XJTLU | Computing

Complexity analysis

A. Multiple Choice Questions

Problem 1. Despite feeling sleepy all day long, Omer has written a *correct* program that returns the median of a given table with n elements. You have not seen Omer's algorithm. Nonetheless, you can still assert that only one of the following statements is true. Reason which.

A. The cost of Omer's algorithm is necessarily $\Theta(n)$.

B. The cost of Omer's algorithm is necessarily $\Omega(n)$.

C. The average-case cost of Omer's algorithm is necessarily $O(\lg n)$.

D. The worst-case cost of Omer's algorithm is necessarily $O(n \lg n)$.

Problem 2. Use the following algorithm to answer the next three questions.

```
FOO
1  for i = 1 to N
2      for j = 1 to i
3          for k = 1 to i
4              sum = sum + 1
5  for p = 1 to N × N
6      for q = 1 to p
7          sum = sum − 1
```

(i) How many times is line 4 executed?

A. $O(N)$

B. $O(N^2)$

C. $O(N^3)$

D. $O(N^4)$

(ii) How many times is line 7 executed?

A. $O(N)$

B. $O(N^2)$

C. $O(N^3)$

D. $O(N^4)$

(iii) What is the total running time of the fragment?

A. $O(N^2)$

B. $O(N^3)$

C. $O(N^4)$

D. $O(N^5)$

B. Questions

Problem 3. What is wrong with the following binary search algorithm?

```
BINARY-SEARCH(A, x, l, r)
    // A: data array
    // x: value to be found
    // l: lower bound
    // r: upper bound
if l == r
    return l
else m = ⌊(l + r)/2⌋
    if x ≤ A[m]
        return BINARY-SEARCH(A, x, l, m)
    else return BINARY-SEARCH(A, x, m, r)
```

Problem 4. An algorithm takes $0.5\,\mathrm{ms}$ for input size 100. How long will it take for input size 500 if the running time is the following (assume low-order terms are negligible) [Weiss 2012]

(a) linear

(b) $O(N \lg N)$

(c) quadratic

(d) cubic

Problem 5. Let A be an array with n elements.

(a) Write an algorithm that makes exactly $n-1$ comparisons to find the maximum element in A.

(b) Write an algorithm that makes exactly $n-1$ comparisons to find the minimum element in A.

(c) Write an algorithm that finds both the maximum and the minimum elements in A and makes only $\frac{3}{2}n$ comparisons.

Problem 6. So that the number of comparisons used by the binary search algorithm when n is not necessarily a power 2 is at most $\lceil \lg n \rceil$.

Problem 7. Solve the following subtractive recurrences. (Assume that $T(1) = 1$ and n is a power of 2)

(a) $T(n) = T(n-1) + \Theta(1)$

(b) $T(n) = T(n-2) + \Theta(1)$

(c) $T(n) = T(n-1) + \Theta(n)$

(d) $T(n) = 2T(n-1) + \Theta(1)$

Problem 8. Solve the following divisive recurrences. (Assume that $T(1) = 1$ and n is a power of 2)

(a) $T(n) = 2T(n/2) + \Theta(1)$

(b) $T(n) = 2T(n/2) + \Theta(n)$

(c) $T(n) = 2T(n/2) + \Theta(n^2)$

(d) $T(n) = 4T(n/2) + n$

(e) $T(n) = 2T(n/2) + n^2$

(f) $T(n) = 9T(n/3) + 3n + 2$

(g) $T(n) = 2T(n/4) + \sqrt{n}$

Problem 9. For each of the following code segments, analyse its time complexity in terms of Big-Oh notation.

(a)
```
public void foo1(int n) {
  int s = 0;
  for (int i = 0; i < n; ++i) {
    ++s;
  }
}
```

(b)
```
public void foo2(int n) {
  int s = 0;
  for (int i = 0; i < n; i += 2) {
    ++s;
  }
}
```

(c)
```
public void foo3(int n) {
  int s = 0;
  for (int i = 0; i < n; ++i) {
    ++s;
  }
  for (int j = 0; j < n; ++j) {
    ++s;
  }
}
```

(d)
```
public void foo4(int n) {
  int s = 0;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
      ++s;
    }
  }
}
```

(e)
```
public void foo5(int n) {
  int s = 0;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
      ++s;
    }
  }
}
```

(f)
```
public void foo6(int n) {
  int s = 0;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
      for (int k = 0; k < n; ++k) {
        ++s;
      }
    }
  }
}
```

Problem 10. Analyze the time complexity of the following recursive functions that compute x^n for $n \geq 0$.

(a)
```java
public double power1(double x, int n) {
  if (n == 0) {
    return 1;
  } else {
    return x * power1(x, n - 1);
  }
}
```

(b)
```java
public double power2(double x, int n) {
  if (n == 0) {
    return 1;
  } else if (n % 2 == 0) {
    double y = power2(x, n / 2);
    return y * y;
  } else {
    double y = power2(x, n / 2);
    return y * y * x;
  }
}
```

(c)
```java
public double power3(double x, int n) {
  if (n == 0) {
    return 1;
  } else if (n % 2 == 0) {
    return power3(x, n / 2) * power3(x, n / 2);
  } else {
    return power3(x, n / 2) * power3(x, n / 2) * x;
  }
}
```

Problem 11. The cost of algorithm A is given by the recurrence $T_A(n) = 7\,T_A(n/2) + n^2$. A competing algorithm B has cost given by $T_B(n) = x\,T_B(n/4) + n^2$. Which is the largest integer x for which B is asymptotically better than A? [∗]

Problem 12. Solve the recurrence $T(n) = T(\sqrt{n} + 1)$ using a change fo variables. Assume that n is of the form 2^{2^i} for an integer $i \geq 0$. [∗]

Problem 13. Consider a bidimensional array of size $n \times n$ where each column has its elements sorted in strict increasing order from top to bottom, and each row has its elements sorted in strict decreasing order from left to right. [∗]

(a) Give an algorithm of cost $\Theta(n)$ that, given an element x, determine whether x is in the array.

(b) Give an algorithm of cost $\Theta(n)$ that, given an element x, determine how many elements in the array are strictly smaller than x.

References

Atserias, Albert et al. (2022). *Data Structure and Algorithms Problem Set*. Universitat Politécnica de Catalunya.
Parberry, Ian and William Gasarch (2002). *Problems on Algorithms*. 2nd.