Informatics II, Spring 2024, Exercise 3

Publication of exercise: March 4, 2024
Publication of solution: March 11, 2024
Exercise classes: March 11 - March 15, 2024

Learning Goal

- Understand the efficiency of algorithms like binary search.
- Learn how to analyze algorithmic complexity and asymptotic complexity of algorithms.
- Learn how to analyze special case and correctness of algorithms.

Task 1: Linear Search and Binary Search [Easy]

Consider an array A with n distinct integers that are sorted in an ascending order and an integer t.

a) The C funtion `linear_search` traverses the integers in A, one after another, from the beginning. If t is found in A `linear_search` returns 1, otherwise 0. Complete the C funtion `linear_search`(int A[], int n, int t) in task1.c file.

b) The C funtion `binary_search`(int A[], int n, int t) that employs binary search to find integer t in A. Reference the following pseudocode for binary search to implement the `binary_search` function. If t is found in A `binary_search` returns 1, otherwise 0. Complete the C funtion `binary_search`(int A[], int n, int t) in task1.c file.

Algo: BinSearch1(A,v)

Input: sequence $A[1..n]$ of length n, value v
Output: either index i such that $v == A[i]$ or `NIL`

```
l = 1; r = n;
m = ⌊(l+r)/2⌋;
while l≤r ∧ v≠A[m] do
    if v<A[m] then r = m-1 else l = m+1;
    m = ⌊(l+r)/2⌋;
if l≤r then return m else return NIL;
```

c) What are the asymptotic complexity for the C funtions `linear_search` and `binary_search`.

d) Compile task1.c file. Run your codes with the following parameters for n and t:

- n = 1000000, t = 1000000
- n = 10000000, t = 10000000
- n = 100000000, t = 100000000.

Report the run time growth for `linear_search` and `binary_search`, respectively.

Task 2: Algorithmic Complexity [Easy]

Below is a pseudocode of a function named `whatDoesItDo`, which takes an array `A[1..n]` of n integers and an integer k as inputs.

```
Algo: whatDoesItDo(A, n, k)
result = -1000
for i = 1 to n do
    current = 0
    for j = i to n by k do
        current = current + A[j]
    if current > result then
        result = current
return result
```

Note: In the above pseudocode, `for` $j = i$ `to` n `by` k means we do not increase j by 1, but each time, we increase it by k, i.e., `j=j+k`.

a) Perform exact analysis of the running time of the algorithm.

b) Determine the asymptotic complexity of the algorithm?

Task 3: Asymptotic Complexity [Easy]

a) Calculate the asymptotic tight bound for the following functions and rank them by their order of growth (lowest first). Clearly work out the calculation step by step in your solution.

$$f_1(n) = (2n+3)!$$
$$f_2(n) = 2\log(6^{\log n^2}) + \log(\pi n^2) + n^3$$
$$f_3(n) = 4^{log_2 n}$$
$$f_4(n) = 12\sqrt{n} + 10^{223} + \log 5^n$$
$$f_5(n) = 10^{\lg 20} n^4 + 8^{229} n^3 + 20^{231} n^2 + 128 n \log n$$
$$f_6(n) = \log n^{2n+1}$$
$$f_7(n) = \log^2(n) + 50\sqrt{n} + \log(n)$$
$$f_8(n) = 14400$$

b) Assume $f_1(n) = O(1)$, $f_2(n) = O(N^2)$, and $f_3(n) = O(N \log N)$. From these complexities it follows that $f_1(n) + f_2(n) + f_3(n) = O(N \log N)$.

Answer: ☐ True ☐ False

Task 4: Special Case and Correctness Analysis [Medium]

Consider the algorithm `algo1`. The input parameters are an array `A[1..n]` with n distinct integers and $k \leq n$.

```
Algo: algo1(A, n, k)
----------------------------------
sum = 0;
for i = 1 to k do
    maxi = i;
    for j = i to n do
        if A[j] > A[maxi] then
            maxi = j;
    sum = sum + A[maxi];
    swp = A[i];
    A[i] = A[maxi];
    A[maxi] = swp;
return sum
```

a) Specify the pre/post conditions of the `algo1` algorithm.

b) For the two `for` loops in the algorithm:

   i. Determine if the loop is `up` loop or `down` loop.

   ii. Determine the invariants of these two loops and verify whether they are hold in three stages: `initialization`, `maintenance` and `termination`.

c) Identify some edges cases of the algorithm and verify if the algorithm has the correct output.

d) Conduct an exact analysis of the running time of algorithm `algo1`.

e) Determine the best and the worst case of the algorithm. What is the running time and asymptotic complexity in each case?