Name:Mansi Jainendra Tandel

Assignment : 2

SCU id: W1606463

1. Use divide-and-conquer technique to search a number in the sorted list of n numbers.

a. Write the pseudocode of the algorithm.

```
int high= arr.length-1

int low=0

static int binarySearch(int arr[], int low, int high, int key)
  {
     if (h>=l)
     {
        int mid = l + (h - l)/2
        if (arr[mid] == key)
           return mid;
        if (arr[mid] > key)
           return binarySearch(arr, l, mid-1, key)
        return binarySearch(arr, mid+1, r, key);
     }
     return -1
  }
```

b. Write the recursive running time equation (recurrence)

```
int high= arr.length-1

int low=0
```

```
static int binarySearch(int arr[], int low, int high, int key)
  {
    if (h>=l)                                                          Θ(1)
    {
      int mid = l + (h - l)/2
      if (arr[mid] == key)                                             Θ(1)
        return mid;
      if (arr[mid] > key)                                              T(n/2)
        return binarySearch(arr, l, mid-1, key)
      return binarySearch(arr, mid+1, r, key);
    }
    return -1
  }
```

Running Time $T(n) = T(n/2)+2 *\Theta(1)$

$= T(n/2)+C$ (C=constant)

$= T(n/2)$

$= \Theta(\log_2 n)$

c. Guess the result of this recurrence, and use the Substitution method to prove your result.

$T(n) = T(n/2)$

$= T(n/2) +1^0$

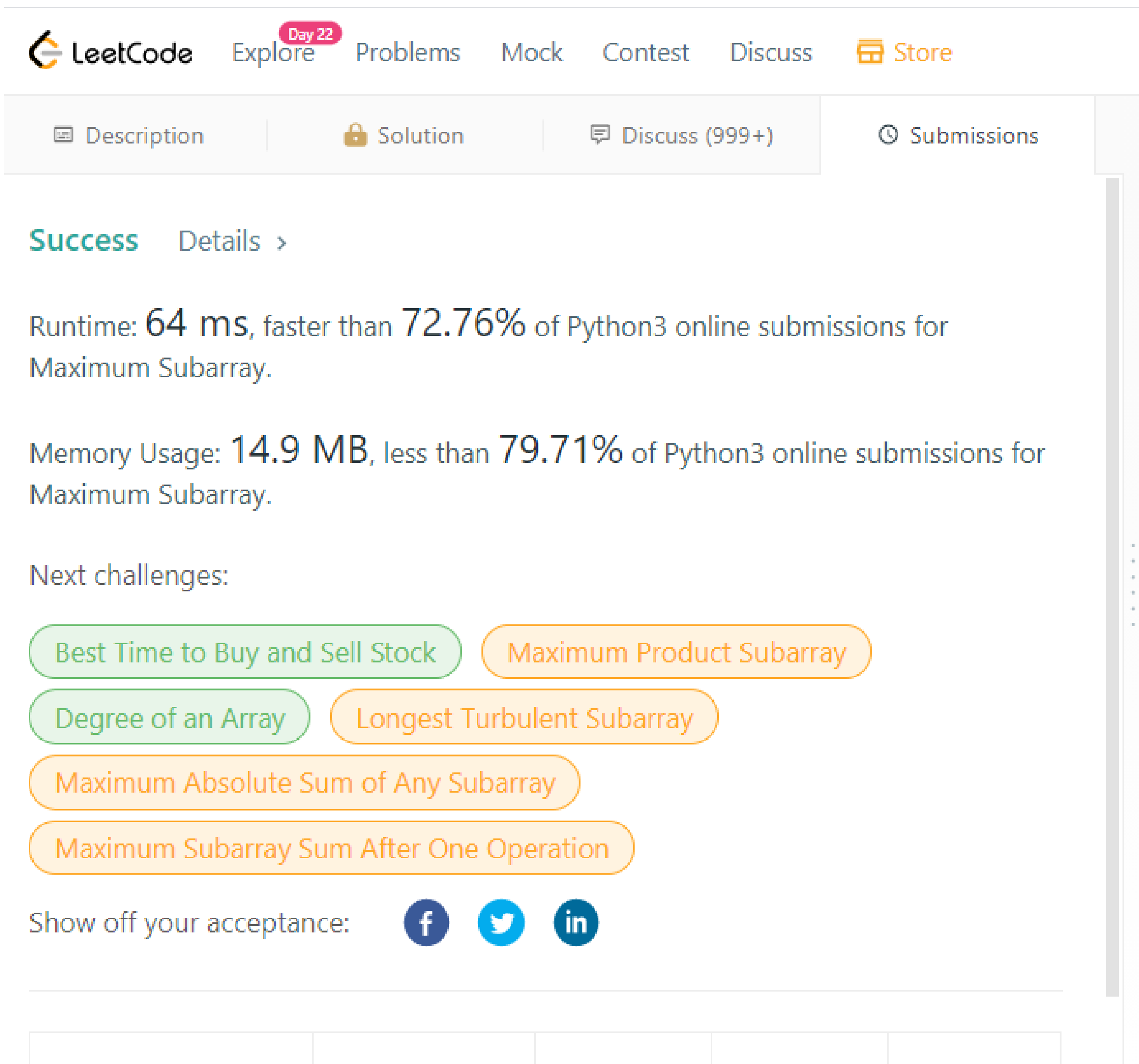
Here a=1,b=2,d=0,n=1 as $T(n) = aT(n/b) + f(n)$

