

GAIRE ANANTA PRASAD

M24W0272

Data Insertion Interval

1. Knuth Shell Sort

Java Code

```
KnuthShellSort.java > KnuthShellSort > shellSort(int[])
1  /*
2   * GAIRE ANANTA PRASAD
3   * M24W0272
4   */
5  import java.util.Arrays;
6  public class KnuthShellSort {
7      public static void shellSort(int[] array) {
8          int n = array.length;
9          // Initialize the gap using Knuth's sequence
10         int gap = 1;
11         while (gap < n / 3) {
12             gap = 3 * gap + 1; // 1, 4, 13, 40, ...
13         }
14         // Perform the sorting with the given gap
15         while (gap >= 1) {
16             for (int i = gap; i < n; i++) {
17                 int temp = array[i];
18                 int j = i;
19                 // Gapped insertion sort
20                 while (j >= gap && array[j - gap] > temp) {
21                     array[j] = array[j - gap];
22                     j -= gap;
23                 }
24                 array[j] = temp;
25             }
26             gap /= 3; // Reduce the gap
27         }
28     }
29     Run | Debug
30     public static void main(String[] args) {
31         int[] array = {23, 42, 4, 16, 8, 15};
32         System.out.println("Original Array: " + Arrays.toString(array));
33         shellSort(array);
34         System.out.println("Sorted Array: " + Arrays.toString(array));
35     }
36 }
```

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

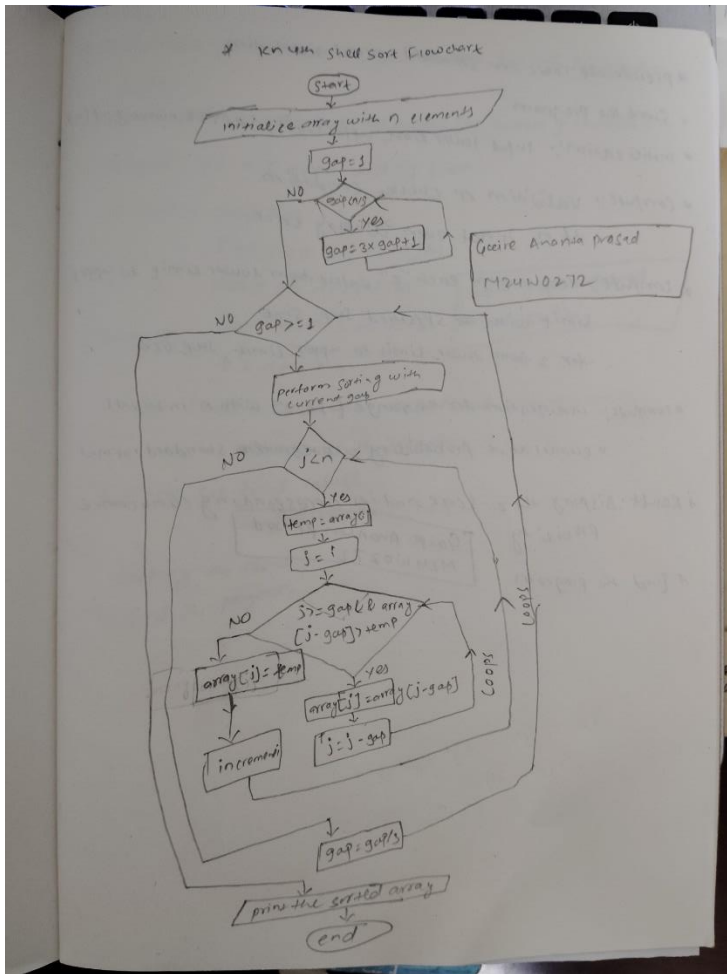
Pseudocode

* pseudocode for knuth shell sort

- knuth's sequence grows as $h = 3 \times h + 1$ and is reduced by $h = h/3$.
- + step 1: Start: initialize the gap sequence using knuth's method set gap to 1.
- + compute: while gap is less than n divided 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
- 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



