CS 430 - Recitation Notes

Len Washington III

September 21, 2023

Contents

0.0.1 After Lecture 01 & 02 1
0.0.2 After Lecture 03 & 04 3
0.0.3 After Lecture 05 & 06 5

After Lecture 01 & 02 – Answer any questions on HW1
Practice Problems (all taken from previous exams)

1. Which of the following is not true of improved bubble sort (keep track of last swap position on the inner loop and use that to reduce outer loop iterations) on the case on input elements sorted?

   a) It is stable

   b) Consumes less memory

   c) Detects whether the input is already sorted

   d) Consumes less time

2.

Statement 1: In insertion sort, after m passes through the array, the first m elements are in sorted order.

Statement 2: And these elements are the m smallest elements in the array.

   a) Both of the statements are true.

   b) Statement 1 is true but statement 2 is false

   c) Statement 1 is false but statement 2 is true

   d) Both of the statements are false

3. Consider the following program that attempts to locate an element x in a sorted array $a[]$ using binary search. The program is erroneous. Under what conditions does the program fail?

Algorithm 1 Erroneous Binary Search

```
1: function BS
2:     int i=1, j=100, k, x          ▷ assume x is assigned a value to search for
3: end function
```

   a) x is the last element of the array $a[]$

b) x is greater than all elements of the array $a[]$

c) Both of the Above

d) x is less than the last element of the array $a[]$

4. What's the worst case of insertion sort if the correct position for inserting element is calculated using binary search?

a) $O(\log n)$

b) $O(n)$

c) $O(n \log n)$

d) $O(n^2)$

5. The following routine takes as input a list of n numbers, and returns the first value of i for which $L[i] < L[i-1]$, or n if no such number exists.

```
int firstDecrease(int* L, int n){
        for(int i=2; i <= n && L[i] >= L[i-1]; i++){}
        return i;
}
```

5a) What is the big-O runtime for the routine, measured as a function of its return value i?

5b) If the numbers are chosen independently at random, then the probability that firstDecrease(L) returns i is $\frac{i-1}{i!}$, except for the special case of $i = n+1$ for which the probability is $\frac{1}{n!}$ Use this fact to write an expression for the expected value returned by the algorithm. (Your answer can be expressed as a sum, it does not have to be solved in closed form. Do not use O-notation.) Use expectation

5c) What is the big-O average case running time of the routine? Hint: Simplify the previous summation until you see a common taylor series.

6. Some sorting algorithms are NOT stable. However if every key in $A[i]$ is changed to $A[i] * n + i - 1$ (assume $1 \leq i \leq n$) then all the new elements are distinct (and therefore stability is no longer a concern). After sorting, what transformation will restore the keys back to their original values? What is the effect on the runtime of any of the sorting algorithm __ add this transformation before executing the sort and un-transformation after the sort?

$$A[i] \rightarrow A[i] * n + i - 1$$

7. Use pseudocode to specify a brute-force algorithm that determines when a sequence of n positive integers is given as input, whether there are two distinct terms of the sequence that have as sum a third term. The

After Lecture 03 & 04 – Answer any questions on HW1
Practice Problems (all taken from previous exams)

1. Which of the following is time complexity of fun()?

```
int fun(int n){
        int count = 0;
        for(int i = 0; i < n; i++){
                for(int j = i; j > 0; j--){
                        count = count + 1;
                }
        }
        return count;
}
```

   a) $O(n)$
   b) $O(n^2)$
   c) $O(n \log n)$
   d) $O(n \log(n) \log(n))$

2. Consider the following function. What is the returned value of the above function?

```
int unknown(int n){
        int i, j, k=0;
        for(i = n/2; i <= n; i++){
                for(j=2; j <= n; j=j*2){
                        k = k + n/2;
                }
        }
        return k;
}
```

   a) $\Theta(n^2)$
   b) $\Theta(n^2 \log n)$
   c) $\Theta(n^3)$
   d) $\Theta(n^3 \log n)$

3. What is the worst-case auxiliary space complexity (including stack space for recursion) of merge sort?

   a) $O(1)$
   b) $O(\log n)$
   c) $O(n)$
   d) $O(n \log n)$

4. Choose the incorrect statement about merge sort from the following:

a) It is a comparison-based sort.

b) It's runtime is dependent on input order.

c) It is not an in-place (all the operations are on the original array) algorithm.

d) It is a stable algorithm.

