ALGORITHMS AND DATABASES

Homework 1

Q1

Define an iterative algorithm *hasInverse(A)* that given a sorted array of positive and negative integers (different from zero), return *True* if exists at least a pair of elements such that *x* and *–x* belong to the array. The solution has to have linear running time.

(a) Write up your idea for a solution to this task.

(b) Write down pseudocode for *hasInverse(A)*.

Q2

Consider the searching problem:

Input: A sequence of n numbers $A = \langle a_1; a_2; \ldots ; a_n \rangle$ and a value v.

Output: An index i such that v = A[i] or the special value NIL if v does not appear in A.

Write pseudocode for linear search, which scans through the sequence, looking for v.

Q3

Bubble sort is a sorting algorithm that works by repeatedly stepping through lists that need to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. This passing procedure is repeated until no swaps are required, indicating that the list is sorted. Bubble sort gets its name because smaller elements bubble toward the top of the list.

a) Write pseudocode for the bubble sort algorithm and give the best-case and worst-case running times in Θ-notation.

b) Bubble sort is really inefficient; can you think of some variations to improve its performance?

c) Do they change the asymptotical running time performance in the worst or best case?

Q4

Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first n-1 elements of A.

a) Write pseudocode for this algorithm, which is known as selection sort.
b) Why does it need to run for only the first n-1 elements, rather than for all n elements?
c) Give the best-case and worst-case running times of selection sort in Θ-notation.

Q5

Referring back to the searching problem (Q2), observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. The binary search algorithm repeats this procedure, halving the size of the remaining portion of the sequence each time.

a) Write the pseudocode for the binary search.
b) Argue that the worst-case running time of binary search if Θ(log n).

Q6

Answer the following questions on the big-oh notation.

a. Explain what g(n) = O(f (n)) means.
b. Explain why it is meaningless to state that *"the running time of algorithm A is at least $O(n^2)$."*

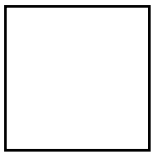
c. Explain the strategy that we usually adopt for guessing a big-oh running time from a polynomial complexity function (e.g., what we do with constants, negative terms, terms with different exponents, …)

Q7

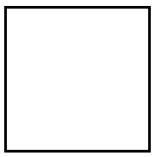
Describe the running time of the following pseudocode in Big-Oh notation in terms of the variable n. Assume all variables used have been declared.

```
int foo(int k) {
      int cost;
      for (int i = 0; i < k; ++i)
            cost = cost + (i * k);
      return cost;
}
```

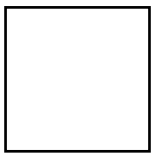
```
I. answ = foo(n);
```



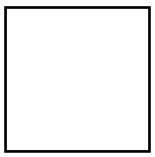
```
II.   int sum;
      for (int i = 0; i < n; ++i) {
      if (n < 1000)
            sum++
      else
            sum += foo(n);
      }
```



```
III.  for (int i = 0; i < n + 100; ++i) {
            for (int j = 0; j < i * n ; ++j){
                  sum = sum + j;
            }
            for (int k = 0; k < n + n + n; ++k){
                  c[k] = c[k] + sum;
            }
      }
```

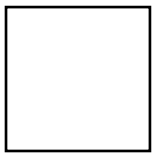


```
IV.   for (int j = 4; j < n; j=j+2) {
            val = 0;
            for (int i = 0; i < j; ++i) {
                  val = val + i * j;
                  for (int k = 0; k < n; ++k){
                        val++;
                  }
```



```
        }
    }
```

V.
```
    for (int i = 0; i < n * 1000; ++i) {
        sum = (sum * sum)/(n * i);
        for (int j = 0; j < i; ++j) {
            sum += j * i;
        }
    }
```



Q8

Show that for any real constants a and b, where b > 0,

$(n+a)^b = \Theta(n^b)$

Q9

Specify whether the following equations hold or not. Give an intuitive justification for your answer. Formal proofs are not needed.

a) $3n^2 + 5n = O(n^2 \log n)$

b) $2^n = O(n^4 \log \log n)$

c) $10n^2 = O(n^2 - 100n)$

d) $(n^2 + n)(n \log n + n^{1.1}) = O(n^3)$

e) $n^3 \log n = \Theta(n^3)$

f) $n^3 \log 5\ n = O(n^{3.01})$

g) $4^n = \Theta\ (3^n)$

h) $n^2 \log n(\log \log n)^4 = O(n^2 \log^2 n)$

i) $n^2 + n^{1.9} = O(n^2 + n^{0.5})$