CSCI 3104 Spring 2018

Problem Set 2

Merola, Michael

06/04/1998

Problem Set 2

Problem 1

(20 pts total) Solve the following recurrence relations using any of the following methods: unrolling, tail recursion, recurrence tree (include tree diagram), or expansion. Each each case, show your work.

(a) $T(n) = T(n-2) + C(n)$ if n > 1, and T(n) = C otherwise

```
T(n)  = T(n-2) + C(n)
      = T(n-2-2) + C(n-2) + C(n)
      = T(n-2-2-2) + C(n-4) + C(n-2) + C(n)
      = T(n-2-2-2-2) + C(n-6) + C(n-4) + C(n-2) + C(n)
      ...
```

$= C + \sum_{i=0}^{n} C(n-2i)$

$= \sum_{i=0}^{n} C(n) + \sum_{i=0}^{n} -C(2i)$ ......split into two sums, constant irrelevant in asymptopia

$= C - 2C\sum_{i=0}^{n} i$ ......pull constants outside of sum

$= \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

$= \Theta(n^2)$

(b) $T(n) = 3T(n-1) + 1$ if n > 1, and T(1) = 3

```
T(n) = 3T(n-1) + 1
     = 9T(n-1-1) + (3*1) + 1
     = 27T(n-1-1-1) + (3*3*1) + (3*1) + 1
     ...
```

$= 3^n * T(1) + 3 + 3^2 + 3^3 + ... + 3^n$ ......eventually reach base case as we go to infinity

$= 3^n * (3) + 3 + 3^2 + 3^3 + ... + 3^n$ ......plug in base case T(1) = 3

$= 3^n * (3) + \sum_{i=0}^{n} 3^i$ ......constants become irrelevant at infinity

$= \Theta(3^n)$

(c) $T(n) = T(n-1) + 3^n$ if n > 1, and T(1) = 3

$T(n) = T(n-1) + 3^n$

$= T(n-2) + 3^{n-1} + 3^n$ ......recurrence unrolling

$= T(n-3) + 3^{n-2} + 3^{n-1} + 3^n$

...

$= T(1) + 3^2 + 3^3 + ... + 3^n$ ......eventually reach base case as we go to infinity

$= 3 + 3^2 + 3^3 + ... + 3^n$ ......plug in base case T(1) = 3

$= \sum_{i=1}^{n} 3^i$ ......establish sum

$= \frac{1-3^{n+1}}{2} = \Theta(3^n)$

(d) $T(n) = T(n^{\frac{1}{4}}) + 1$ if n > 2 , and T(n) = 0 otherwise

$T(n) = T(n^{\frac{1}{4}}) + 1$

$= T(n^{\frac{1}{16}} + n^{\frac{1}{4}}) + 1$ ......recurrence unrolling

$= T(n^{\frac{1}{64}} + n^{\frac{1}{16}} + n^{\frac{1}{4}}) + 1$

...

$= T(1) + \sum_{i=1}^{n} n^{\frac{1}{4^i}} + 1$ ......eventually reach base case as we go to infinity

$= 0 + \sum_{i=1}^{n} n^{\frac{1}{4^i}} + 1$ ......plug in base case T(n) = 0

$= \Theta(log(n))$

Problem 2

(10 pts) Consider the following function:

In [2]:
```
def foo(n):
    if (n > 1):
        print("hello")
        foo(n/3)
        foo(n/3)
```

In terms of the input n, determine how many times is *hello* printed. Write down a recurrence and solve using the Master method.

Master Method - $a * T(\frac{n}{b}) + f(n)$ and $\log_b a$

a = 2 - recursive call foo 2 times every iteration

b = 3 - recursive input divides n by 3 every iteration

f(n) = 1 - print(hello) does not rely on n

$T(n) = 2T(\frac{n}{3}) + 1$

$\log_3 2 = 0.6$

$\epsilon > 0$ because f(n) is 1

$T(n) = \Theta(n^{0.6})$

Problem 3

(30 pts) Professor McGonagall asks you to help her with some arrays that are raludominular. A raludominular array has the property that the subarray A[1::i] has the property that A[j] > A[j + 1] for 1 ≤ j < i, and the subarray A[i::n] has the property that A[j] < A[j + 1] for i ≤ j < n. Using her wand, McGonagall writes the following raludominular array on the board A = [7, 6, 4, −1, −2, −9, −5, −3, 10, 13], as an example.

