CSCI432 GW3

Brock Ellefson, Seth Severa

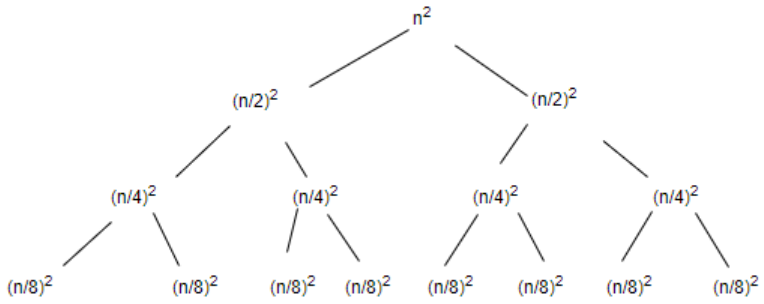
October 18, 2017

1 Solve the following recurrences:

1.a Show that T(n) = T(n?1)+n is O(n^2) using the substitution method

Prove that $T(n) = T(n-1) + n$ is in $O(n^2)$ complexity.
We begin by guessing that $T(n)$ is in $O(n^2)$
For the base case of n = 1, it's trivial to see that there exists a c where $1 < c1^2$
For $n > 1$, we have
$T(n) = T(n-1) + 1 \leq c(n-1)^2 + n = cn^2 - n(2c-1) + c$
We can see that this last simplification is less than cn^2 for some positive n and c. Since we are limiting our $n > 1$ already, we meet the requirement that there does exist some c where $T(n) < cn^2$.
Thus, by induction, $T(n) is O(n^2)$.

1.b Use a recursion tree to determine a good asymptotic upper bound for T(n) = T(n/2) + n^2



This is a geometric series, $\frac{n^2}{2^i}$, So our asymptotic upper bound of T(n) = T(n/2) + n^2 is approximately O(n^2)

1.c Use the master method to solve T(n) = 2T(n/4) + 1.

Master's Theorem:
T(n) = aT($\frac{n}{b}$) + f(n)
Where, a $\geq$ 1, b > 1, f(n) is asymptotically positive

T(n) = 2T(n/4) + 1
a = 2, b = 4, f(n) = 1
$n^{log_b a} \Rightarrow n^{log_4 2} \Rightarrow n^{1/2}$
Case 1:
if f(n) = $O(n^{log_b a - \epsilon})$ for some $\epsilon > 0$ then T(n) = $\Theta(n^{1/2})$.
Therefore, T(n) = $\Theta(\sqrt{n})$

1.d Use the master method to solve T(n) = 2T(n/4) + $\sqrt{n}$.

Master's Theorem:
T(n) = aT($\frac{n}{b}$) + f(n)
Where, a $\geq$ 1, b > 1, f(n) is asymptotically positive

T(n) = 2T(n/4) + $\sqrt{n}$
a = 2, b = 4, f(n) = $\sqrt{n}$
$n^{log_b a} \Rightarrow n^{log_4 2} \Rightarrow n^{1/2}$
Case 2:
if f(n) is $\theta(n^{log_b a})$ then T(n) = $\theta(n^{log_b a} \lg n)$
Therefore, T(n) = $\theta(\sqrt{n} \lg n)$

1.e Use the master method to solve T(n) = 2T(n/4) + n

Master's Theorem:
T(n) = aT($\frac{n}{b}$) + f(n)
Where, a $\geq$ 1, b > 1, f(n) is asymptotically positive

T(n) = 2T(n/4) + n
a = 2, b = 4, f(n) = n
$n^{log_b a} \Rightarrow n^{log_4 2} \Rightarrow n^{1/2}$
Case 3: if f(n) is $\Omega(n^{log_b a + \epsilon})$ for some $\epsilon > 0$ and if f(n/b) $\leq$ f(n), then T(n) = $\theta(f(n))$
Therefore, T(n) = θ(n)

2 Suppose that the random choice of vertex always chooses the vertex with the smallest index. What are the return values of each recursive call?

Algorithm 1 SEB

1: **Input:** S, Σ
2: **Output:** B, the smallest ball enclosing S with points of Σ on the boundary.
3:
4: **if** $|S| = 0$ **then**
5: Compute the smallest ball with Σ on boundary
6: **end if**
7: $i \leftarrow \text{RANDOM}(0, n-1)$
8: $B = \text{SEB}(S \backslash S[i], \Sigma)$
9: **if** $S[i] \in B$ **then**
10: **return** B
11: **else**
12: **return** $\text{SEB}\left((S \backslash S[i], \Sigma \cup \{S[i]\}\right)$
13: **end if**

1st Return: $\emptyset$
2nd Return: ball centered at (4,1) radius (0)
3rd Return: ball centered at (2.5,1) radius (1.5)
4th Return: ball centered at (2.5,1) radius (1.5)
5th Return: ball centered at (2.5,2) radius (1.8)
6th Return: ball centered at (2.5,2) radius (1.8)
7th Return: ball centered at (4.5,2.5) radius (1.6)
8th Return: ball centered at (3,2.5) radius (2.5)
9th Return: ball centered at (3,2.5) radius (2.5)
10th Return: ball centered at (3,2.5) radius (2.5)
11th Return: ball centered at (3,2.5) radius (2.5)