Tun Myat
51705354

1.(1) $f_1(n) = n^{0.999999} \lg n$ ~~$\approx n \lg n$~~ ~~$= \Theta(n \lg n)$~~

$f_2(n) = 10000000\,n = \Theta(n)$

$f_3(n) = 1.000001^n$ ~~$= \Theta(1^n)$~~ ~~$= \Theta(1)$~~

$f_4(n) = n^2 = \Theta(n^2)$

Fastest (1) ~~$f_3(n) = \Theta(1^n)$~~ ~~$\Theta(1)$~~ $f_1(n) = n^{0.999999} \lg n$

(2) $f_2(n) = \Theta(n)$

(3) ~~$f_1(n) = \Theta(n \lg n)$~~ $f_3(n) = 1.000001^n$

slowest (4) $f_4(n) = \Theta(n^2)$

1.(2) $f_1(n) = 2^{2^{1000000}} = \Theta(1)$

$f_2(n) = 2^{100000n} = \Theta(2^n)$

$f_3(n) = n \lg n = \Theta(n \lg n)$

$f_4(n) = n\sqrt{n} = \Theta(n\sqrt{n})$

Fastest (1) $f_1(n) = \Theta(1)$

(2) $f_3(n) = \Theta(n \lg n)$

(3) $f_4(n) = \Theta(n\sqrt{n})$

slowest (4) $f_2(n) = \Theta(2^n)$

(3) $f_1(n) = \cancel{2^{\sqrt{n}}}\ n^{\sqrt{n}}$

$f_2(n) = 2^n$

$f_3(n) = n^{10} \cdot 2^{n/2}$

$f_4(n) = \sum_{i=1}^{n} (i+1)$

~~fastest (1) f_3 $f_1(n) = n^{\sqrt{n}}$~~

~~(2) $f_2(n) = 2^n$~~

~~(3) $f_4(n) = \sum_{i=1}^{n} (i+1)$~~

~~slowest (4) $f_3(n) = n^{10} \cdot 2^{n/2}$~~

Fastest (1) $f_3(n) = n^{10} \cdot 2^{n/2}$

(2) $f_4(n) = \sum_{i=1}^{n} (i+1)$

(3) $f_2(n) = 2^n$

(4) $f_1(n) = n^{\sqrt{n}}$

EECS 114

Homework 1

Tun Myat

51705354

Master method : $T(n) = aT(\frac{n}{b}) + f(n)$

$a \geq 1$, $b > 1$

compare : $f(n)$ with $n^{\log_b a}$

2. (a) $T(n) = 2T(n/3) + n\lg n$

$a = 2$, $b = \cancel{1/3}\ 3$, $f(n) = n\lg n$

$n^{\log_b a} = n^{\log_3 2} = n^{0.6}$ $\quad n\log n > n^{0.6}$

$f(n) = \Omega(n^{\log_3 2} + \epsilon)$

$\therefore T(n) = \Theta(n\lg n)$

(b) $T(n) = 3T(n/5) + \lg^2 n$

$a = 3$, $b = 5$, $f(n) = \lg^2 n$

$n^{\log_b a} = n^{\log_5 3} = n^{0.6}$ $\quad$ ~~$\lg^2 n > n^{0.6}$~~ $\quad n^{0.6} > \lg^2 n$

$f(n) = \Omega(n^{\log_5 3} - \epsilon)$

~~$\therefore T(n) = \Theta(\lg^2 n)$~~

$T(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_5 3})$

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

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a=1$, $b=2$, $f(n) = 2^n$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ $\quad 2^n > n^0$

