ADA Mini Homework 2

(1)

Func: merge(int arr[], int l, int m, int r, int &ans)

L[m - l] = arr[l -> m]

// L contains the elements of arr from l to m

R[r - m] = arr[m + 1 -> r]

// R contains the elements of arr from m + 1 to r

i = 0, j = 0, k = l;

while i < n1 && j < n2

if L[i] <= R[j] :

arr[k] = L[i]

ans += j

// Count numbers that are smaller than L[i] and from the right side

i++

else

arr[k] = R[j];

j++

k++;

while i < n1

arr[k] = L[i]

i++, k++

while j < n2

arr[k] = R[j]

j++, k++

Func: mergeSort(int arr[], int l, int r, int &ans)

if l < r

// Implement merge sort with divide and conquer

m = l + (r - l) / 2;

mergeSort(arr, l, m, ans);

mergeSort(arr, m + 1, r, ans);

merge(arr, l, m, r, ans);

Func main

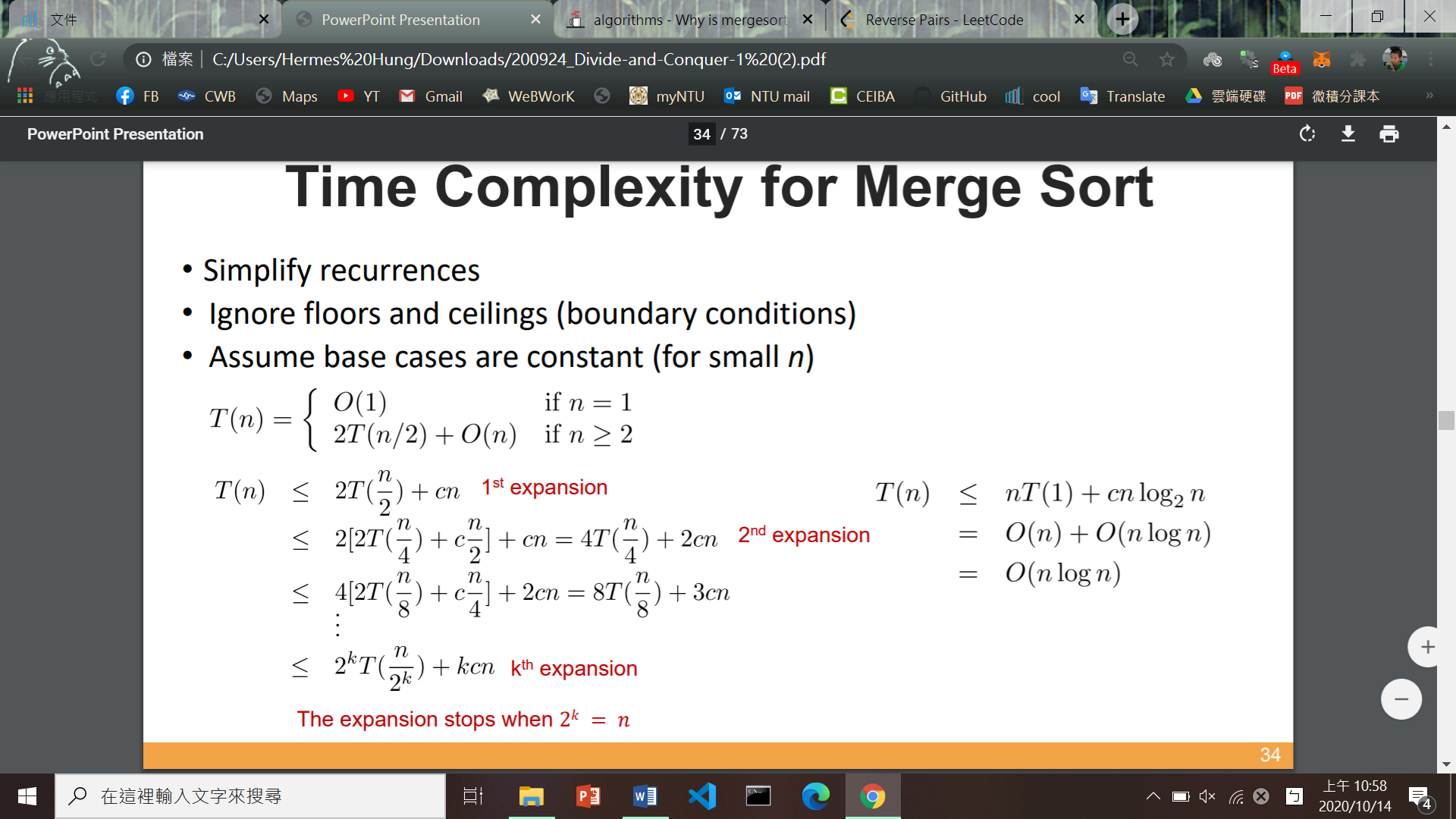
ans = 0

mergeSort(arr, 0, N – 1, ans)

print(ans)

(2)

* Simplify recurrences
* Ignore floors and ceilings (boundary conditions)
* Assume base cases are constant (for small n)

As a result, we can follow the following steps to prove the time complexity.

