|  |  |  |
| --- | --- | --- |
|  | **SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**  **AUGUST 2021 SEMESTER**  **TAKE HOME FINAL EXAMINATION** | |
| **Subject Code : BIT208**  **Subject Name : DATA STRUCTURES AND ALGORITHMS** | | |
| **This examination carries 60% of the total assessment for this subject.** | | |
| Examiner(s) **DR ABDUL QAYOOM HAMAL** | |  |
| **Duration: 4 HOURS** | | |
| **Start Time:   Day: TUESDAY**  **Date: 30 NOVEMBER 2021  Release Time: 9:00 AM** | | **Due Date and Time:   Day: TUESDAY  Date: 30 NOVEMBER 2021  Due Time: 1:00 PM** |
| **Declaration**   * I have read and understood the [Academic Integrity Policy](https://lms.help.edu.my/helpelearning/pluginfile.php/90637/mod_resource/content/1/Academic%20Integrity%20Policy%20%28FCDT%29.pdf) that explains on **plagiarism**, and I testify that, unless otherwise acknowledged, the work submitted herein is entirely my own. * I declare that no part of this examination has been written for me/ by any other person(s) * I authorize the University to test any work submitted by me using text comparison software, for instances of plagiarism. I understand this will involve the University or its contractors copying my/our work and storing it on a database to be used in future to test work submitted by others.   Note: 1) The attachment of this statement on any electronically submitted examinations will be deemed to have the same authority as a signed statement.  **Student Name: Shogo Terashima Student ID: B2001046** | | |
| Instructions: 1. Download the exam paper from the e-learning course/subject folder at the stipulated start time.  2. Answer ALL the exam questions in Microsoft Word using the Cover Sheet attached as the first page.  3. You are not allowed to discuss the exam questions or answers with anyone, including your lecturers, friends, classmates or any other parties.  4. Label your answers clearly.  5. Save your Word document at regular intervals to avoid losing any data.  6. Name your Word document with your StudentID followed by subject code (e.g.B1223432BIT208.doc)  7. It is your responsibility to keep track of the time.  8. Once you have completed your exam submit your Word document to LMS Turnitin.  9. You are only allowed to do ONE submission to the LMS Turnitin.  10. The duration of the exam is 4 hours. Therefore, you will need to submit your answers to Turnitin within this time frame. You are advised to submit earlier than the actual end time to avoid high traffic on the submission sites.  11. Any submissions after the stipulated exam end time will be considered as a NON-SUBMISSION.  12. Once the answer is submitted for grading, NO requests for amendments or submission of additional documents will be permitted or accepted.  13. In case of any questions about the examination you may post questions on the GENERAL Teams channel for your subject.  14. In case of any technical problems e.g., internet connection or computer malfunction, you must immediately contact your lecturer via email or Teams with screenshots of the issues.  15. You are required to adhere to the [Academic Integrity Policy](https://lms.help.edu.my/helpelearning/pluginfile.php/90637/mod_resource/content/1/Academic%20Integrity%20Policy%20%28FCDT%29.pdf). Students found with high similarities to each other or other sources via Turnitin will receive penalties and be investigated for academic misconduct. | | |
|  | | |

**Question 1**

**a)**

Let T(n) as running time. (n is problem size)

T(100) = c (100)3 = 15 sec (c is unit of running time ) …. eq1

Let x as size of problem I want to get.

T(x) = c x3 = 2 min = 120 sec ….eq2

eq2 / eq1

x3 / (100)3 = 15 / 120 = 8

∴ x3 = (2 \* 100 )3

∴ x = 200 (∵ x > 0)

**b)**

ⅰ f(n) = 5n2 – 4n+ 3 is Ω(n2)

f(n) >= c g(n) for c>0, n>= n0

5n2 – 4n+ 3 > = 5n2 -4n2 + 3 where n >= 1

>= n2

c = 1 n0 = 1

ⅱ f(n) = 5n2 – 4n- 3 is Ω(n2)

f(n) >= c g(n) for c>0, n>= n0

5n2 – 4n- 3 > = 5n2 -4 n2 where n >= 2

>= n2

c = 1 n0 = 2

ⅲ. f(n) = 2n+100log n

f(n)<= c g(n) for c>0, n>= n0

2n+100log n <= 2n+ n n>2

= 3n

C = 3 , n0 = 2

ⅳ f(n) = 5n2 – 4n – 3

5n2 – 4n – 3 < = 5n2 n>= 1

C = 5 , n0 = 1

**c)**

ⅰ

first = array[0] … 2ops

middle = array[size /2] … 3 ops

last = array[size-1]…3ops

sum = first + middle + last … 3 ops

So, this case, big-oh will be constant. O(11)

ⅱthis case, largest element is the last element of the array.