$T(n) = \Theta(n^d \log_b n)$

$= \Theta(\log_2 n)$

Thus running time for binary search is $\Theta(\log_2 n)$, using Master theorem

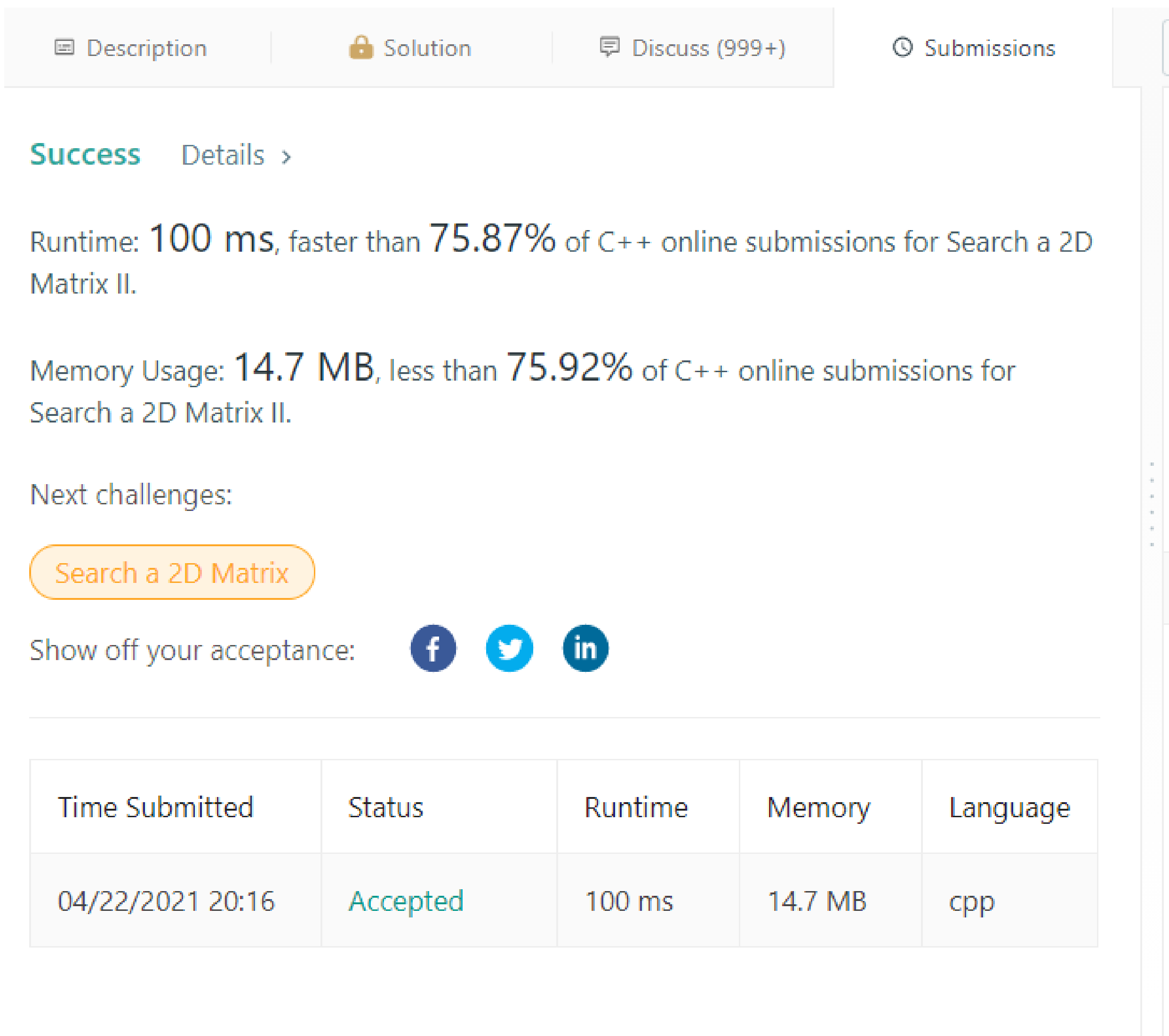
2. Solve the leetcode question no 53 (Max Subarray), Implement a solution that submission can be acceptation. Provide screen shot of your submission. (Check discussion for solution if you cannot figure it out yourself, a linear solution can be found in the file bentley-max-subarray.pdf in camino under week2, lecture 2-2)