2. Hibbard Shell Sort

Java Code

```
HibbardShellSort.java > HibbardShellSort > shellSort(int[])
1  /*
2   * GAIRE ANANTA PRASAD
3   * M24W0272
4   */
5  import java.util.Arrays;
6  public class HibbardShellSort {
7      public static void shellSort(int[] array) {
8          int n = array.length;
9          // Initialize gap using Hibbard's sequence
10         int gap = 1;
11         while (gap < n) {
12             gap = 2 * gap + 1; // 1, 3, 7, 15, 31, ...
13         }
14         gap = (gap - 1) / 2;
15         // Perform the sorting with the given gap
16         while (gap >= 1) {
17             for (int i = gap; i < n; i++) {
18                 int temp = array[i];
19                 int j = i;
20                 // Gapped insertion sort
21                 while (j >= gap && array[j - gap] > temp) {
22                     array[j] = array[j - gap];
23                     j -= gap;
24                 }
25                 array[j] = temp;
26             }
27             gap = (gap - 1) / 2; // Reduce the gap
28         }
29     }
30     public static void main(String[] args) {
31         int[] array = {23, 42, 4, 16, 8, 15};
32         System.out.println("Original Array: " + Arrays.toString(array));
33         shellSort(array);
34         System.out.println("Sorted Array: " + Arrays.toString(array));
35     }
36 }
```

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          }
10         gap = (gap - 1) / 2;
11         while (gap >= 1) {
12             for (int i = gap; i < n; i++) {
13                 int temp = array[i];
14                 int j = i;
15                 while (j >= gap && array[j - gap] > temp) {
16                     array[j] = array[j - gap];
17                     j -= gap;
18                 }
19                 array[j] = temp;
20             }
21             gap = (gap - 1) / 2;
22         }
23     }
24     public static void main(String[] args) {
25         int[] array = {23, 42, 4, 16, 8, 15};
26         System.out.println("Original Array: " + Arrays.toString(array));
27         shellSort(array);
28         System.out.println("Sorted Array: " + Arrays.toString(array));
29     }
30 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS D:\kcg\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:\kcg\Java>
```

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

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

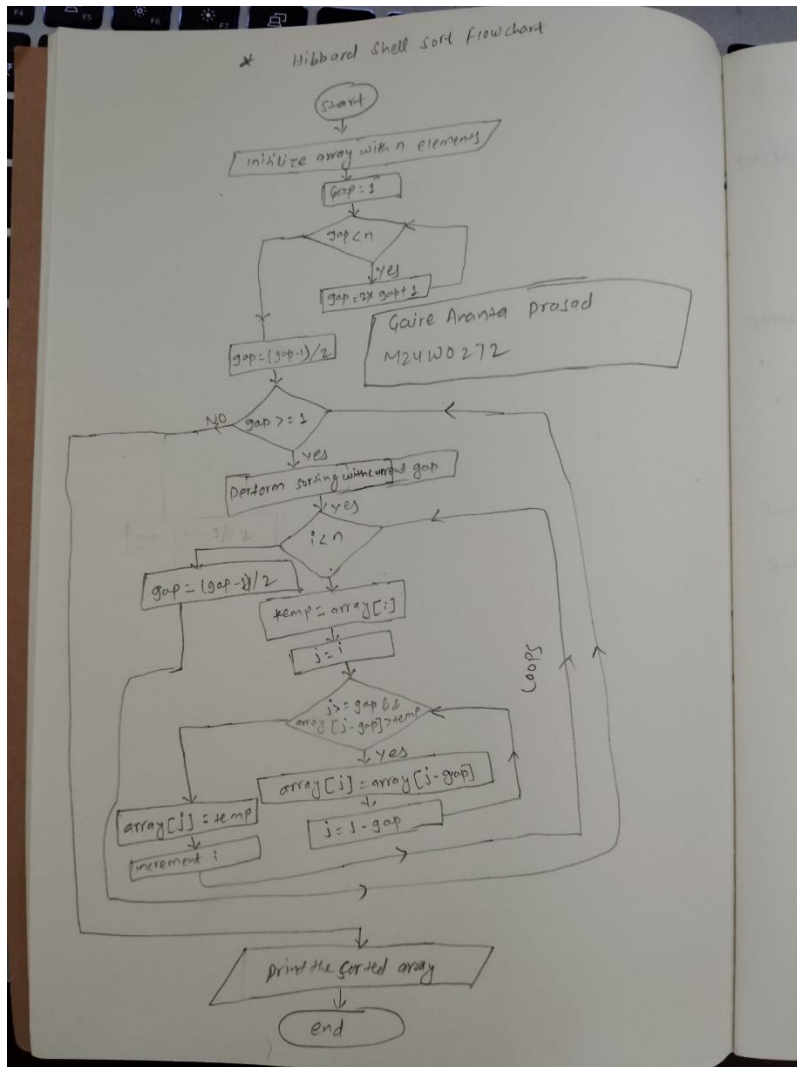
 Set gap to (gap - 1) divided by 2

Display the sorted Array

End.

Gaire Ananta Prasad
M24W0272

Flow-Chart



GAIRE ANANTA PRASAD
M24W0272

3. Sedgewick Shell Sort

Java Code