So, it need to traverse one by one. So, O(n)

ⅲ. This case, the element want to search is not in the sorted linked list.

Traverse from head to tail. So, it depends on the number of elements. O(n)

ⅳ. This case, the element want to search is not in the array.

Traverse all element then you can find that the element is not in the list. So, it depends on the number of elements. O(n)

**d)**

ⅰ Add: Create new node => set next of new node to head => set head to the new node.

Delete : Set head to head.next

So, the big oh will be constant.

ⅱ Add: Create new node => set next of tail to new node=> set tail to the new node.

Delete : traverse head to before tail which is size -2 (index start from 0) =>set tail to new last element. Set next of tail to null.

So, big oh will be constant for addition, but for delete, it will be O(n) depends on number of element in the list.

ⅲ This case, it need to traverse from head to particular element. So, big oh will be O(n) it depends of index of particular element.

**Question 2**

public static void displayInSeries(SLinkedList<Integer> list) {

Node temp = list.getHead();

NodeQueue<Integer> queue = new NodeQueue<Integer>();

int element = 0;

int first = 0;

int last = 0;

System.out.println("First Line:");

while(temp != null) {

// this is for first line

element = (int)temp.getElement();

last = element % 10; // last digit

first = (element / 10) % 10; // first difit

if( (first + last) % 3 == 0){

System.out.print(temp.getElement() + " "); // first line

}else {

// this is for second line

// enqueue to the list

queue.enqueue(element);

}

temp = temp.getNext();// get next

}

// show second line

System.out.println("\nSecond Series:");

Integer temp2 = queue.dequeue();

