日期: /

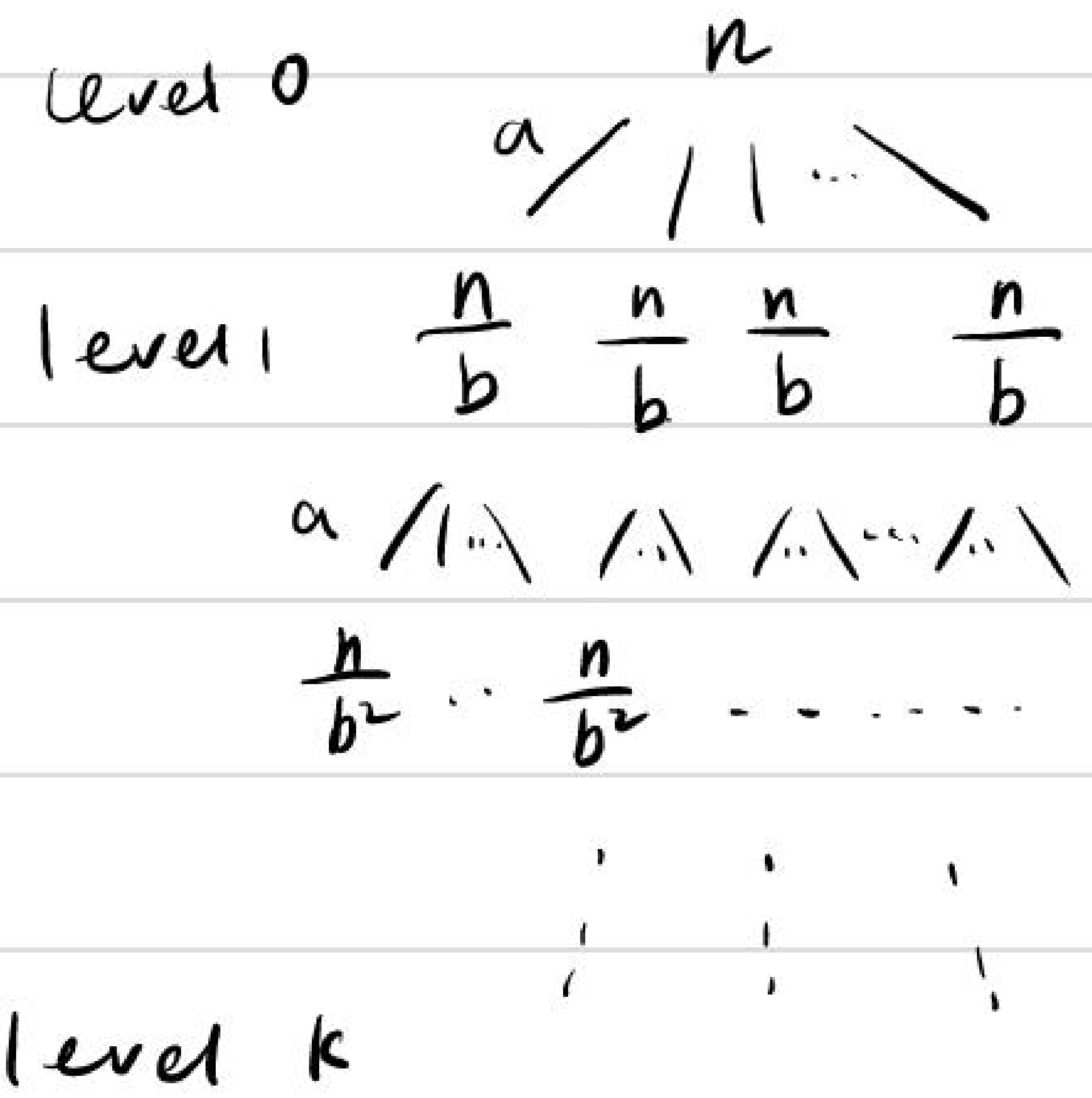
Assignment I for AI2615

王书琦 520030910357

Problem 1. Generalization of Master Theorem

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d \log^w n)$$

$$\rightarrow T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$



$$c a \left(\frac{n}{b}\right)^d \log^w\left(\frac{n}{b}\right)$$

$$c a^2 \left(\frac{n}{b^2}\right)^d \log^w\left(\frac{n}{b^2}\right)$$

$$\vdots$$

$$c \cdot a^k \left(\frac{n}{b^k}\right)^d \log^w\left(\frac{n}{b^k}\right)$$

It reaches the bottom when $k = \log_b n$

Case (A). $a < b^d$

$$T(n) = \sum_{k=0}^{\log_b n} a^k \left(\frac{n}{b^k}\right)^d \log^w\left(\frac{n}{b^k}\right) + a^{\lfloor \log_b n \rfloor + 1}$$

$$= n^d \cdot \sum_{k=0}^{\lfloor \log_b n \rfloor} \left(\frac{a}{b^d}\right)^k \log^w \frac{n}{b^k}$$

