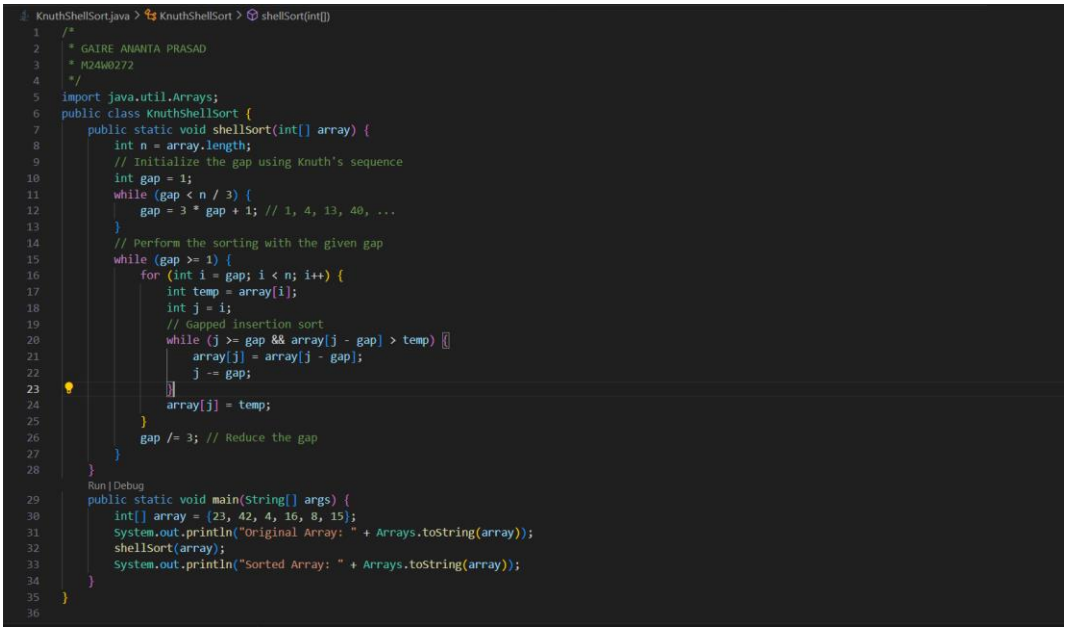
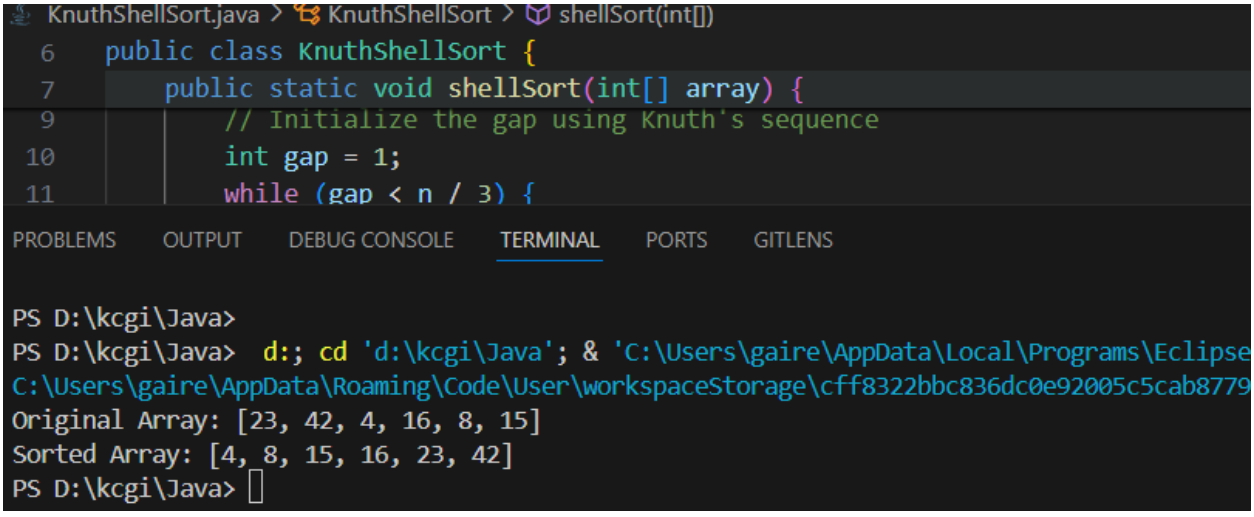
GAIRE ANANTA PRASAD
M24W0272
Data Insertion Interval

