Abel Marin

Professor Kanj

CSC 321

October 7, 2019

Homework #2

1. Problem 1

a) When T(n) = 2T(n/3)+1 is compared to the master theorem equation aT(n/b) + $O(n^d)$ then a=2>0, b=3>1 and d=0≥0 thus satisfying the master theorem. Next, we compare $c=\log_3 2\approx0.63$ to d=0. Since c>d then T(n) = $O(n^{\log 32})$.

b) When T(n) = 5T(n/4)+n is compared to the master theorem equation aT(n/b) + $O(n^d)$ then a=5>0, b=4>1 and d=1≥0 thus satisfying the master theorem. Next, we compare $c=\log_4 5\approx1.16$ to d=0. Since c>d then T(n) = $O(n^{\log 45})$.

d) When T(n) = 9T(n/3)+n^2 is compared to the master theorem equation aT(n/b) + $O(n^d)$ then a=9>0, b=3>1 and d=2≥0 thus satisfying the master theorem. Next, we compare $c=\log_3 9=2$ to d=2. Since c=d then T(n) = $O(n^2$ * logn).

g) We cannot use the master theorem on this problem since $b=1\not>1$. Therefore we must use the iteration method. Iterations:

1st: T(n) = T(n-1) + 2

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

2nd: T(n) = [T(n-2) + 2] + 2 = T(n-2) +2(2)

T(n-2) = T(n-2-1) + 2 = T(n-3) + 2

3rd: T(n) = [T(n-3) + 2] + 2(2) = T(n-3) +3(2)

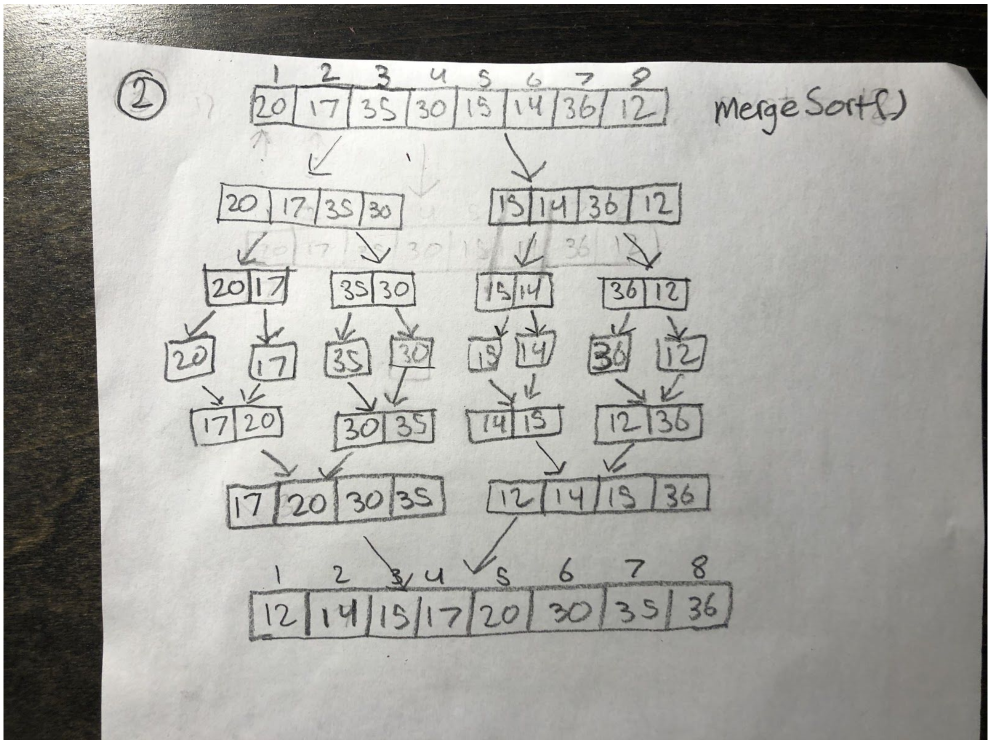
T(n-3) = T(n-3-1) + 2 = T(n-4) + 2

…

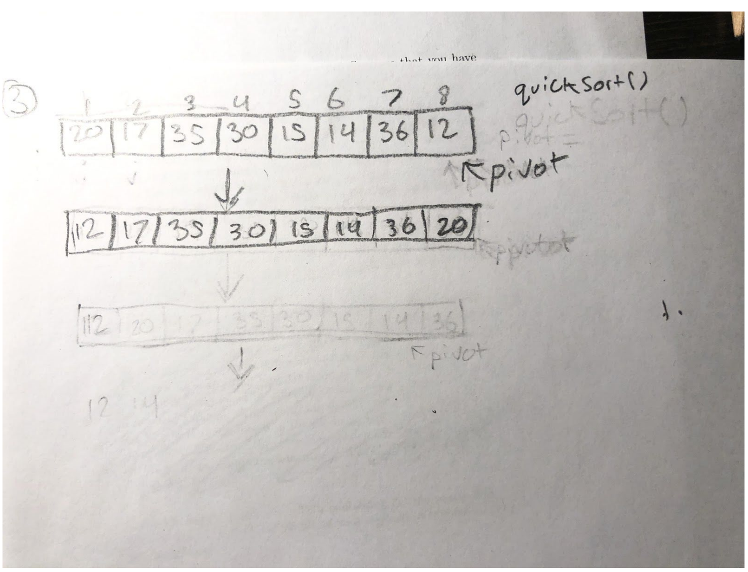
ith: T(n) = T(n-i) +i(2)

Since the relation seems to grow by i(2) for each iteration T(n) = Θ(n).

2. Problem 2



3. Problem 3



4. Problem 4

Psuodocode:

```
insertionRecursive(A, n)
If n > 1 then
return insertionRecursive(A, n-1)
last = A[n-1]
j = n-2
while j >= 0 and A[j] > last do {
    A[j+1] = A[j]
    j = j-1 }
arr[j+1]=last
```

Recursive Relation: T(n) { 1 when n = 1; $T(n-1) + n$ *when* $n > 1$

We cannot use the master theorem on this problem since $b=1 \ngtr 1$. Therefore we must use the iteration method. Iterations:

1st: T(n) = T(n-1) + n

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

2nd: T(n) = [T(n-2) + n-1] + n = T(n-2) +n + n-1

T(n-2) = T(n-2-1) + n-1-1 = T(n-3) + n-2

3rd: T(n) = [T(n-3) + n] + 2(n) = T(n-3) +n + n-1 + n-2

T(n-3) = T(n-3-1) + n-2-1 = T(n-4) + n-3

…

ith: T(n) = $n(n-1)/2+\theta(1)$

Thus the relation is $n(n-1)/2+\theta(1)$ and thus the running time is $O(n^2)$.

5. Problem 5

Pseudocode:

```
median(A, last)

if len(l) % 2 == 1 then

    return median(l, len(l) / 2, pivot_fn)

else
```

```
        return 0.5 * (median(l, len(l) / 2 - 1, pivot_fn)+ median(l, len(l) / 2, pivot_fn))
```

6. Problem 6

For this problem I will use a modified version of Binary Search since this is essentially finding a value in a sorted array. Pseudocode:

```
indexFind(A, first, last)
    if first < last then {
        middle = first + last / 2;
        if middle = A[middle] then return (true);
        if middle > A[middle] then
            return indexFind(A, middle+1, last);
        else return indexFind(A, first, middle-1); }
    else { return (false); }
```

The running time of this algorithm is O(lgn) since it is essentially the same as Binary Search except it checks for the index instead of looking for a specific key. Thus the amount of comparisons would be roughly the same.