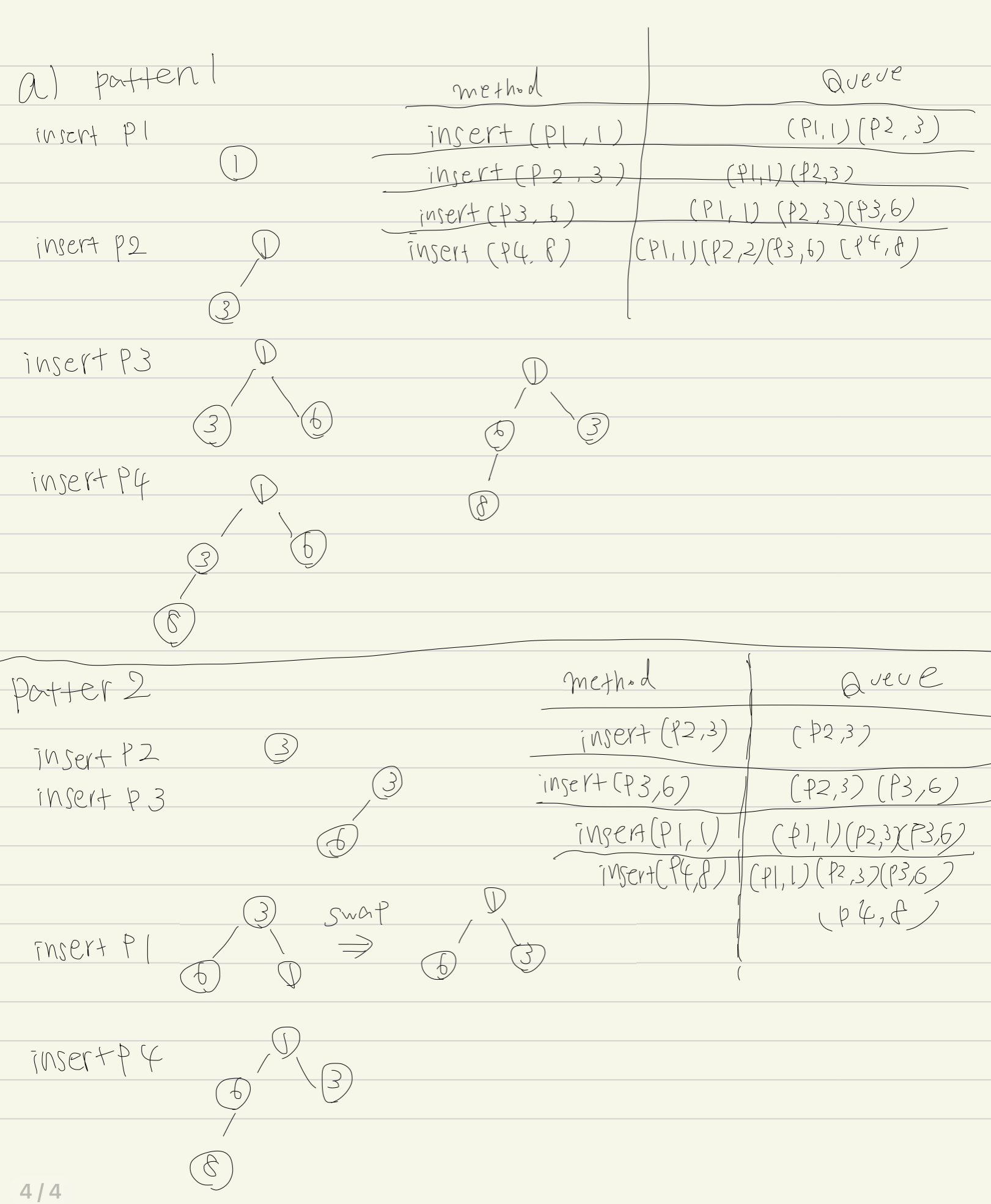
while(temp2 != null){

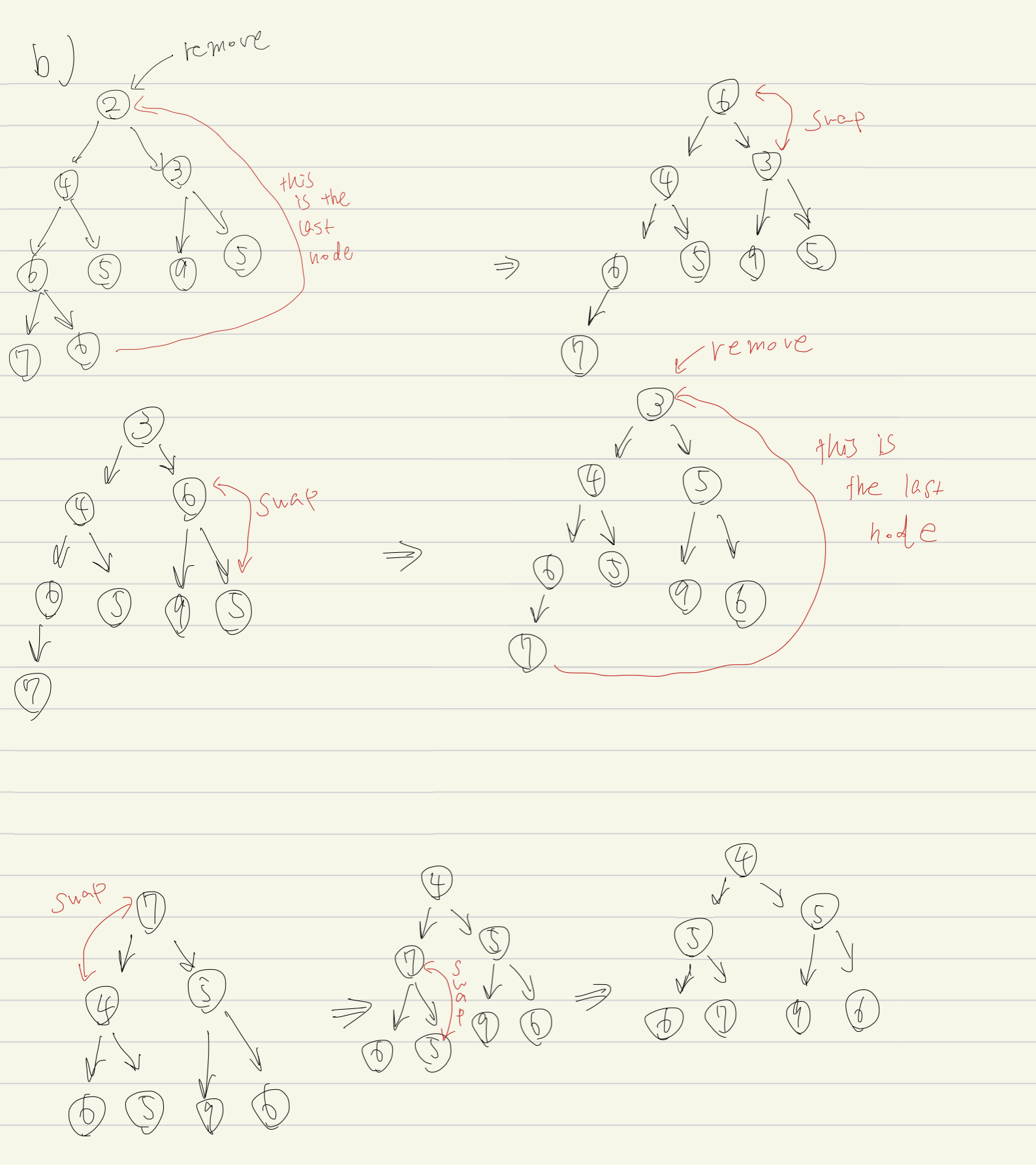
System.out.print(temp2 + " ");