1. Knuth Shell Sort
   Java Code



```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */
import java.util.Arrays;
public class KnuthShellSort {
    public static void shellSort(int[] array) {
        int n = array.length;
        // Initialize the gap using Knuth's sequence
        int gap = 1;
        while (gap < n / 3) {
            gap = 3 * gap + 1; // 1, 4, 13, 40, ...
        }
        // Perform the sorting with the given gap
        while (gap >= 1) {
            for (int i = gap; i < n; i++) {
                int temp = array[i];
                int j = i;
                // Gapped insertion sort
                while (j >= gap && array[j - gap] > temp) {
                    array[j] = array[j - gap];
                    j -= gap;
                }
                array[j] = temp;
            }
            gap /= 3; // Reduce the gap
        }
    }
    Run | Debug
    public static void main(String[] args) {
        int[] array = {23, 42, 4, 16, 8, 15};
        System.out.println("Original Array: " + Arrays.toString(array));
        shellSort(array);
        System.out.println("Sorted Array: " + Arrays.toString(array));
    }
}
```

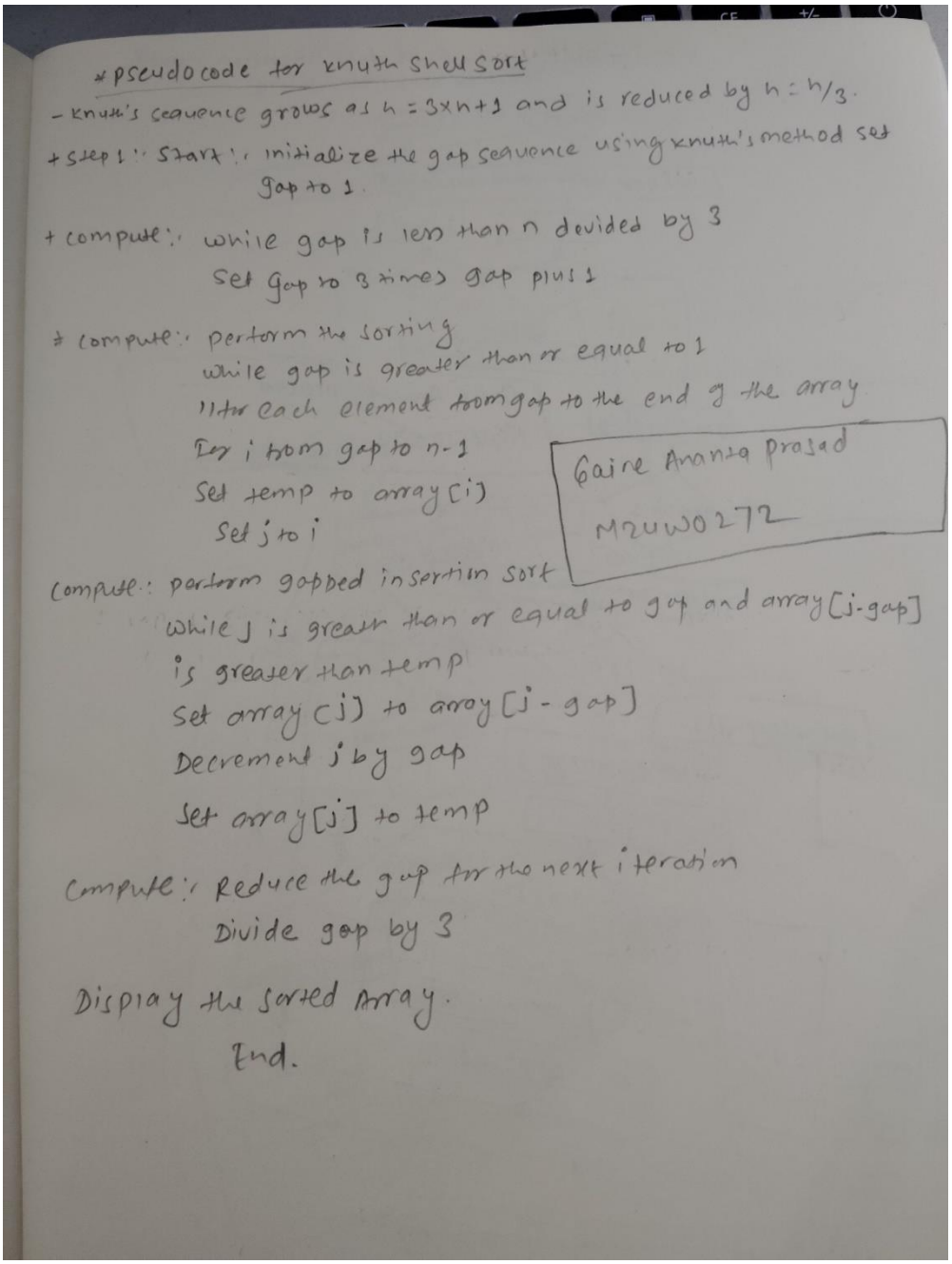
Output



```
KnuthShellSort.java > KnuthShellSort > shellSort(int[])
  6   public class KnuthShellSort {
  7       public static void shellSort(int[] array) {
  9           // Initialize the gap using Knuth's sequence
 10           int gap = 1;
 11           while (gap < n / 3) {

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS D:\kcgi\Java>
PS D:\kcgi\Java>  d:; cd 'd:\kcgi\Java'; & 'C:\Users\gaire\AppData\Local\Programs\Eclipse
C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cff8322bbc836dc0e92005c5cab8779
Original Array: [23, 42, 4, 16, 8, 15]
Sorted Array: [4, 8, 15, 16, 23, 42]
PS D:\kcgi\Java>
```