5. Use the definition of big-O to prove or disprove.

5a) is $2^{n+1}? =?O\left(2^n\right)$True:

$$2^{n+1} = C2^n$$
$$2 = C \text{ if c } \geq 2$$

5b) is $2^{2n}? =?O\left(2^n\right)$False:

$$2^{2n} = C2^n$$
$$2^n = C$$

However 2^n has an infinite range so it cannot be upperbounded

6. Although merge sort runs in $\Theta(n \lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small n. Thus, it makes sense to use insertion sort within merge sort when sub-problems become sufficiently small. Consider a modification to merge sort in which n/k sub-lists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

6a) Show the n/k sub-lists; each of length k, can be sorted by insertion sort in $\Theta(nk)$ worst-case time.

$$k \to \Theta\left(k^2\right)$$
$$\frac{n}{k} \to \Theta\left(\frac{n}{k} \times k^2\right) = \Theta(nk)$$

6b) Show that the sub-lists can be merged in $\Theta(n \lg\left(\frac{n}{k}\right))$ worst-case time.

$$\frac{n}{k} \to \Theta\left(\frac{n}{2k}\right) \text{ for merging}$$
$$\Theta(2k) \text{ m?? for } 2k \text{ elements}$$
$$\Theta\left(\frac{n}{2k} \times 2k\right) = \Theta\left(n\right)\Theta\left(\frac{n}{4k} \times 4k\right) = \Theta\left(n\right)$$

6c) Given that the modified algorithm runs in $\Theta(nk + n \lg\left(\frac{n}{k}\right))$ worst-case time, what is the largest asymptotic (Θ-notation) value of k as a function of n for which the modified algorithm has the same asymptotic running time as standard merge sort?

6d) How should k be chosen in practice?

7. The Fibonacci sequence 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... is defined recursively as

Algorithm 2 The Fibonacci sequence

```
1: function FIB(n)      ▷ This mathematical definition leads naturally to a recursive algorithm
2:     if n ≤ 1 thenreturn n
3:     end if
4:     return FIB(n-1) + FIB(n-2)
5: end function
```

7a) Write the recurrence relation, $T(n)$, for the asymptotic runtime for procedure FIB(n) shown above, and solve the recurrence relation to show that $T(n) = O\left(2^{n-2}\right)$.

7b) Another recursive procedure which computes the nth Fibonacci number is below.

Algorithm 3

```
1: function F1(n)
2:     if n < 2 thenreturn n
3:     else return F2(2, n, 1, 1)
4:     end if
5: end function
6:
7: function F2(i,n,x,y)
8:     if i ≤ n then
9:         F2(i+1, n, y, x+y)
10:    end if
11:    return x
12: end function
```

Trace out the algorithm as it computes $F1(1)$, $F1(2)$, $F1(3)$, $F1(4)$, explain how the algorithm works, and then compare its asymptotic runtime to the time for procedure FIB(n).

8. Use mathematical induction to show that when n is an exact power 2, the solution of the recurrence

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

is $T(n) = n \lg n$

After Lecture 05 & 06 – Answer any questions on HW1
Practice Problems (all taken from previous exams)

1. What are the max number of levels in the recusion tree for this recurrence relation?

   a) $\log_4(n)$

   b) $\log_2(n)$

   c) $\log_{\frac{4}{3}}(n)$

d) $\log_{\frac{1}{3}}(n)$

2. Under what case of Master's Theorem will the recurrence relation of binary search fail?

   a) 1

   b) 2

   c) 3

   d) It cannot be solved using Master's Theorem.

