CS300 – Spring 2022-2023 - Sabancı University

Homework #2: Algorithm Complexity

Due April 11, Tuesday, 22:00

Question1

Given the following code:

```
void printStarsHelper()
{
    cout << "***" << endl;
}

/* Assume n=3^k for some k*/
void unnecessaryStars(int n)
{
     if (n == 1) {
         cout << "*" << endl;
     }
     else if (n % 2 == 0) {
         printStarsHelper();
         unnecessaryStars(n / 3);
         unnecessaryStars(n / 3);
         unnecessaryStars(n / 3);
      }
      else if (n % 2 == 1) {
          unnecessaryStars(n / 3);
          unnecessaryStars(n / 3);
          unnecessaryStars(n / 3);
          printStarsHelper();
      }
 }
```

1. Write a recurrence for the number of stars printed by the **`unneccesaryStars`** function.

   The last two else if statements are not important because in each recursive iteration, only one of them will be reached (Consider only one of them).

   The first if statement is the base step for the recurrence.

2. Solve the recurrence exactly by successive expansions and express it as O(f(n)).

Question2

Given the following code:

```
void printZerosHelper(int n)
{
    for (int i = 1; i <= n; i *= 2) {
        cout << "0" << endl;
    }
}

/* Assume n=2^k for some k */
void unnecessaryZeros(int n)
{
    if (n == 0) {
        cout << "0" << endl;
    }
    else if (n % 2 == 0) {
        printZerosHelper(n);
        unnecessaryZeros(n - 2);
        unnecessaryZeros(n - 2);
    }
    else if (n % 2 == 1) {
        printZerosHelper(n);
        unnecessaryZeros(n - 2);
        unnecessaryZeros(n - 2);
    }
}
```

1. Write a recurrence for the number of zeros printed by the **unneccesaryZeros** function.

   The last two else if statements are not important because in each recursive iteration, only one of them will be reached (Consider only one of them).

   The first if statement is the base step for the recurrence.
   unneccessaryZerosHelper iterates log N times. (We have seen this in the lectures/recitations)

2. Solve the recurrence exactly by successive expansions and express it as O(f(n)).