3. Solve the leetcode question no. 240 (Search a 2D Matrix II)

a. Implementation a solution with recursive calls. Should pass all test cases except efficiency test, that is "exceed time limit" is ok (show screen snapshot)

b. Implement a solution that can be accepted. You can check "discussion" if you cannot figure out an efficient solution. (show screen snapshot)



5. Use master theorem to solve (if master theorem can not be applied, write the reason):

a. $T(n) = 9T(n/3) + n$
Here $a=9, b=3, d=1, f(n)=1$ as $T(n) = aT(n/b) + f(n)$

$\log_b a = \log_3 9 = 2 > 1 = d$
$f(n) = \Theta(n^d)$
$= \Theta(n^2)$

Thus by case 1:
$T(n) = \Theta(n^{\log_b a})$
$= \Theta(n^2)$

b. $T(n) = 9T(n/3) + 1000n^2$
Here $a=9, b=3, d=2, f(n)=1000n^2, k=0$ as $T(n) = aT(n/b) + f(n)$

$\log_b a = \log_3 9 = 2 = d$

$f(n) = \Theta(n^{\log_b a} \log^k n)$
$1000n^2 = \Theta(n^2)$

Thus by case 2:
$T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$
$= \Theta(n^2 \log n)$

c. $T(n) = 9T(n/3) + 1000n^3$
Here $a=9, b=3, d=3, f(n)=1000n^3$ as $T(n) = aT(n/b) + f(n)$

$\log_b a = \log_3 9 = 2 < d$

$f(n) = \Theta(n^{\log_b a})$
$n^3 = \Theta(n^{2+\varepsilon})$ put $\varepsilon = 1$, then the equality will hold.
$n^3 = \Theta(n^{2+1}) = \Omega(n^3)$
Thus by case 3:

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

d. $T(n) = 9T(n/3) + n^2 \log n$
Here $a=9, b=3, d=2, f(n)=n^2 \log n, k=0$ as $T(n) = aT(n/b) + f(n)$

$\log_b a = \log_3 9 = 2 = d$

$f(n) = \Theta(n^{\log_b a} \log^k n)$

$n^2 \log n != \Theta(n^2)$

Comparing $n^{\log_b a} = n^2$ and $f(n) = n^2 \log n$ Does not satisfy either Case 1 or 2 or 3 of the Master's theorem Case 3 states that f(n) should be polynomial larger but here it is asymptotically larger than $n^{\log_b a}$ by a factor of log n .Thus **Master theorem cannot be applied**.

e. $T(n) = 0.5T(n/2) + n$

Here a=0.5,b=2,d=1,f(n)=n as $T(n) = aT(n/b) + f(n)$

Master theorem cannot be applied as the value of a i.e.(no. of sub problems) cannot be less than 1.

f. $T(n) = 2T(n/2) - n$

Here a=2,b=2,d=1,f(n)=-n as $T(n) = aT(n/b) + f(n)$

Master theorem cannot be applied as the value of f(n) cannot be negative.

g. $T(n) = nT(n/2) + n\log n$

Here a=n,b=2,d=1,f(n)=n log n as $T(n) = aT(n/b) + f(n)$

Master theorem cannot be applied as the value a should be constant i.e.(no. of sub problems) should be constant

h. $T(n) = T(n-2) + n^2$

Here a=1,b=n/(n-2),d=1,f(n)=n log n as $T(n) = aT(n/b) + f(n)$

Master theorem cannot be applied as the value b should be constant.

i. $T(n) = T(7n/10) + n$

Here a=1,b=10/7,d=1,f(n)=n as $T(n) = aT(n/b) + f(n)$

$\log_b a = \log_{10/7} 1 = 0 < d$

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

$n = \Theta(n^{0+\varepsilon})$ put $\varepsilon = 1$, then the equality will hold.