temp2 = queue.dequeue(); // deququq while not null

}

}

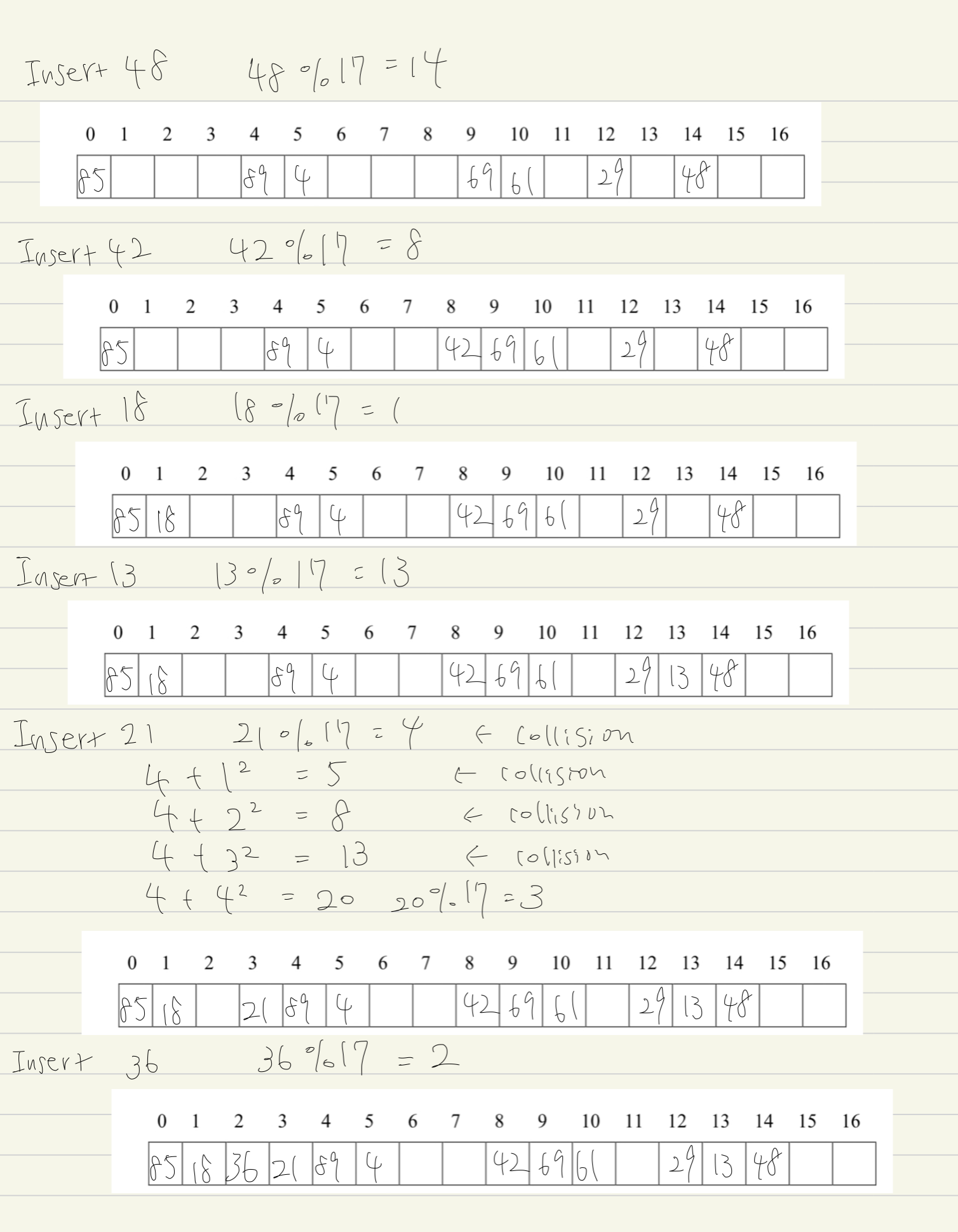
**Question 3**

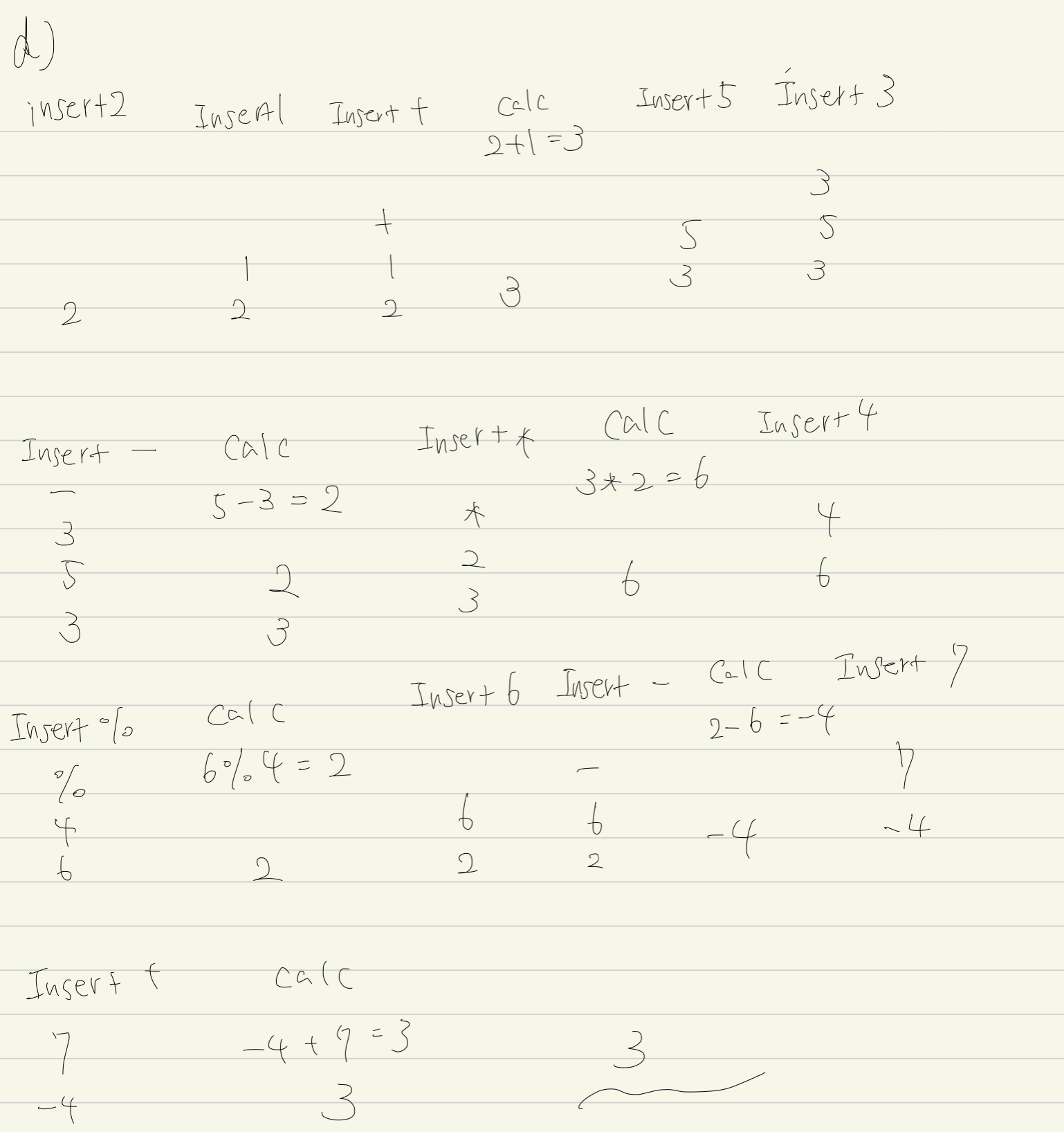


ダイアグラム

自動的に生成された説明

テーブル

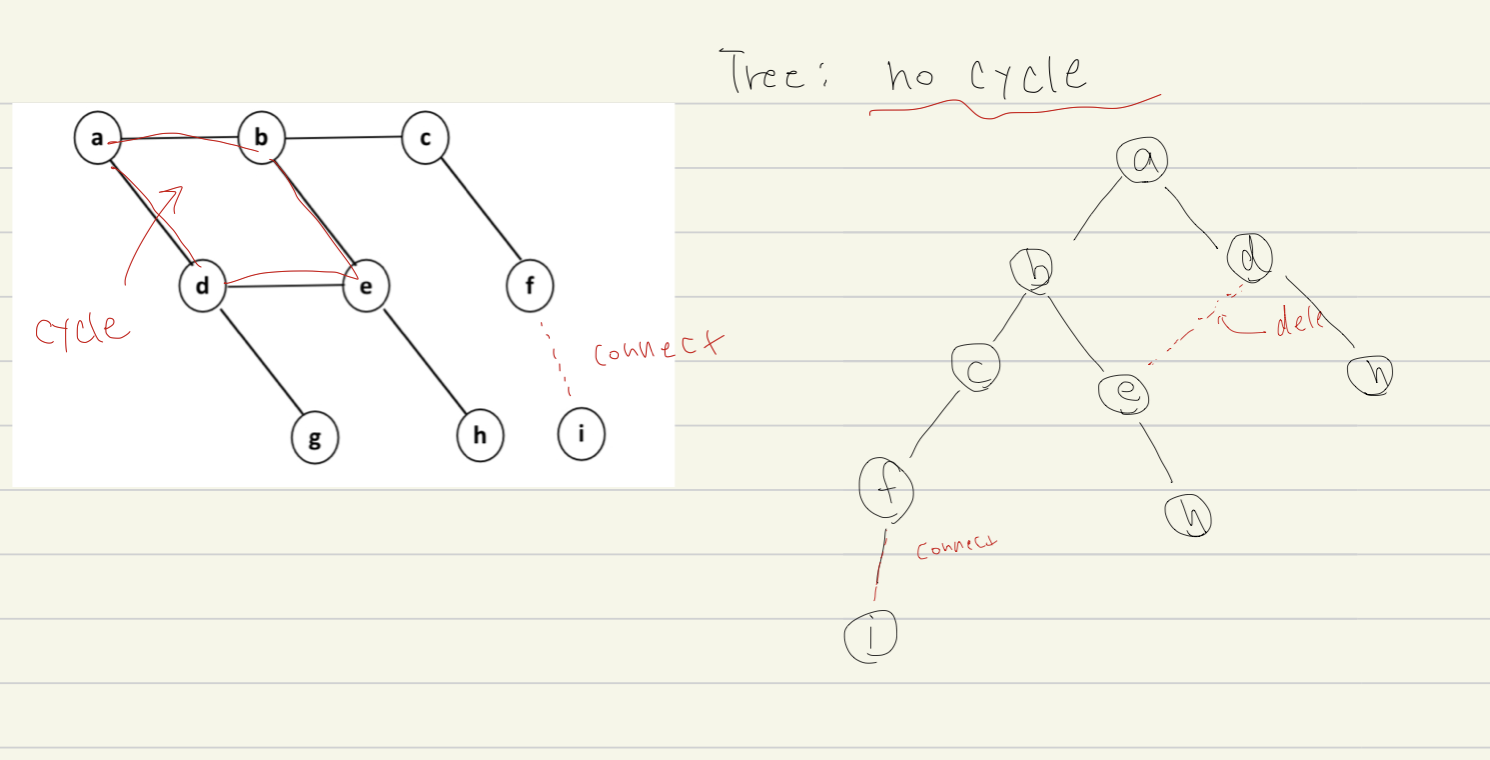
自動的に生成された説明



**Question 4**

ⅰThis is graph is not connected because vertex I is isolated.

ⅱ vertex d is connected to a,e,and g so, degree of vertex d is 3.

ⅲ 

**b)**

public boolean checkFull(BNode root){

//check if root is null empty

if(root==null)

return true;

// if it is leaf node, return true

if(root.hasLeftChild() ==false && root.hasRightChild()==false)

return true;

//check if node has only one child

if((root.hasLeft() && root.hasRight() ==false )||( root.hasLeft()=false && root.hasRight()))

return false;

//recursively check left and right child

boolean result = checkFull(root.getLeftChild()) && checkFull(root.getRightChild());

return result;

}

**Question 5**

ⅰInserttion Sort

パソコンの画面

中程度の精度で自動的に生成された説明Insertion Sort (Basic Insertion Method) is a method of taking values one by one from an "unaligned array" and inserting them into the appropriate position in an "aligned array".

If you want to put an element in the right place in the aligned place, you have to move it to the right one by one to open the place.

Worst Case is when the input is sorted in reverse order. Need to traverse and check all aligned columns. So, 𝑂 (n) \* n times. => 𝑂(n2)

Best Case is when the input is already sorted. Only need traverse once.

