Now, what if there were two people with blue faces? On the first day, they don't do anything because they see the other blue face person and think that this other person may be the only one with a blue face. Both blue faced people actually think the same way, and so neither commit suicide on the first midnight. Now, on the second day, they see each other and realize that the other didn't commit suicide. How could that be? It has to be because the other sees another person with a blue face. But each person realizes that this other blue-faced person must be them, because they doesn't see anybody else with a blue face. Thus, at midnight of the second day, the two blue faced people will commit suicide.

What if there were three? On the first two days, nothing will happen. However, on the third, if I have a blue face and see that the two other blue faced people I see have not committed suicide yet, I conclude that it has to be because they another person with a blue face, which has to be me. So, on the third day, all three blue faced people commit suicide.

In general, if there are n people with blue faces, nothing will happen on the first $n-1$ days, but then seeing that nothing happend during those days, each blue faced person will conclude that they have a blue face and commit suicide that midnight.

Question 9 (20 points)

In this question, you are asked to implement in Java a program that will help you cheat next time you play Scrabble. Use the code available at

`http://www.cs.mcgill.ca/~blanchem/250/hw2/Scrabble.java`

and the accompanying file

`http://www.cs.mcgill.ca/~blanchem/250/hw2/englishDictionary.txt`

In Scrabble, you are given a set of n characters (possibly with some letters present more than once) and have to assemble English words out of these letters. You want a program that will find and return the longest English word that can be made out the given letters (without using any of the n letters more than once, but allowing to use certain letters zero time). For example, given the letters aajpv, your method should return the String java. Given the letters bpocuretm, it should return computer. If there are more than one valid words of maximal length, then your method should return any one of them.

Hint: This can be done surprisingly simply and elegantly using a recursive algorithm (my code is 12 lines long). Here is how. We will generate every combination of letters and check if it is in the dictionary using the "contains" method of the myDictionary object (see example in code).

Your method will have the following structure:
public static String longestWord(char avail[], String prefix)

The method will return the longest valid English word that can be obtained by starting with the word prefix and adding any subset of the letters in array avail. For example,
longestWord(['a','e','r','t'], "compu") should return "computer".
longestWord(['x','z'], "compu") should return "".
longestWord(['b','p','o','c','u','r','e','t','m'], "") should return "computer"

The method will enumerate all words that start with the string prefix in order to find the longest valid extension. This will be done recursively, using the following principle: The longest valid extension of the word prefix is either

1. The empty string ""
2. prefix itself, if prefix is in the dictionary
3. the longest word that can be made starting with the word prefix + avail[0], completed with the letters avail[1],...,avail[avail.length-1]
4. the longest word that can be made starting with the word prefix + avail[1], completed with the letters avail[0],avail[2],...,avail[avail.length-1] ...

5. the longest word that can be made starting with the word prefix + avail[avail.length-1], completed with the letters avail[0],avail[1],...,avail[avail.length-2]

Solution:

```
 public static String longestWord(char availableLetters[], String prefix) {

        String longest="";
        if (myDictionary.contains(prefix)) longest=prefix;

        for (int i=0;i<availableLetters.length;i++) {
            // considering adding letter availableLetters[i] to prefix

            // remove letter i from available letters
            char newAvailableLetters[] = new char[availableLetters.length-1];
            for (int j=0;j<i;j++) newAvailableLetters[j] = availableLetters[j];
            for (int j=i+1;j<availableLetters.length;j++) newAvailableLetters[j-1]=
                                                                availableLetters[j];

            String extension = longestWord(newAvailableLetters,
                                                 prefix + availableLetters[i] );

            if ( extension.length() > longest.length() ) longest = extension;
        }
        return longest;
}
```