$f(n) = \Omega(n^{\log_b a} + \varepsilon)$

$\therefore T(n) = \Theta(f(n)) = \Theta(2^n)$

(d) ~~$T(n) = T(\sqrt{n}) + \Theta(\lg\lg n)$~~

~~$a=1$, $b=1$, $f(n) = \Theta(\lg\lg n)$~~

~~$n^{\log_b a} = n^{\log_1 1} \geq 1$~~

~~$T(2^m) = T(2^{m/2}) + \Theta(\lg m)$~~

~~Let $S(m) = T(2^m)$~~

~~$S(m) = S\left(\frac{m}{2}\right) + \Theta(\lg m)$~~

~~$n^{\log_b a}$~~ ~~$n^{\log_b a} = n^{\log_2 1} = 1$~~] case 3

~~$f(n) = \Theta(\lg m)$~~

~~$T(n) = T(2^m) = S(m) = \Theta(\lg m)$~~

~~$T(n) = \Theta(\lg\lg n)$~~

(e) $T(n) = 10T(n/3) + 17n^{1.2}$

$T(n) = a\,T(\frac{n}{b}) + f(n)$

$a = 10$, $b = 3$, $f(n) = 17n^{1.2}$

$n^{\log_b a} = n^{\log_3 10} = n^{2.09} > 17n^{1.2}$

$f(n) = \Omega(n^{\log_b a - \varepsilon})$

~~$\therefore T(n) = \Theta(f(n)) = \Theta(17n^{1.2})$~~

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 10})$

(f) $T(n) = 7T(\frac{n}{2}) + n^3$

$a = 7$, $b = 2$, $f(n) = n^3$

$n^{\log_b a} = n^{\log_2 7} = n^{2.8} < f(n) = n^3$

$f(n) = \Omega(n^{\log_b a} + \varepsilon)$

$\therefore T(n) = \Theta(f(n)) = \Theta(n^3)$

(g) ~~$T(n) = T(n/2 + \sqrt{n}) + \sqrt{6046}$~~

~~For $n \to \infty$, $\sqrt{n} \ll \frac{n}{2}$~~

~~$\therefore T(n) = T(\frac{n}{2}) + \sqrt{6046}$~~

~~$a = 1$, $b = 2$, $f(n) = \sqrt{6046}$~~

~~$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ ⇐ case 2~~

~~$f(n) = \Theta(n^{\log_b a} \cdot \lg^k n)$ ⇐ $n^{\log_b a} \cdot \lg^k n = \lg^k n$~~

Homework 1

2.(g) $T(n) = T(\frac{n}{2} + \sqrt{n}) + \sqrt{6046}$

$T(n)$ is monotonically increasing function

For huge n value, $T(\frac{n}{2}) \leq T(\frac{n}{2} + \sqrt{n}) \leq T(\frac{3n}{4})$

$$\Theta(\lg n) \leq T(\frac{n}{2} + \sqrt{n}) \leq \Theta(\lg n)$$

$$\therefore T(n) = \Theta(\lg n)$$

(h) $T(n) = T(n-2) + \lg n$

$$T(n) = \sum_{j=1}^{n/2} \lg 2i \geq \sum_{j=1}^{n/2} \lg j \geq (n/4)(\lg \frac{n}{4}) = \Omega(n \lg n)$$

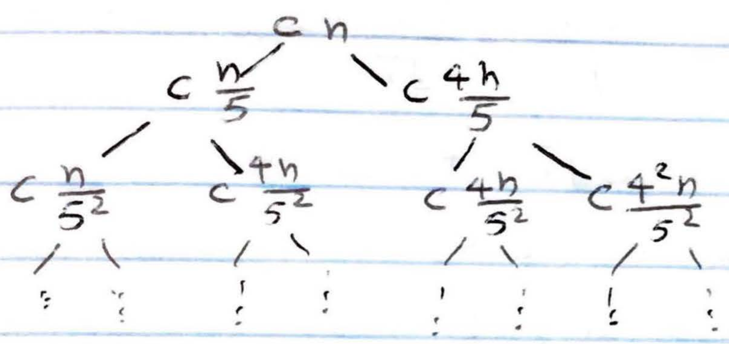
upper bound,

$T(n) \leq S(n)$

$S(n) = S(n-1) + \lg n$

$\therefore T(n) = \Theta(n \log n)$

(i) $T(n) = T(n/5) + T(4n/5) + \Theta(n)$



worst case : $h = \log_{5/4} n$

best case : $h = \log_5 n$

guess: $T(n) = \Theta(n \lg n)$

$\therefore T(n) = \Theta(n \lg n)$

(j) $T(n) = \sqrt{n}\, T(\sqrt{n}) + 100n$ $\qquad u(n) = \Theta(\lg m)$

$S(n) = \Theta(\lg \lg n)$

let $S(n) = \frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 100$

$S(n) = S(\sqrt{n}) + 100$

$S(2^m) = S(2^{m/2}) + 100$

let $u(m) = S(2^m)$

$u(m) = u(\frac{m}{2}) + 100$

$m^{\log_b a} = m^{\log_2 1} = 1$

$m^{\log_b a} \quad \lg^k m = 1$ $\qquad$ case 2

$f(n) = 100 = \Theta(1)$

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

(d) $T(n) = T(\sqrt{n}) + \Theta(\lg \lg n)$

let $m = \lg n$

$S(m) = S(m/2) + \Theta(\lg m)$ $\qquad$ case 2 :

$\therefore T(n) = \Theta((\lg \lg n)^2)$

Name - Tun Myat

ID - 51705354

3. Part A

Exercise 2.3-5

Recursive-Binary-Search pseudo-code

```
BinarySearch (A, value, low, high)
    if high <= low
        if value > A[low]
            return low+1
        else return low
    mid = (low + high)/2
    if value == A[mid]
        return mid+1
    else if value > A[mid]
        return BinarySearch (A, value, mid+1, high)
    else BinarySearch (A, value, low, mid-1)
```

The input parameters of BinarySearch are sorted array A, a selected value, and a range low and high of the array which contains the selected value. BinarySearch function will compare the value with each element and return the location that needs to be inserted the value at.