$n^0 = \Theta(n^{0+1}) = \Theta(n)$

Thus by case 3:

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

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

j. $T(n) = 4T(n/2) + n^2 \log n$.

This does not fit any of the three cases of Master Theorem. But we can have an upper and lower bound based on Master Theorem. Clearly T (n) ≥ 4T (n) + n 2 and T (n) ≤ 4T (n) + n 2+ϙ for some ϙ > 0. The first recurrence, using the second form of Master theorem gives us a lower bound of Θ(n 2 log n). The second recurrence gives us an upper bound of Θ(n 2+ϙ). The actual bound is not clear from Master theorem. We use a recurrence tree to bound the recurrence.

$T (n) = 4T (n/2) + n^2 \log n$

$= 16T (n/4) + 4(n^2) 2 \log n/2 + n^2 \log n$

$= 16T (n/4) + n^2 \log n/2 + n^2 \log n$

$= . . .$

T (n) = n 2 log n + n 2 log n/2 + n 2 log n/4 + . . . + n 2 log n/(2log n)

= n 2 (log n + log n/2 + log n + 4 + . . .)

= n 2 (log n · n/2 · n/4 + . . . + n/(2logn)) (Transforming logs)

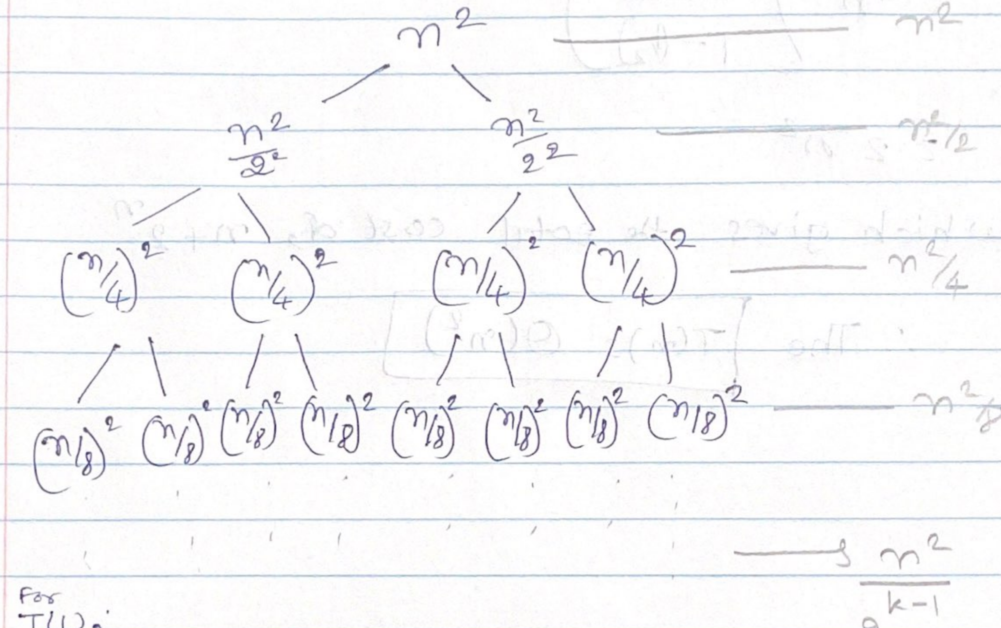
= n 2 (log 2log n) (Using geometric series)

= n 2 log n (Using 2logn = n)

Thus, T (n) = n 2 log n.

4) Use the recursion tree method to solve the recurrence.

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



→ For $T(1)$,

$$T(n/2) = 2T\left(\frac{n}{2^2}\right) + \frac{n^2}{2^2}$$

$$T\left(\frac{n}{2^2}\right) = 2T\left(\frac{n}{2^3}\right) + \frac{n^2}{4^2}$$

$$T\left(\frac{n}{2^k}\right) = T(1)$$

→ $n = 2^k = \log n$

$$2^k = 2^{\log n} \Rightarrow n^{\log 2}$$

which gives,

$$n^2\left[\left(\frac{1}{2}\right)^0+\left(\frac{1}{2}\right)^1+\left(\frac{1}{2}\right)^2+\cdots+\left(\frac{1}{2}\right)^{k-1}\right]$$

$$= n^2\left[\frac{1}{1-(1/2)}\right]$$

$$= 2n^2$$

which gives the total cost of, $n+2^n$

$\therefore$ The $\boxed{T(n) = \Theta(n^2)}$