Design of Modern Algorithms

Khazar University, Fall 2019

Final exam preparation questions

1. Algorithmics
    - Explain the concept of algorithm in your own words.
    - What is correctness of an algorithm?
    - What is the difference between partially and fully correct algorithm?
    - What is termination proof?
    - What are two main qualities to consider when analyzing the efficiency of an algorithm?
    - Which of the following formulas can be considered an algorithm for computing the area of a triangle whose side lengths are given positive numbers a, b, and c?
        a. $S = \sqrt{p(p-a)(p-b)(p-c)}$, where $p = (a+b+c)/2$
        b. $S = \frac{1}{2}bcsinA$, where A is the angle between sides b and c
        c. $S = \frac{1}{2}ah_a$, where h_a is the height to base a
    - What are some of the problem types in algorithmics?
    - Give some examples of graph problems.
    - Give some examples of string processing problems.
    - What is traveling salesman problem?
    - What are some real-world applications of string matching algorithms?
    - What is the worst-case and best-case efficiency of an algorithm?
    - For each of the following functions, indicate how much the function's value will change if its argument is increased fourfold.
        a. $\log_2 n$
        b. $\sqrt{n}$
        c. n
        d. n^2
        e. n^3
        f. 2^n
    - For each of the following algorithms, indicate (i) a natural size metric for its inputs, (ii) its basic operation, and (iii) whether the basic operation count can be different for inputs of the same size:
        a. computing the sum of n numbers
        b. computing $n!$
        c. finding the largest element in a list of n numbers
    - Imagine that after washing 5 distinct pairs of socks, you discover that two socks are missing. Of course, you would like to have the largest number of complete pairs remaining. What are the best-case and the worst-case scenarios? Assuming that the probability of disappearance for each of the 10 socks is the same, find the probability of the best-case scenario; the probability of the worst-case scenario.

- Give an example of an algorithm that should not be considered an application of the brute-force approach.
- What is the time efficiency of the brute-force algorithm for computing a^n as a function of n?
- Sort the list [22, 3, 4, -7, 90, 5, 0, 11] using (i) bubble sort, (ii) merge sort.
- How many comparisons (both successful and unsuccessful) will be made by the brute-force algorithm in searching for each of the following patterns in the binary text of one thousand zeros?

  a. 00001 b. 10000 c. 01010
- What are some real-world applications of closest-pair problem?
- What is knapsack problem?

2. Pseudocode
   - Write pseudocode for an algorithm for finding real roots of equation
     $$ax^2 + bx + c = 0$$
     for arbitrary real coefficients a, b, and c. (You may assume the availability of the square root function $sqrt(x)$.)
   - Write pseudocode for an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5 and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively?
   - Consider the problem of counting, in a given text, the number of substrings that start with an *A* and end with a *B*. For example, there are four such substrings in *CABAAXBYA*. Write pseudocode for a brute-force algorithm for this problem. What input is the worst-case scenario?
3. Python code
   - Code the following algorithm in Python and explain what it does.

ALGORITHM *Alg1 (P)*

```
//Finds ?
//Input: A list P of n (n ≥ 2) points p1(x1, y1), . . . , pn(xn, yn)
//Output: ?
d ← ∞
for i ← 1 to n − 1 do
    for j ← i + 1 to n do
        d ← min(d, sqrt((xi − xj)^2 + (yi − yj)^2)) //sqrt is square root
return d
```

   - Code the following algorithm in Python and explain what it does.

ALGORITHM *Alg2*$(A[0..n-1])$

```
//Does ?
//Input: An array A[0..n − 1] of orderable elements
//Output:?
for i ← 0 to n − 2 do
    min ← i
    for j ← i + 1 to n − 1 do
        if A[j] < A[min]  min ← j
    swap A[i] and A[min]
```

- Code the following algorithm in Python and explain what it does.

ALGORITHM *Alg3 (P, Q)*

```
//Solves ?
//Input: An array P of n ≥ 2 points in the Cartesian plane sorted in
//       nondecreasing order of their x coordinates and an array Q of the
//       same points sorted in nondecreasing order of the y coordinates
//Output: ?
if n ≤ 3
    return the minimal distance found by the brute-force algorithm
else
    copy the first ⌈n/2⌉ points of P to array P_l
    copy the same ⌈n/2⌉ points from Q to array Q_l
    copy the remaining ⌊n/2⌋ points of P to array P_r
    copy the same ⌊n/2⌋ points from Q to array Q_r
    d_l ← Alg3 (P_l, Q_l)
    d_r ← Alg3 (P_r, Q_r)
    d ←min{d_l, d_r}
    m ← P[⌈n/2⌉ − 1].x
    copy all the points of Q for which |x − m| < d into array S[0..num − 1]
    dminsq ← d^2
    for i ← 0 to num − 2 do
        k ← i + 1
        while k ≤ num − 1 and (S[k].y − S[i].y)^2 < dminsq
            dminsq ← min((S[k].x − S[i].x)^2 + (S[k].y − S[i].y)^2, dminsq)
            k ← k + 1
return sqrt(dminsq)
```