So, 𝑂 (n)

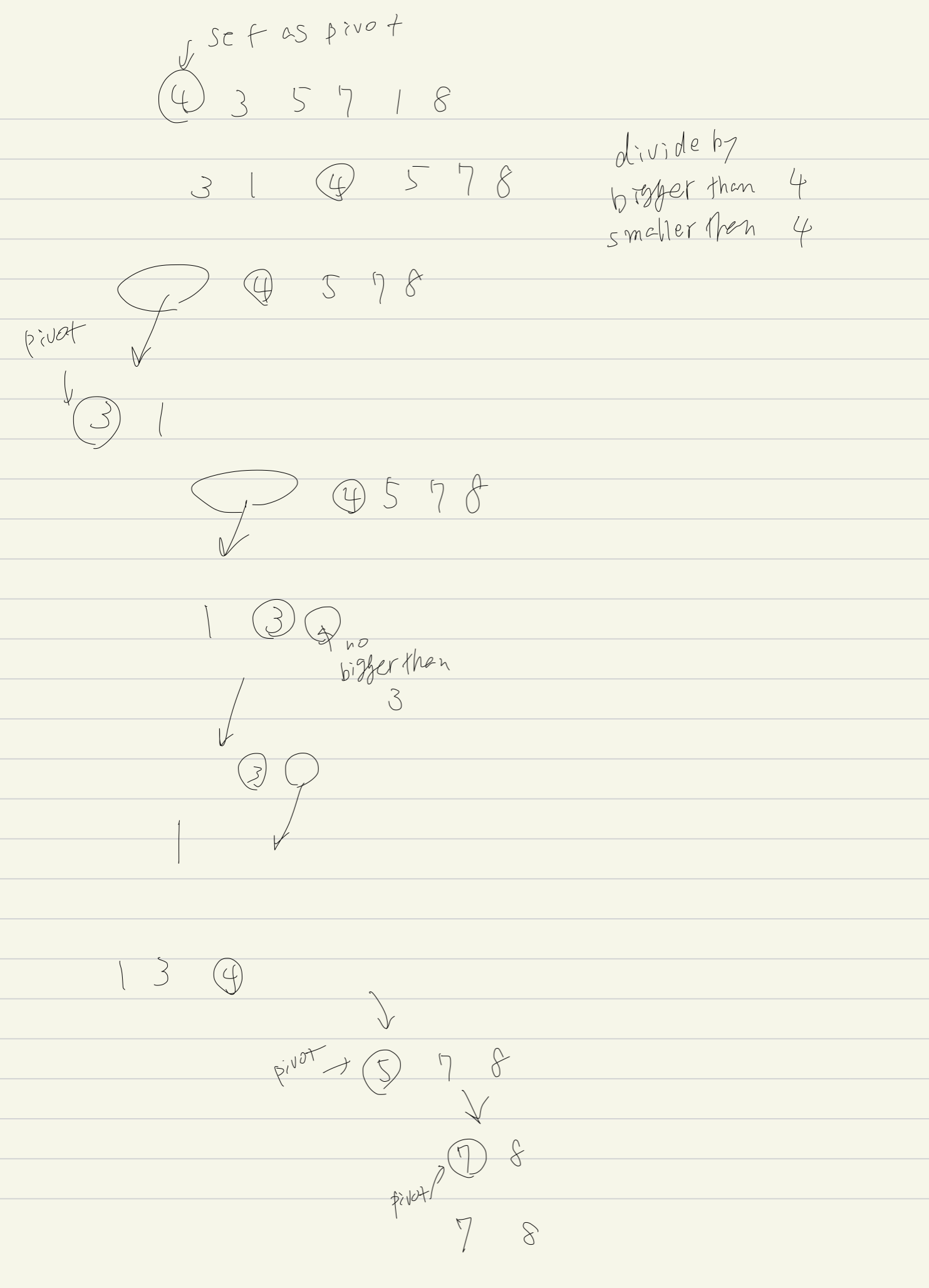
At some point, there are m unsorted data, and m increases from 1 to (n-1), so the number of comparisons is as follows