$\frac{a}{b^d} < 1$

$$\Rightarrow < n^d \cdot \sum_{k=0}^{\log_b n} \log^w \frac{n}{b^k}$$

$$< n^d \cdot \sum_{k=0}^{\log_b n} lo^w n$$

$$= n^d \cdot \frac{\left(\frac{a}{b^d}\right)^{\lfloor \log_b n \rfloor + 1} - 1}{\frac{a}{b^d} - 1} \leq \frac{n^d \cdot \log^w n}{\frac{a}{b^d} - 1}$$

$$= O(n^d \log^w n)$$

Case (B) $a > b^d$

$$T(n) = \sum_{k=0}^{\log_b n} a^k \left[\frac{n}{b^k}\right]^d \log^w \left[\frac{n}{b^k}\right] + a^{\lfloor \log_b n \rfloor + 1}$$

$$= n^d \cdot \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k \log^w \frac{n}{b^k} + a^{\lfloor \log_b n \rfloor + 1}$$

$$= O(n^{\log_b a})$$

Case (C) $a = b^d$

$$T(n) = \sum_{k=0}^{\log_b n} a^k \left[\frac{n}{b^k}\right]^d \log^w \left[\frac{n}{b^k}\right] +$$

$$= n^d \cdot \sum_{k=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k \log^w \frac{n}{b^k}$$

$$= n^d \sum_{k=0}^{\log_b n} \log^w \frac{n}{b^k}$$

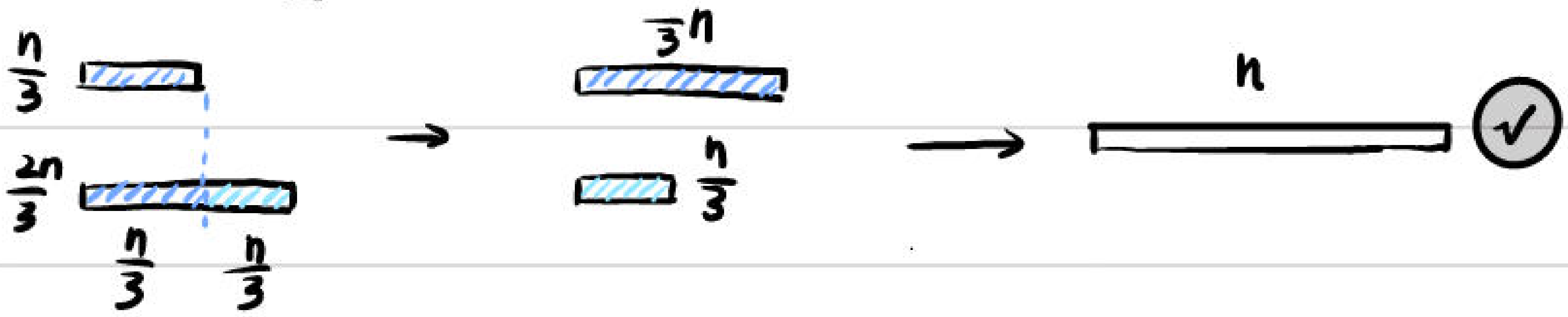
$$\leq n^d \log_b n \cdot \log^w n$$

$$= n^d \log^{w+1} n$$

Problem 2. Merge Sort

(1) Let the rest part of the larger sequence be a new sequence, compare with the sequence just generated. Repeat until no part is left.

e.g. 1/3



(2) Merge Sort by one third dividing approach :

日期:　　　/

```
def Merge_sort (a[1,...,n]) :
    if n==1  return a
    b = M_sort( a[1,... ⌊n/3⌋])
    c = M_sort( a[ ⌊n/3⌋+1], n])
  return Merge (b,c)

def Merge (b[1,...n1] , c[1,...,n2]) :
  {   i=1,  j=1, k=1
      while ( i≤n1 and j≤n2 ) :
        { if ( b[i] < c[i])
              { d[k]= b[i] , i++ , k++;}
            else { d[k] = c[j] , j++, k++ ;}
        }                          i
      if (i>n1) :  # Because ⌊n/3⌋ < ⌊2n/3⌋, operate once is enough
                { b = d
                  c = c[j, ... , n2]
                  d = Merge (b, c)}
      else:     { b = b[j, ... , n2]
                  c = d
                  d = Merge (b, c)}
      return d ;
  }
```

(3) Time complexity $T(n)$

$$T(n) \leq T(\tfrac{n}{3}) + T(\tfrac{2}{3}n) + c \cdot n$$
$$\leq 2T(\tfrac{2}{3}n) + c \cdot n .$$

