University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatics II
Exercise 3

Mar 09, 2020

Algorithmic Complexity and Correctness

Task 1. Algorithm `whatDoesItDo(A,k)` gets an array `A[1...n]` of n integers as an input.

```
Algo: WHATDOESITDO(A,k)
sum = 0;
for i = 1 to k do
    mini = i;
    for j = i + 1 to length(A) do
        if A[j] < A[mini] then
            mini = j;
    sum = sum + A[mini];
    swp = A[i];
    A[i] = A[mini];
    A[mini] = swp;
return sum
```

a) Implement the algorithm as a C program that reads the elements of A and prints the result.

Given an array a and integer k, run the nested loop where outer loop runs from $i = 0$ to k and inner loop starts from $i + 1$ to array size. Inner loop find the index of i^{th} lowest element. After exiting from inner loop, i^{th} lowest element is added to sum variable and then this i^{th} lowest value is swaped with the element present in i^{th} index so that the same lowest element is not found again during the next iteration of outer loop.

```c
int sumKLowest(int A[], int k, int n) {
  int i, j, swp;
  int mini;
  int sum = 0;
  for(i = 0; i < k; i++) {
    mini = i;
    for(j = i + 1; j < n; j++) {
      if(A[j] < A[mini]) mini = j;
    }
```

```
    sum += A[mini];

    swp = A[i];
    A[i] = A[mini];
    A[mini] = swp;
  }
  return sum;
}
```

b) Describe what the algorithm does.

This algorithm takes as an input an unsorted array $A[1..n]$ of integers and calculates the sum of its k-lowest integers. For example, given the input array is $A = [12, 4, 10, 2, 8]$, if $k = 3$, the result is $2 + 4 + 8 = 14$ whereas, if $k = 4$, the result is $2 + 4 + 8 + 10 = 24$

c) Do an exact analysis of the running time of the algorithm.

Instruction	# of times executed	Cost
$sum := 0$	1	c_1
for $i := 1$ **to** k **do**	$k+1$	c_2
$mini := i$	k	c_3
for $j := i+1$ **to** n **do**	$\left(kn - \frac{k(k+1)}{2}\right)^* + 2k^{**}$	c_4
if $A[j] < A[mini]$ **then**	$kn - \frac{k(k+1)}{2}$	c_5
$mini := j$	$\alpha\left(kn - \frac{k(k+1)}{2}\right)^{***}$	c_6
$sum := sum + A[mini]$	k	c_7
$swp := A[i]$	k	c_8
$A[i] := A[mini]$	k	c_9
$A[mini] := swp$	k	c_{10}
return sum	1	c_{11}

$$* \ (n-2+1) + (n-3+1) + \ldots + (n-k) = \sum_{q=1}^{k}(n-q) = kn - \frac{k(k+1)}{2}$$

** k times for $i+1$ and k times for termination condition

*** $0 \leq \alpha \leq 1$

$$T(n) = c_1 + c_2(k+2) + c_3k + c_4(kn - \tfrac{k(k+1)}{2} + 2k) + c_5(kn - \tfrac{k(k+1)}{2}) + \\ + c_6(\alpha(kn - \tfrac{k(k+1)}{2})) + (c_7 + c_8 + c_9 + c_{10})k + c_1 1$$

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

Best case

$\alpha = 0, k = 1,$

$$T_{\text{best}}(n) = c_1 + 2c_2 + c_3 + c_4(n+1) + c_5(n-1) + 0 + c_7 + c_8 + c_9 + c_{10} + c_{11}$$

Worst case

$\alpha = 1, k = n,$

$$T_{\text{worst}}(n) = c_1 + c_2(n+1) + c_3n + c_4(\tfrac{n^2}{2} + \tfrac{3}{2}n) + c_5(\tfrac{n^2}{2} - \tfrac{n}{2}) + c_6(\tfrac{n^2}{2} - \tfrac{n}{2}) +$$

$(c_7 + c_8 + c_9 + c_{10})n + c_{11}$

Asymptotic complexity of best and worst case $T_{\text{best}}(n) = \Theta(n)$ $T_{\text{worst}}(n) = \Theta(n^2)$

e) What influence has the parameter k in the asymptotic complexity?
The parameter k has no direct influence on asymptotic complexity because it is fixed for the two cases and defines best and worst case.