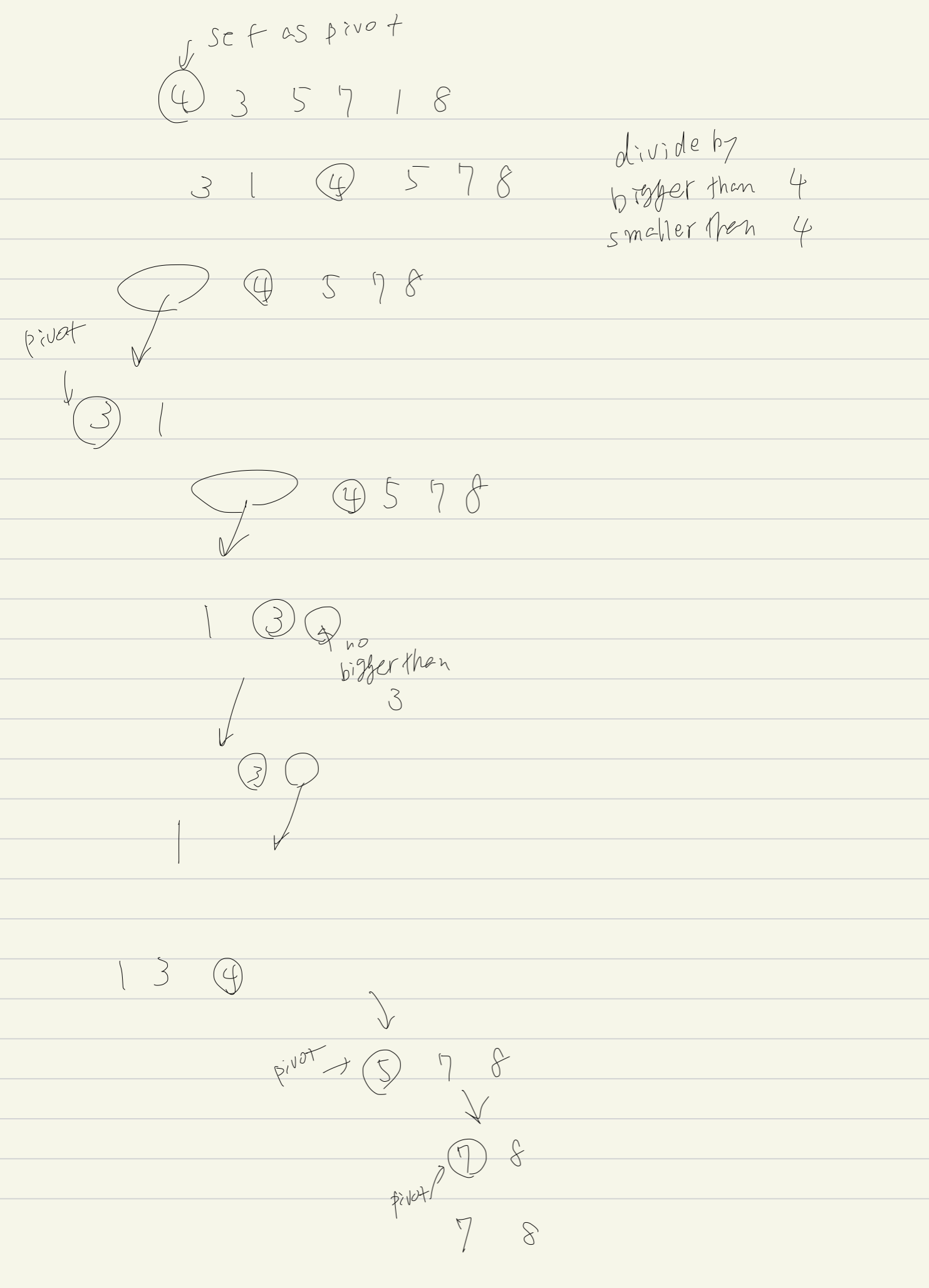
グラフ

中程度の精度で自動的に生成された説明

So, average complexity is 𝑂(n2)

ⅱ Quick Sort

divide-and-conquer alignment algorithm that recursively divides data into those above a certain value (the value of an axis element) and those below (or below).



Worst Case is when the input is sorted in reverse order.

Tn=n−1+Tn−1(n≥2)Tn=n−1+Tn−1(n≥2)

Tn = n−1+n−2+・・・+1(∵T1=0)

= n \* (n-1) / 2

𝑂(n2)

Best case is when the division proceeds evenly.

If it can divide it in half each time, it can divide it to 1 element in 𝑛log n times. 𝑂(n logn)

Average case is 𝑂(n logn)

The condition is that every division pattern occurs with equal probability in all division processes. With respect to the number of comparisons, we can formulate an incremental formula

テキスト, 概略図

自動的に生成された説明

Solve this, and we will get 𝑂(n logn)

Compared to both average case. 𝑂(n2) and 𝑂(n logn), quick sort is better performance.

But, insertion sort will be faster when the array is almost sorted.