GAIRE ANANTA PRASAD
M24W0272

Pseudocode



* pseudocode for knuth shell sort

- knuth's seauence grows as h = 3xh+1 and is reduced by h = h/3.

+ step 1 : start : initialize the gap seauence using knuth's method set gap to 1.

+ compute : while gap is len than n devided by 3
set gap to 3 times gap plus 1

compute : perform the sorting
while gap is greater than or equal to 1
//for each element from gap to the end of the array
For i from gap to n-1
Set temp to array[i]
Set j to i

Gaire Ananta prasad
M24W0272

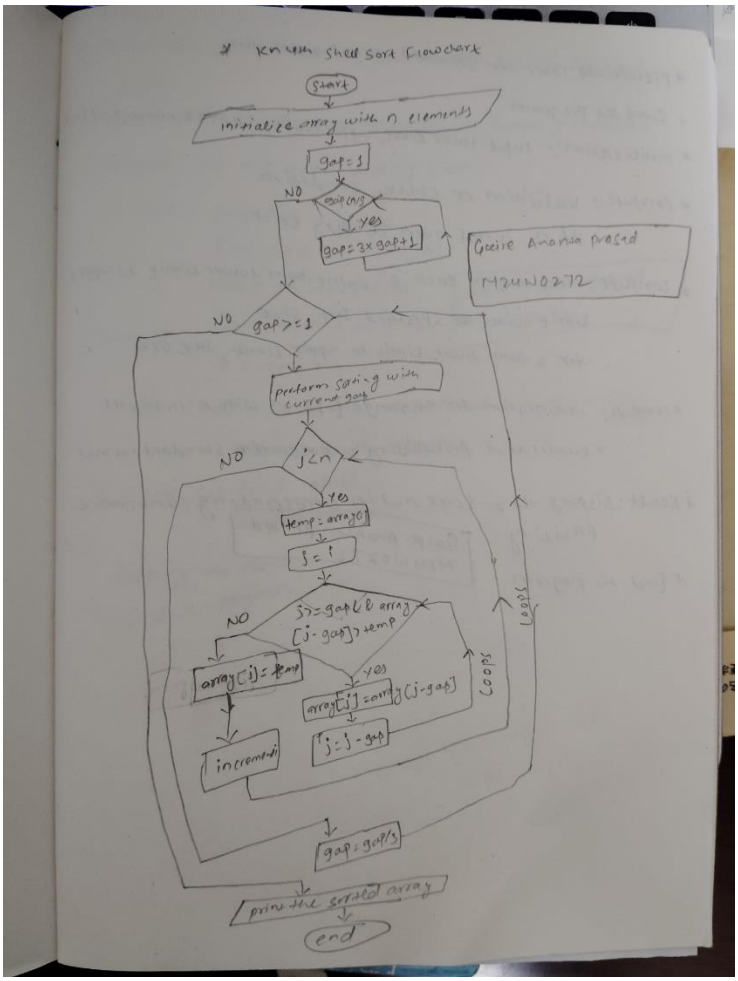
compute : perform gapped insertion sort
while j is greater than or equal to gap and array[j-gap]
is greater than temp
Set array[j] to array[j - gap]
Decrement j by gap
Set array[j] to temp