Asymptotic Complexity

Task 2. Calculate the asymptotic tight bound for the following functions and rank them by their order of growth (lowest first). Clearly work out the calculation steps in your solution.

$$
\begin{aligned}
f_1(n) &= n^n + 2^{2n} + 13^{124} \\
f_2(n) &= \log\left(14(n-1)n^{3n+2}\right) \\
f_3(n) &= 4^{log_2 n} \\
f_4(n) &= 12\sqrt{n} + 10^{223} + \log 5^n \\
f_5(n) &= n^2 \log(n+1) + n \log n^2 + 0.5n \\
f_6(n) &= 7n^4 + 100n \log n + \sqrt{32} + n \\
f_7(n) &= \log(\min(n, \sqrt{n})) \\
f_8(n) &= \log^2(n) + 50\sqrt{n} + \log(n) \\
f_9(n) &= (n+3)! \\
f_{10}(n) &= 2\log(6^{\log n^2}) + \log(\pi n^2) + n^3
\end{aligned}
$$

- $f_1(n) = n^n + 2^{2n} + 13^{124} = n^n + 4^n + 13^{124} \in \Theta(n^n)$
- $f_2(n) = \log\left(14(n-1)n^{3n+2}\right) = \log(14) + \log(n-1) + (3n+2)\log n \in \Theta(n \log n)$
- $f_3(n) = 4^{log_2 n} = (2^2)^{log_2 n} = (2^{log_2 n})^2 = n^2 \in \Theta(n^2)$
- $f_4(n) = 12\sqrt{n} + 10^{223} + \log 5^n = 12\sqrt{n} + 10^{223} + n \log 5 \in \Theta(n)$
- $f_5(n) = n^2 \log(n+1) + n \log n^2 + 0.5n \in \Theta(n^2 \log n)$
- $f_6(n) = 7n^4 + 100n \log n + \sqrt{32} + n \in \Theta(n^4)$
- $f_7(n) = \log(\min(n, \sqrt{n})) = \log \sqrt{n} = \frac{1}{2} \log n \in \Theta(\log n)$
- $f_8(n) = \log^2(n) + 50\sqrt{n} + \log(n) \in \Theta(\sqrt{n})$
- $f_9(n) = (n+3)! \in \Theta((n+3)!)$
- $f_{10}(n) = 2\log(6^{\log n^2}) + \log(\pi n^2) + n^3 = 2 \log n^2 \log 6 + \log \pi + \log n^2 + n^3 = 4 \log 6 \log n + \log \pi + 2 \log n + n^3 \in \Theta(n^3)$

$f_7 < f_8 < f_4 < f_2 < f_3 < f_5 < f_{10} < f_6 < f_9 < f_1$

University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Special Case Analysis

Task 3. Given two strings A and B, develop an algorithm that checks if B is a substring of A and, if so, returns the number of occurrences of B in A.

a) Specify all the special cases that need to be considered and provide examples of the input data for each of them.

#	Case	A	B	result
1	A is empty	[]	mis	0
2	B is empty	understanding	[]	0
3	A is `NULL`	`NULL`	mis	0
4	B is `NULL`	understanding	`NULL`	0
5	B is longer than A	under	understanding	0
6	A = B	understanding	understanding	1
7	A contains several B	abbababba	abba	2
8	B overlapping in A	aaaa	aa	3

b) Write a C program implementing your algorithm and make sure it runs for all the special cases you provided. Include a function `int substrings(char A[], char B[])` which returns the number of occurrences of B in A. Your program should print the number of occurrences along with the starting and ending index of each occurrence.

This function first checks for special cases where either A or B is null or empty and if any of these conditions is satisfied, then it returns 0. Otherwise it matches each character of A with each character of B and increase counter `matchingChars` in character of A matches with character of B. It also prints pair containing starting and ending point of matching substrings once complete string B is matched with substring of A and increases the occurence counter as well.

```c
int substrings(char A[], char B[]) {

  /* Cases 3 & 4 */
  if(A == NULL || B == NULL) {
    return 0;
  }

  /* Cases 1 & 2*/
  if(A[0] == '\0' || B[0] == '\0') {
    return 0;
  }

  /* Calculate length of B */
  int sizeB = 0;
  while(B[sizeB] != '\0') {
    sizeB++;
  }

  int numOccurrences = 0;
  int currentPosition = 0;

  /* #chars matching B, starting from current position */
```

```c
  int matchingChars = 0;

  while(A[currentPosition] != '\0') {

    matchingChars = 0;

    /* Cases 5 & 6 */
    while ( B[matchingChars] != '\0' &&
    A[currentPosition + matchingChars] == B[matchingChars]) {
      matchingChars++;
    }
    if(matchingChars == sizeB) {
      printf("(%d,%d) ", currentPosition + 1, currentPosition + sizeB);
      numOccurrences++;
    }
    /* Cases 7 & 8 */
    currentPosition++;
  }

  return numOccurrences;
}
```

Attention! You are not allowed to use string-functions and/or `string.h`.