From The Master Theorem, $T(n) = n\log n$

Problem 3

small → large

ptrA

$A = \{a_1, a_2, \ldots, a_m\}$

$B = \{b_1, b_2, \ldots, b_n\}$

ptrB

1.
```
def count_1D_pairs (A[m], B[n]):
    A = Merge_Sort (A)
    B = Merge_Sort (B)
    int ptrA = 1, ptrB = 1, cnt = 0  # Suppose start from A[1]
    while ptrA != m :
        while (A[ptrA] ≥ B[ptrB]) and (ptrB ≤ n):
            ptrB++
        cnt = cnt + ptrB
        ptrA++
    return cnt
```

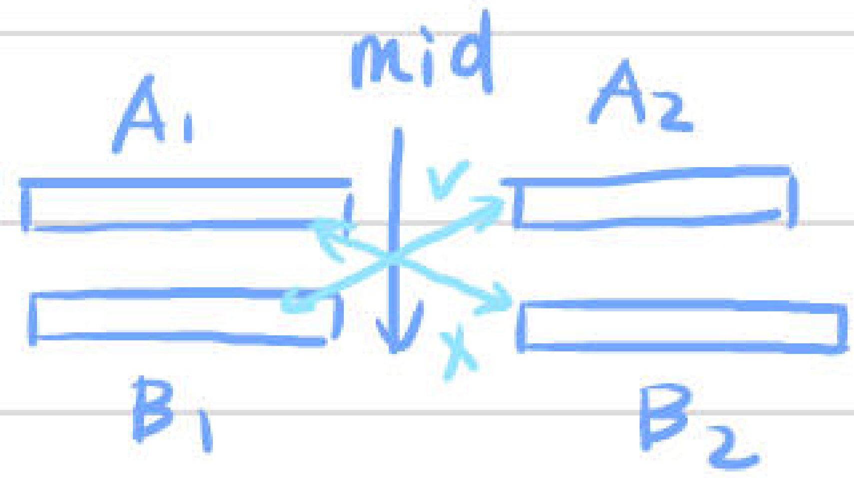
2.
```
def count_2D_pairs (A[m], B[n]):
    if (A == ϕ) or (B == ϕ): return 0
    # Suppose a∈A is O(1) by recording the message when sorting
    S = A∪B, cnt = 0
    S = MergeSort_by_1st_element (S)  # from small to large
    med = S[(m+n)/2]  # To guarantee a close propotion in expectation
```

$A_1 = \{a \in A \mid a[1] < mid\}$

$B_1 = \{b \in B \mid b[1] < mid\}$

$A_2 = \{a \in A \mid a[1] > mid\}$

$B_2 = \{b \in B \mid b[1] > mid\}$



A_recurse = squeeze (A_2) # decrease dimension

B_recurse = squeeze (B_1)

cnt = count_2D_pairs (A_1, B_1) + count_2D_pairs (A_2, B_2)

+ count_1D_pairs (A_recurse , B_recurse)

return cnt

2*. Time complexity

$T(n) = 2T(\frac{n}{2}) + O(n\log n)$

From the Generalization of Master Theorem

$T(n) = O(n\log^2 n) < O(n^{1.1})$

3. def count_dD_pairs (A[m], B[n], d):

if ($A == \phi$) or ($B == \phi$) : return 0

if (d==1) : return count_1D_pairs (A , B)

$S = A \cup B$, cnt = 0

S = MergeSort_by_1st_element (S)

mid = S[(m+n)/2]

$A_1 = \{a \in A \mid a[1] < mid\}$

$B_1 = \{b \in B \mid b[1] < mid\}$

日期: /

$A_2 = \{ a \in A \mid a[1] > mid \}$

$B_2 = \{ b \in B \mid b[1] > mid \}$

```
A_recurse = squeeze (A1) # decrease dimension
B_recurse = squeeze (B2)
cnt = count_dD_pairs (A1, B1, d)
      + count_dD_pairs (A2, B2, d)
      + count_dD_pairs (A_recurse, B_recurse, d-1)
return cnt
```

$$\begin{cases} T(n,d) = 2T(\frac{n}{2}, d) + T(\frac{n}{2}, d-1) \\ T(n,1) = n\log n \end{cases}$$

$\Rightarrow T(n) = n\log^d n$

日期:　　　/

Problem 4

1. x is close to the median of A.

Because at least $\frac{1}{3}$ of the sequence less than x.

And at least $\frac{1}{3}$ of the sequence more than x.

Proof:

A is devided into 3 pieces equally: A_1, A_2, A_3

x_1, x_2, x_3 is corresponding median.