compute : Reduce the gap for the next iteration
Divide gap by 3

Display the sorted Array.
End.

GAIRE ANANTA PRASAD
M24W0272

Flow-Chart



GAIRE ANANTA PRASAD
M24W0272

2. Hibbard Shell Sort

   Java Code

```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */
import java.util.Arrays;
public class HibbardShellSort {
    public static void shellSort(int[] array) {
        int n = array.length;
        // Initialize gap using Hibbard's sequence
        int gap = 1;
        while (gap < n) {
            gap = 2 * gap + 1; // 1, 3, 7, 15, 31, ...
        }
        gap = (gap - 1) / 2;
        // Perform the sorting with the given gap
        while (gap >= 1) {
            for (int i = gap; i < n; i++) {
                int temp = array[i];
                int j = i;
                // Gapped insertion sort
                while (j >= gap && array[j - gap] > temp) {
                    array[j] = array[j - gap];
                    j -= gap;
                }
                array[j] = temp;
            }
            gap = (gap - 1) / 2; // Reduce the gap
        }
    }
    Run | Debug
    public static void main(String[] args) {
        int[] array = {23, 42, 4, 16, 8, 15};
        System.out.println("Original Array: " + Arrays.toString(array));
        shellSort(array);
        System.out.println("Sorted Array: " + Arrays.toString(array));
    }
}
```