3. What is the purpose of using randomized quick sort over standard quick sort?

   a) Improve the worst-case runtime

   b) To eliminate the possibility that a particular input order will always yied worst-case runtime

   c) To improve accuracy of output

   d) To improve average case time complexity

4. The non-recursive work in quicksort is done in which step of the divide-conquer-combine algorithm?

   a)

5. Give big-O bounds of T(n) in each of the following recurrences. Use induction, iteration or Master Theorem.

   5a)

$$T(n) = T(n-1) + n \; T(1) = O(1)$$

$$T(n) = n + T(n-1)$$
$$T(n) = n + n - 1 + T(n-2)$$
$$T(n) = n + n - 1 + n - 2 + T(n-3)$$
$$T(n) = n + n - 1 + n - 2 + \cdots + T(1)$$
$$T(n) = n + n - 1 + n - 2 + \cdots + O(1)$$
$$T(n) = \sum_{1}^{n} n$$
$$T(n) = O(\frac{n^2 + n}{2})$$
$$T(n) = O(n^2)$$

   5b)

$$t(n) = 2T\left(\frac{n}{4} + n^{\frac{1}{2}}T(1) = O(1)\right)$$

   5c)

$$T(n)$$

6. Throughout this course, we assume that parameter passing during procedure calls takes constant time, even if an N-element array is being passed. This assumption is valid in most systems because a pointer to the array is passed, not the array itself. This problem examines the implications of three parameter-passing strategies:

   1. An array is passed by pointer. Time = $\theta(1)$.
   2. An array is passed by copying. Time = $\theta(N)$, where N is the size of the array.
   3. An array is passed by copying only the subrange that might be accessed by the called procedure. Time = $\theta(q - p + 1)$ if the subarray $A[p \ldots q]$ is passed. Use $n = q - p + 1$, where n is the size of the subarray passed.

   Consider the recursive binary search algorithm for finding a number in a sorted array. Give recurrences for the worst-case running times of binary search when arrays are passed using each of the three methods above, and give good upper bounds on the solutions of the recurrences. Let N be the size of the original problem and n be the size of a subproblem.Binary search works by comparing the element for which you are searching to the elemenet at index $\frac{p-r}{2}$ of a subarray of size n, where p is the first index of the subarray and r is the last index (integer division is used). Therefore, the array passed inot binary search is continually divided in hald.

   1)

   $$T(n) = T\left(\frac{n}{2}\right) + O(1) \text{ with } T(1) = O(1)$$

   The array is passed by pointer, which is constant time. Therefore, the time involed in tha

   2)

7. Use the definition of Θ abd induction to prove that the recurrence $T(n) = T(N-1) + \theta(n)$ (worst case Quicksort) has the solution $T(n) = \Theta(n^2)$. Since we are not given any boundary conditions, we cann assume the basis step for the inductive proof. Assume the claim is true for $n = k$.
   $T(k) = \theta(k^2)$, in other words assume $c_1 k^2 \leq T(k) \leq c_2 k^2$ for some $c_1 > 0$, $c_2 > 0$ and k large enough.
   Use that to prove the claim

8. What is you are sorting a collection of data that can have multiple entries of the sum of the values. When calling Quicksort's Partition(A, p, r), where do elements equal to the pivot end up and why?How could we modify Quicksort and Partition (write psuedocode) so that if we happen to partition on a pivot that had many duplicate values, we can improve the runtime of Quicksort by having smaller recursive calls?

Algorithm 4 Quicksort1

```
1: function QUICKSORT1(A, p, r)                          ▷
2:     if p < r then
3:         (q1, q2) = PARITION1(A, p, r)                 ▷ two return values
4:         (()A, p, q1 − 1)
5:     end if
6: end function
```

Algorithm 5 Partition1

```
1: function PARTITION1(A, p, r)                          ▷ Two return values
2:     endLow ← p − 1, endEqual ← p − 1
3:     pivot ← A[r]
4:     for j=p to r-1 do
5:         get function
6:     end for
7: end function
```