Write a recursive algorithm that takes asymptotically sub-linear time to find the minimum element of A.

(a) Write a recursive algorithm that takes asymptotically sub-linear time to find the minimum element of A.

In [7]:
```
def findMin_init():
    A = [7, 6, 4, -1, -2, -9, -5, -3, 10, 13]
    findMin(A, 0, length(A))

def findMin(A, left, right):
    cut = floor(left + right / 2)
    if (A[cut] > A[cut + 1]):  #if the value at cut is greater than the n
ext value, array is decreasing
        findMin(A, cut, right) #enter right half
    elif (A[cut] > A[cut - 1]):#if the value at cut is greater than the l
ast value, array is increasing
        findMin(A, left, cut)  #enter left half
    else:
        return A[cut]
```

(b) Prove that your algorithm is correct. (Hint: prove that your algorithm's correctness follows from the correctness of another correct algorithm we already know.)

Loop Invariant: If v is the min in A, then v must always be between A[left] and A[right] or less than both of them (A[left] > v < A[right])

Initialization: v is a value in array A with bounds left = 0 and right = length(A) ; so (A[left] <= v <= A[right])

Maintenance: During every recursion, the bounds reduce by half.

```
                If A[cut] > A[cut + 1], then it is between bounds cut and ri
ght
                If A[cut] > A[cut - 1], then it is between bounds left and c
ut
                If A[cut] < then the other elements, then the algorithm term
inates and v is the minimum.
```

Termination: The bounds left and right eventually converge on the smallest value based on a raludominular sequence.

(c) Now consider the multi-raludominular generalization, in which the array contains k local minima, i.e., it contains k subarrays, each of which is itself a raludominular array. Let k = 2 and prove that your algorithm can fail on such an input.

$A = [6, 4, 2, 5, 7, 16, 14, 13, 12, 15, 17] = [6, 4, 2, 5, 7, 16] + [14, 13, 12, 15, 17]$.two raludominular subarrays

min = 5, 12

Counter-Example:

```
1) cut = A[5] == 16
2) A[cut] > A[cut + 1] == 16 > 14  **enters right half even though 16 is als
o greater than 7
3) cut2 == 13
4) A[cut2] > A[cut2 + 1] == 13 > 12 **enters right half again
5) Algorithm determines 12 is the minimum value in A
```

The algorithm fails to return 5 as the minimum value of A because it is designed to find the minimum in the pattern of one raludominular array. The algorithm greedily chooses the next index to compare as the greatest in step (2), so it completely overlooks the minimum in the left half of A.

(d) Suppose that k = 2 and we can guarantee that neither local minimum is closer than n=3 positions to the middle of the array, and that the joining point of the two singly-raludominular subarrays lays in the middle third of the array. Now write an algorithm that returns the minimum element of A in sublinear time. Prove that your algorithm is correct, give a recurrence relation for its running time, and solve for its asymptotic behavior.

$A = [6, 4, 2, 5, 7, 16, 18, 14, 13, 12, 15, 17]$...Ideal k=2 multi-randominular array

I don't know how to do this problem.

Problem 4

(15 pts extra credit) Asymptotic relations like O, Ω, and Θ represent relationships between functions, and these relationships are transitive. That is, if some f(n) = Ω(g(n)), and g(n) = Ω(h(n)), then it is also true that f(n) = Ω(h(n)). This means that we can sort functions by their asymptotic growth.

Sort the following functions by order of asymptotic growth such that the final arrangement of functions g1, g2, ..., g11 = Ω(g12).

Give the final sorted list and identify which pair(s) functions f(n), g(n), if any, are in the same equivalence class, i.e., f(n) = Θ(g(n)).

Fastest Growth

$n!$,

e^n,

$(\frac{5}{4})^n$,

$(log(n))!$

$n^{1.5}$,

$n\log(n) = \Theta(\log(n!))$,

n,

$8^{log(n)}$,

$n^{\frac{1}{log(n)}}$,

$4^{\log*(n)}$,

42

Slowest Growth

Sources

People

Krish Dholakiya

Gustav Solis

George Allison

Eric Oropezaelwood

Erika Bailon