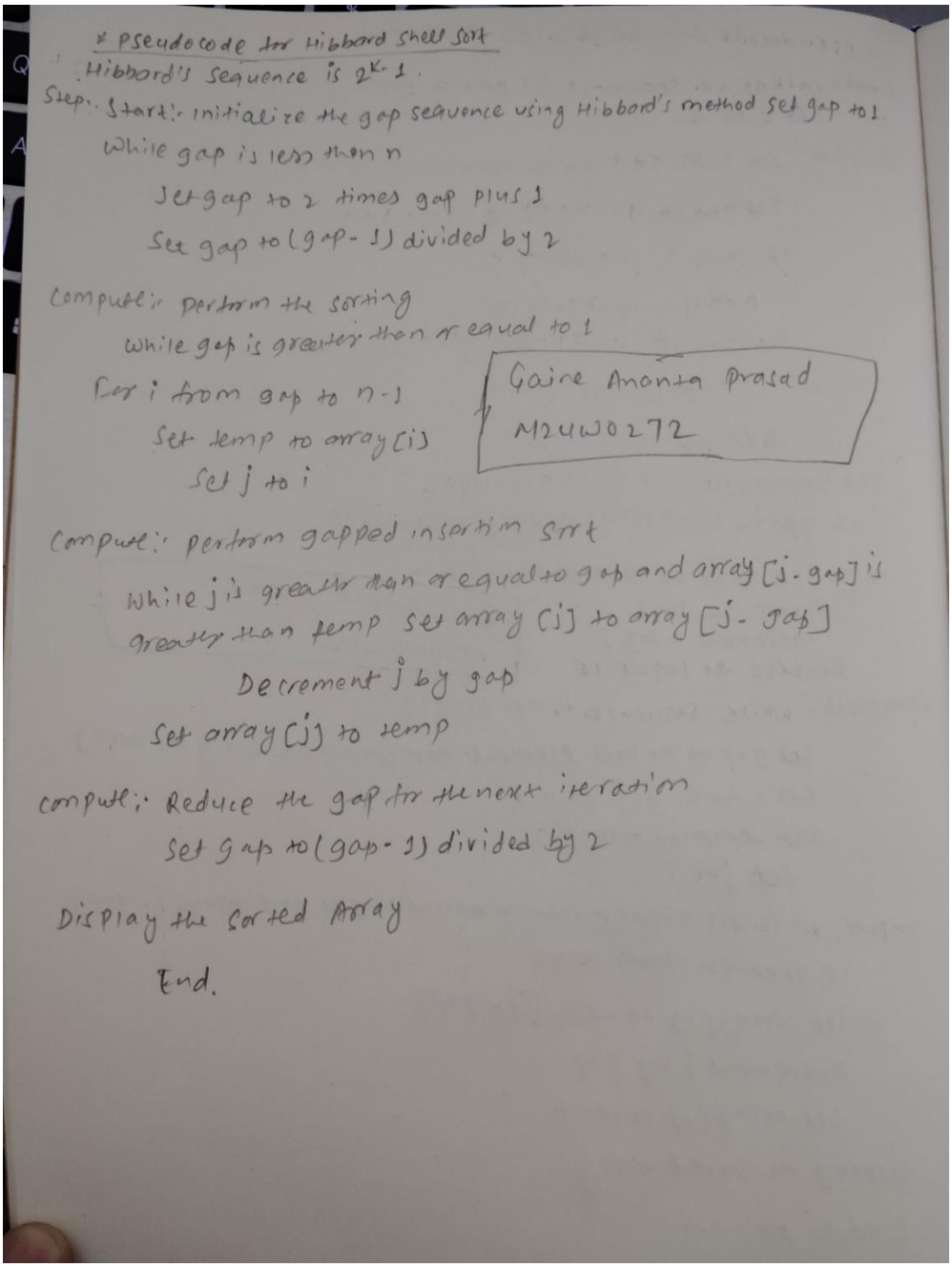
Output

```
HibbardShellSort.java > HibbardShellSort > shellSort(int[])
1   import java.util.Arrays;
2   public class HibbardShellSort {
3       public static void shellSort(int[] array) {
4           int n = array.length;
5           // Initialize gap using Hibbard's sequence
6           int gap = 1;
7           while (gap < n) {
8               gap = 2 * gap + 1; // 1, 3, 7, 15, 31, ...
9           }

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS D:\kcgi\Java>  & 'C:\Users\gaire\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-XX:+S
'C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cff8322bbc836dc0e92005c5cab87794\redhat.java\jdt_ws\Java_c3b
Original Array: [23, 42, 4, 16, 8, 15]
Sorted Array: [4, 8, 15, 16, 23, 42]
PS D:\kcgi\Java>
```

GAIRE ANANTA PRASAD
M24W0272

GAIRE ANANTA PRASAD
M24W0272

Pseudocode



* Pseudocode for Hibbard shell sort

Hibbard's sequence is $2^K - 1$.

Step: Start: Initialize the gap sequence using Hibbard's method set gap to 1

While gap is less than n

Set gap to 2 times gap plus 1

Set gap to (gap - 1) divided by 2

Compute: perform the sorting

While gap is greater than or equal to 1

For i from gap to n-1

Set temp to array[i]

Set j to i

Gaire Ananta Prasad

M24W0272

Compute: perform gapped insertion sort

While j is greater than or equal to gap and array[j - gap] is greater than temp set array[j] to array[j - gap]