BinarySearch will only work on the sorted list. In the sorted list, we will put the pointer to low and high; then, find the middle position with mid = (low + high)/2. The middle position value will be compared with our selected value whether it is smaller or bigger.

If the selected value is bigger, we can eliminate the whole left side of the array. Then, the new low pointer will be set at mid+1. We will have a new mid pointer also. If the selected value is smaller, we will eliminate the whole right side of the array. Then, the new high pointer will be set at mid-1. We will have a new mid pointer also. The same process will occur recursively until the value is found or low >= high.

The procedure will terminate the search unsuccessfully when the range is empty such that low > high. It will terminate successfully if the searched value is found. Then, BinarySearch will return that pointer location. The search will continue with the range divided by 2.

The recurrence for the procedures is $T(n) = T\left(\frac{n}{2}\right) + \theta(1)$. So, the worst-case running time of binary search is $\theta(\lg n)$. The worst-case will happen when the selected value is not found in the whole array. In worst-case, we have to divide the array by 2 repletely until it runs out of elements.

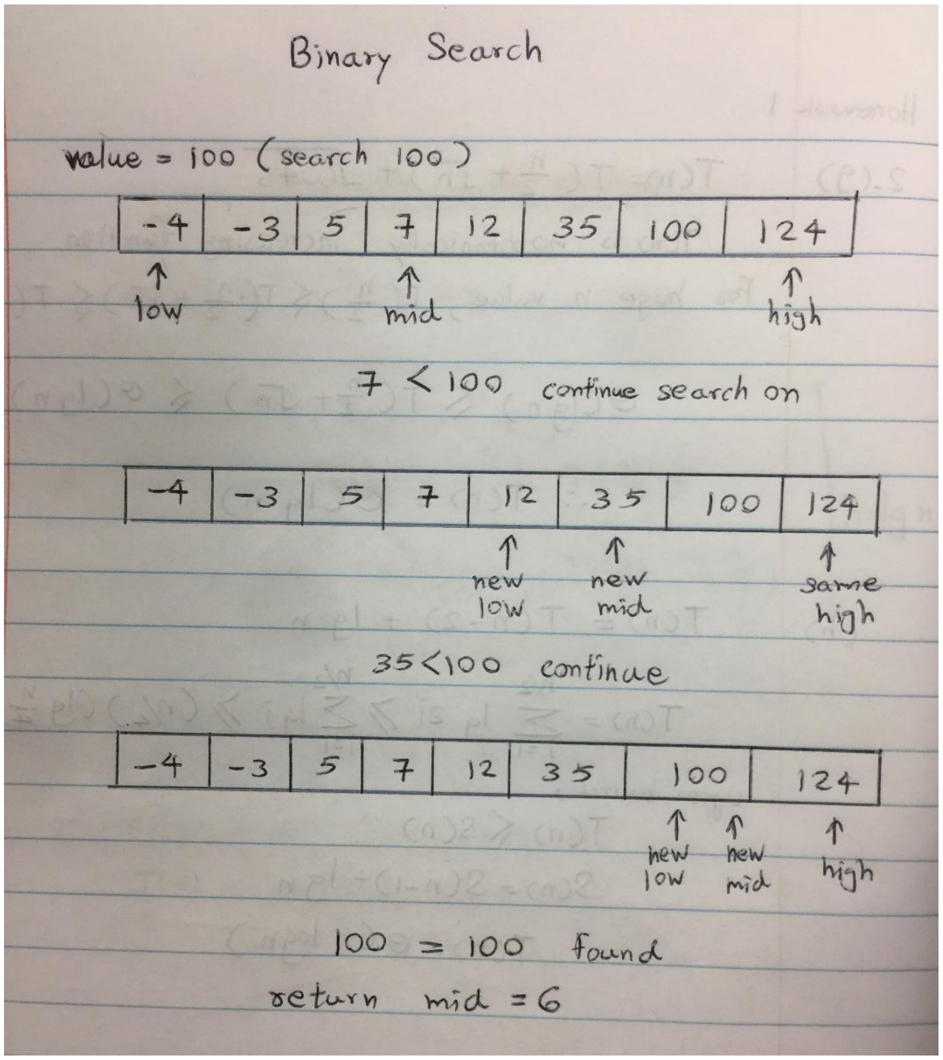


Fig 1. Binary Search working diagram

Exercise 2.3-6

No, binary search will not improve the worst-case running time. Insertion sort uses an incremental method. Binary Search is being used to reduce the number comparisons to find the necessary position. In the worst case, Binary Search will perform $\lceil \log^2(n) \rceil$ comparisons.

But the whole algorithm will still run for $\theta(n^2)$ run time because the shifting process runs at $\theta(n)$. And we have to shift all the elements greater than the selected value towards the end of the array to insert the value. Even in average case, we still need to shift half of the elements. So, the overall worst-case running time will still be $\theta(n^2)$.