Without loss of generality, we suppose that $x_1 \leq x_2 \leq x_3$

$\therefore x = x_2$

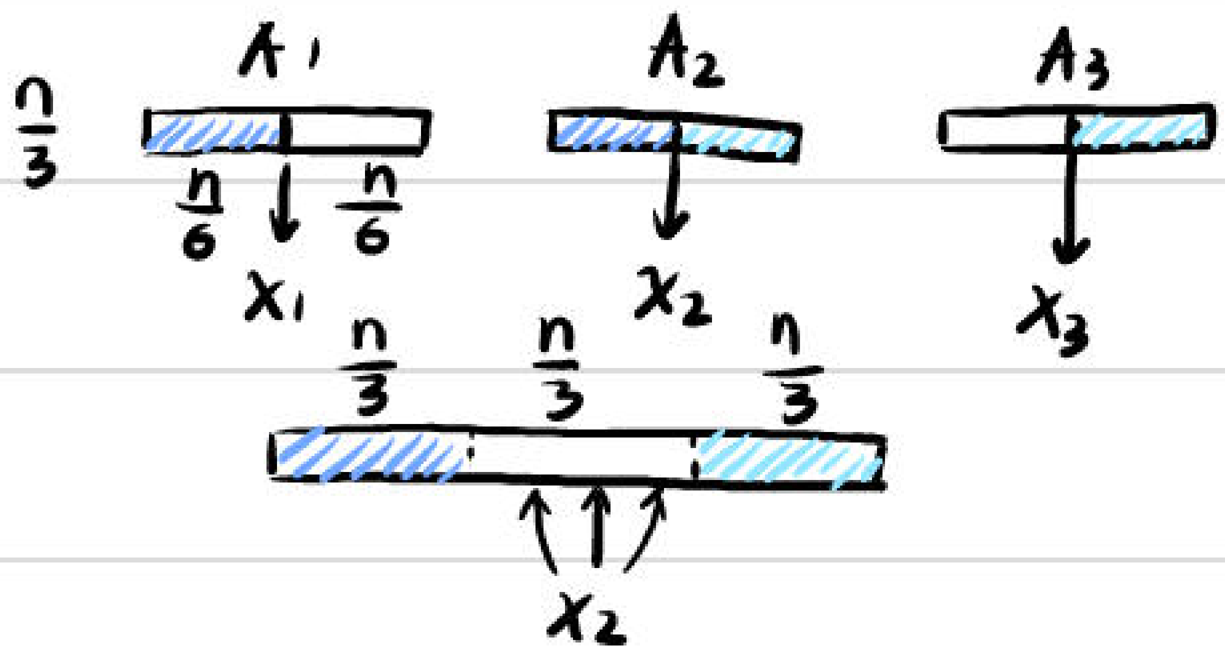
$Card(\{d \in A_1 \mid d < x_1\}) = \lfloor \frac{n}{6} \rfloor$, they also smaller than x.

$Card(\{d \in A_2 \mid d < x\}) = \lfloor \frac{n}{6} \rfloor$.

$\therefore Card(\{d \in A \mid d < x\}) \geq \lfloor \frac{n}{6} \rfloor + \lfloor \frac{n}{6} \rfloor \approx \frac{n}{3}$

Also, $Card(\{d \in A \mid d > x\}) \approx \frac{n}{3}$

To sum up, it can be seen that x is close to the median of A.



2. Runtime $= O(n)$

def median_of_medians (A, k):

$A_1 = A[1, \ldots, \lfloor \frac{n}{3} \rfloor]$

$A_2 = A[\lfloor \frac{n}{3} \rfloor + 1, \ldots, \lfloor \frac{2n}{3} \rfloor]$

日期: /

$A_3 = A[\lfloor \frac{2n}{3} \rfloor + 1, \ldots, n]$

$A_1 =$ merge_sort (A_1)

$A_2 =$ merge_sort (A_2)

$A_3 =$ merge_sort (A_3).

$x_1 = A_1[\mathrm{len}(A_1)/2]$

$x_2 = A_2[\mathrm{len}(A_2)/2]$

$x_3 = A_3[\mathrm{len}(A_3)/2]$

```
# x is the median of x1, x2, x3
x = merge_sort([x1, x2, x3])[3/2]
# use x as the pivot
B = [ ], C = [ ]  # initialization
for i in len(A):
    if A[i] < x :  B.append(A[i])
    if A[i] > x :  C.append(A[i])
if len(B) = k-1 :  return x
if len(B) > k-1 :  return median_of_medians(B, k)
if len(B) < k-1 :  return median_of_medians(C, k-len(B)-1)
```

Runtime Analysis

Sorting very small list takes linear time :

$T(n) \leq T(\frac{n}{3}) + O(n) \leq O(n)$

日期:　　/

Problem 5

1. 10h (at least)

2.

Problem	1	2	3	4	5
Difficulty	4	3	5	3	1

3. Discussed with Xinyan Chen, Qi Chen.
And get inspirations from TA Zichen Zhu, Yihang Qiu.