Decrement j by gap

Set array[j] to temp

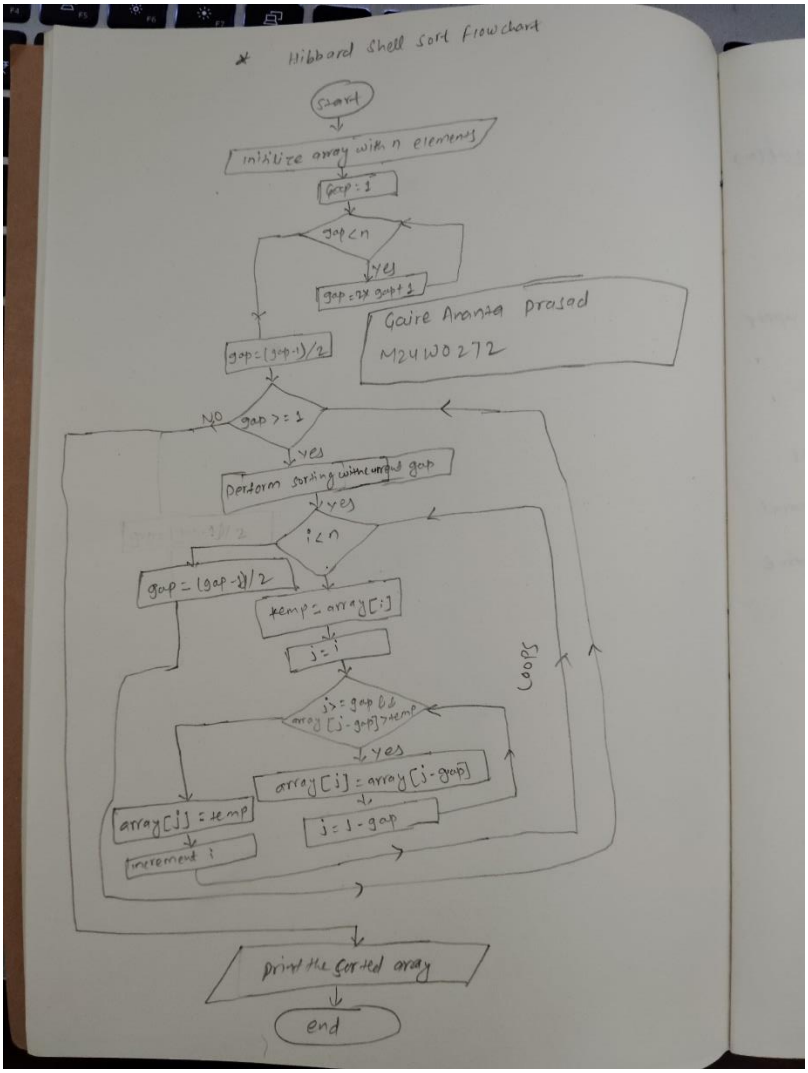
compute: Reduce the gap for the next iteration

Set gap to (gap - 1) divided by 2

Display the sorted Array

End.

Flow-Chart



GAIRE ANANTA PRASAD
M24W0272

3. Sedgewick Shell Sort

Java Code

SedgewickShellSort.java > ...

```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */
import java.util.Arrays;
import java.util.ArrayList;
import java.util.Collections;
public class SedgewickShellSort {
    public static void shellSort(int[] array) {
        int n = array.length;
        // Generate Sedgewick sequence
        ArrayList<Integer> sequence = generateSedgewickSequence(n);
        // Perform the sorting with the given gaps
        for (int gap : sequence) {
            for (int i = gap; i < n; i++) {
                int temp = array[i];
                int j = i;
                // Gapped insertion sort
                while (j >= gap && array[j - gap] > temp) {
                    array[j] = array[j - gap];
                    j -= gap;
                }
                array[j] = temp;
            }
        }
    }
    private static ArrayList<Integer> generateSedgewickSequence(int n) {
        ArrayList<Integer> sequence = new ArrayList<>();
        int k = 0;
        int gap;
        // Use Sedgewick's sequence: 9*4^k - 9*2^k + 1 and 4^k + 3*2^(k-1) + 1
        while (true) {
            gap = (int)(9 * Math.pow(a:4, k) - 9 * Math.pow(a:2, k) + 1);
            if (gap < n) {
                sequence.add(gap);
            }
            gap = (int)(Math.pow(a:4, k) + 3 * Math.pow(a:2, k - 1) + 1);
```

