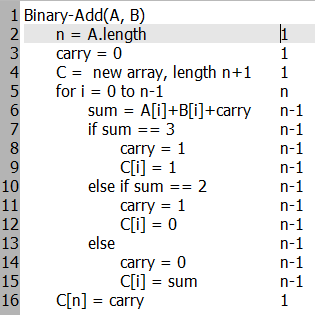
Assignment 1 4041 Algorithms and Data Structures

DUE: Sunday, Sept 18th, 10:00 pm Fall 2016

Wyatt Kormick

1. Binary Addition. Write pseudocode to add two n-bit binary numbers. Use two n-element arrays to represent each of the operands and an n+1-element array to store the results. **The least significant bit is stored in the last element of the array.** Analyze the runtime as demonstrated on INSERTION-SORT(A) in Chapter 2. For each line, define the number of times it is executed, and write the equation for the best and worst case runtime of the entire algorithm.



T(n)=c1+c2+c3+c16+c5n+c6(n-1)…c15(n-1) = c5n+(c6+c7+…+c15)(n-1)+C

C=c1+c2+c3+c16

Best Case (Only first if occurs):

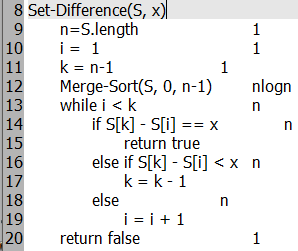
T(n)=c5n+(c6+c7+c8+c9)(n-1)+C

Worst Case (Only Else occurs):

T(n)=c5n+(c6+c7+c10+c13+c14+c15)(n-1)+C

The runtime of the loop doesn’t get much better or worse. No matter the size of n (the length of A and B) the loop will run n times. There is some variation given the nature of the conditionals.

2. Write pseudocode for a THETA(nlgn) algorithm that, given a set S (i.e. unique numbers) of n integers and another integer x, determines whether or not there exist two numbers in S whose difference is exactly x. Briefly justify the runtime. Use a loop invariant to prove your algorithm is correct.



nlogn+cn+C. Merge-Sort has been proven to run in Θ(nlgn) time, and is the most expensive operation in this algorithm. Worst case, the while-loop runs in O(n) time.

S[k]-S[i] gets closer to x every iteration

LI: S[0]<=S[i] <= S[k]<=S[n-1],

Base Case: n=2, (S[0]=S[i])<(S[k]=S[1]),

Termination: i=k, S[0]<= S[i]=S[k]<=S[n-1] or S[k]-S[i]=x

3.Use mathematical induction to show that when *n* is an exact power of 2, the solution of the following recurrence is *T*(*n*) = *n* lg *n*. Don't forget to provide a base case.

T(n) = { 2 if n = 2

2T(n/2) + n if n = 2k, for k > 1

Base Case: n=21 (k=1), T(2) = 2

T(4) = 2T(2)+4 = 8 = 4\*log\_2\_4

T(8) = 2T(4)+8 = 24 = 8\*log\_2\_8

Guess: n log n

T(2k) = 2T((2k-1))+2k = n log n

k=k+1 -> T(2k+1) = 2T(2k)+2k+1

=T(2k+1)=2(2T((2k-1))+2k)+2k+1

=4T(2k-1)+2k+2 = n log n

4. Let f(n) and g(n) be asymptotically positive functions. Prove or disprove each of the following conjectures. To disprove a statement, you need to provide only 1 counterexample.

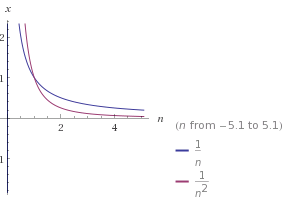
1. f(n) = O(g(n)) implies g(n) = O(f(n))
   1. Counterexample:
   2. f(n) = n, g(n) =n2
   3. f(n) = O(n2)
   4. g(n) != O(n)
   5. Because f(n) is smaller than n2 (it can grow to n2 with O(f(n))), but g(n) is already asymptotically larger than f(n)

b. f(n) = O(g(n)) implies lg(f(n)) = O(lg(g(n)), where lg(g(n)) ≥ 1 and f(n) ≥ 1 for all sufficiently large n

c. f(n) = O((f(n))2)

a. f(n) = 1/n

b. f(n) is asymptotically larger than f(n)2. f(n) is an upper bound on f(n)2



5. Substitution Method

Using the master method in Section 4.5, you can show that the solution to the recurrence T(n) = 8T(n/4) + n is T(n) = ϴ(nlog48 ). Show that a substitution proof for establishing the upper bound with the assumption T(k) ≤ ck log48 fails. Then show how to subtract off a lower-order term to make a substitution proof work.

nlog\_4\_8=n1.5, O(n1.5)=n2, Guess: n2

T(n) <= cn2

T(n) <= 8T(n/4) +n <= 8c(n2/16)+n <= cn2/2 +n <= cn2+n which is not <= cn2

We can subtract the lower order term n to fix this

New Assumption: T(n) <= cn2-xn

T(n) <= 8T(n/4) +n <= 8(cn2/16 – xn/4)+n <= (cn2/2-2xn+n = cn2/2-n(2x+1)) <= cn2+n(1-2x)

When x = 1, cn2 – n, and cn2 - n <= cn2-xn which is our original assumption

6. Master Theorem

Use the Master Theorem to prove the bounds of the following. Justify your answer by defining f(n), a, b, and a bound for epsilon, if appropriate. If using rule 3, show that the second condition also holds.

1. T(n) = 8T(n/3) + n2.
   1. a=8, b=3, f(n)=n2
   2. log\_b\_a = log\_3\_8 = ~1.89
   3. f(n) = n2 = Omega(nC), ((c = 2) > 1.89), Case Three
   4. 8\*f(n/3) <= kf(n)
      1. (8n2)/9 <= kn2, k<1, There exists a k that makes this true
   5. T(n) = Theta(f(n)) = Theta(n2)
2. T(n) = 7T(n/3) + nlgn
   1. a=7, b=3, f(n)=nlgn
   2. log\_3\_7 = ~1.77
   3. f(n) = O(nc) c< 1.77, case one
   4. T(n) = Theta(nlog\_3\_7)
3. T(n) = 4T(n/2) + n2.
   1. a=4, b=2, f(n)=n2
   2. log\_2\_4 = 2
   3. f(n) = Theta(nlog\_2\_4) = Theta (n2), case two
   4. T(n) = Theta(n2logn)