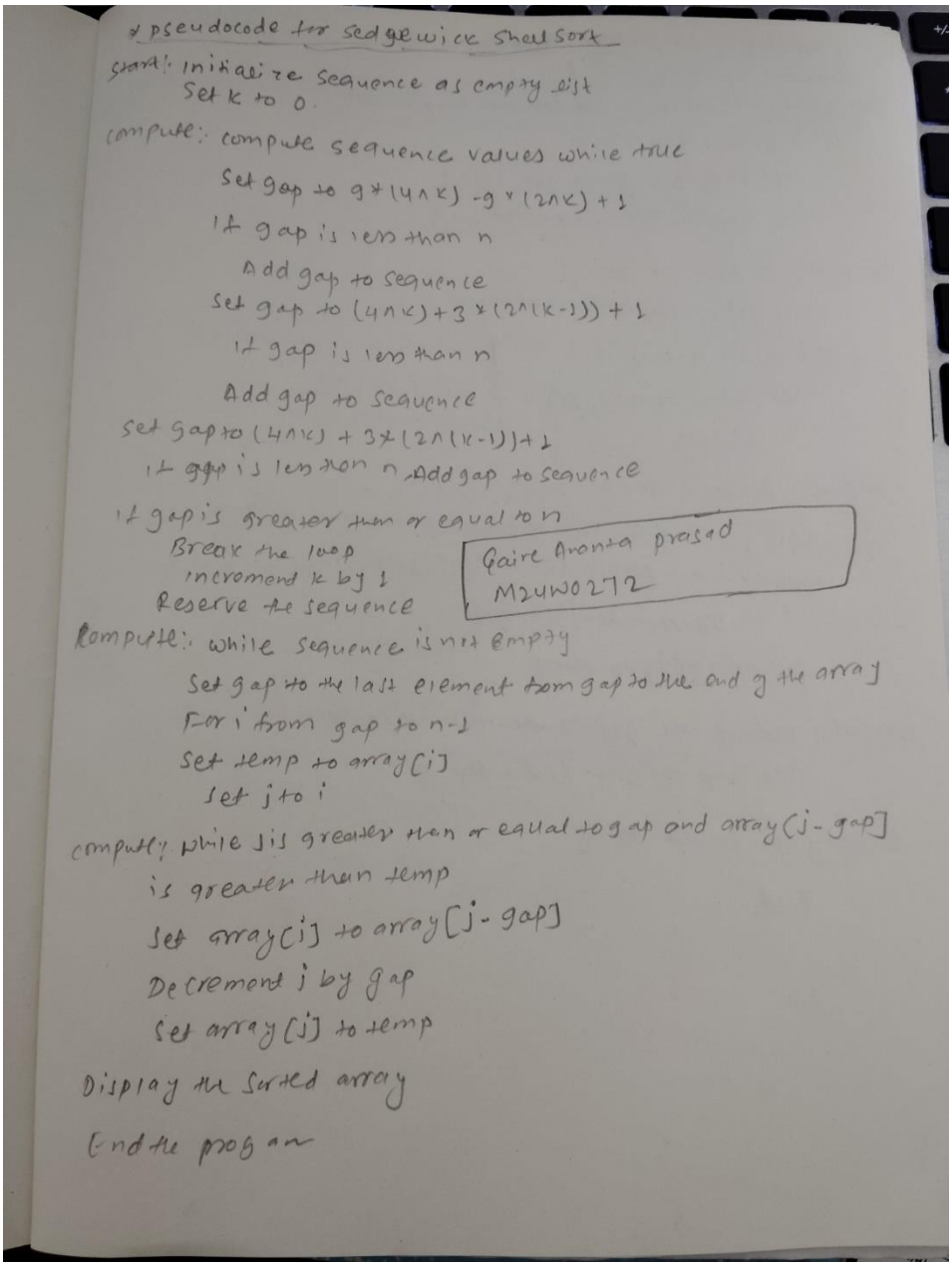
Output

SedgewickShellSort.java > SedgewickShellSort > generateSedgewickSequence(int)

```java
public class SedgewickShellSort {
    private static ArrayList<Integer> generateSedgewickSequence(int n) {
            gap = (int)(9 * Math.pow(a:4, k) - 9 * Math.pow(a:2, k) + 1);
            if (gap < n) {
                sequence.add(gap);
            }
            gap = (int)(Math.pow(a:4, k) + 3 * Math.pow(a:2, k - 1) + 1);
            if (gap < n) {
                sequence.add(gap);
            }
            if (gap >= n) {
                break;
            }
            k++;        }
        // Reverse the sequence for use in sorting
        Collections.reverse(sequence);
        return sequence;    }
    Run | Debug
    public static void main(String[] args) {
        int[] array = {23, 42, 4, 16, 8, 15};
        System.out.println("Original Array: " + Arrays.toString(array));
        shellSort(array);
        System.out.println("Sorted Array: " + Arrays.toString(array));
    }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS D:\kcgi\Java> & 'C:\Users\gaire\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '
'C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cff8322bbc836dc0e92005c5cab87794\redhat.java\jdt_ws\Java_c3b3109e\bin' 'SedgewickShellSort'
Original Array: [23, 42, 4, 16, 8, 15]
Sorted Array: [4, 8, 15, 16, 23, 42]
PS D:\kcgi\Java>
```

Pseudocode



pseudocode for sedgewick shell sort

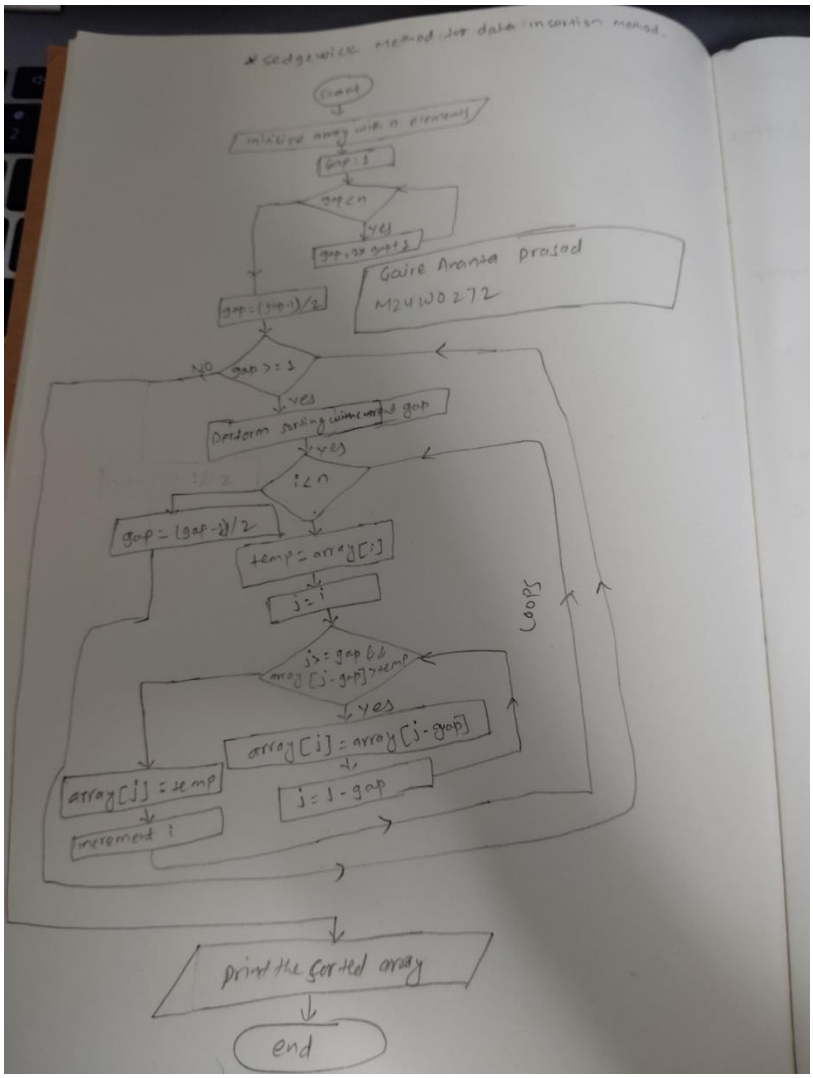
start: initialize sequence as empty list
Set k to 0.

compute: compute sequence values while true
Set gap to 9*(4^k) - 9*(2^k) + 1
If gap is less than n
Add gap to sequence
Set gap to (4^k) + 3*(2^(k-1)) + 1
If gap is less than n
Add gap to sequence
Set gap to (4^k) + 3*(2^(k-1)) + 1
If gap is less than n, Add gap to sequence
If gap is greater than or equal to n
Break the loop
Increment k by 1
Reserve the sequence

Gaire Ananta prasad
M24W0272

Compute: while sequence is not empty
Set gap to the last element from gap to the end of the array
For i from gap to n-1
Set temp to array[i]
Set j to i
compute: while j is greater than or equal to gap and array[j-gap]
is greater than temp
Set array[j] to array[j-gap]
Decrement j by gap
Set array[j] to temp
Display the sorted array
End the program

Flow-Chart



GAIRE ANANTA PRASAD
M24W0272