```
SedgewickShellSort.java > ...
1
2  * GAIRE ANANTA PRASAD
3  * M24W0272
4  */
5  import java.util.Arrays;
6  import java.util.ArrayList;
7  import java.util.Collections;
8  public class SedgewickShellSort {
9      public static void shellSort(int[] array) {
10         int n = array.length;
11         // Generate Sedgewick sequence
12         ArrayList<Integer> sequence = generateSedgewickSequence(n);
13         // Perform the sorting with the given gaps
14         for (int gap : sequence) {
15             for (int i = gap; i < n; i++) {
16                 int temp = array[i];
17                 int j = i;
18                 // Gapped insertion sort
19                 while (j >= gap && array[j - gap] > temp) {
20                     array[j] = array[j - gap];
21                     j -= gap;
22                 }
23                 array[j] = temp;
24             }
25         }
26     }
27
28     private static ArrayList<Integer> generateSedgewickSequence(int n) {
29         ArrayList<Integer> sequence = new ArrayList<>();
30         int k = 0;
31         int gap;
32         // Use Sedgewick's sequence:  $9 \cdot 4^k - 9 \cdot 2^k + 1$  and  $4^k + 3 \cdot 2^k(k-1) + 1$ 
33         while (true) {
34             gap = (int)(9 * Math.pow(4, k) - 9 * Math.pow(2, k) + 1);
35             if (gap < n) {
36                 sequence.add(gap);
37             }
38             gap = (int)(Math.pow(4, k) + 3 * Math.pow(2, k - 1) + 1);
39             if (gap < n) {
40                 sequence.add(gap);
41             }
42             k++;
43         }
44         // Reverse the sequence for use in sorting
45         Collections.reverse(sequence);
46         return sequence;
47     }
48
49     public static void main(String[] args) {
50         int[] array = {23, 42, 4, 16, 8, 15};
51         System.out.println("Original Array: " + Arrays.toString(array));
52         shellSort(array);
53         System.out.println("Sorted Array: " + Arrays.toString(array));
54     }
55 }
```

Output

```
SedgewickShellSort.java > SedgewickShellSort > generateSedgewickSequence(int)
8  public class SedgewickShellSort {
27  private static ArrayList<Integer> generateSedgewickSequence(int n) {
33      gap = (int)(9 * Math.pow(4, k) - 9 * Math.pow(2, k) + 1);
34      if (gap < n) {
35          sequence.add(gap);
36      }
37      gap = (int)(Math.pow(4, k) + 3 * Math.pow(2, k - 1) + 1);
38      if (gap < n) {
39          sequence.add(gap);
40      }
41      if (gap >= n) {
42          break;
43      }
44      k++;
45      // Reverse the sequence for use in sorting
46      Collections.reverse(sequence);
47      return sequence;
48  }
49
50  public static void main(String[] args) {
51      int[] array = {23, 42, 4, 16, 8, 15};
52      System.out.println("Original Array: " + Arrays.toString(array));
53      shellSort(array);
54      System.out.println("Sorted Array: " + Arrays.toString(array));
55  }
56 }
```

Run | Debug

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS D:\kcg\Java> & 'C:\Users\gaire\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
'C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cfff8322bbc836dc0e92005c5cab87794\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:\kcg\Java>
```

GAIRE ANANTA PRASAD
M24W0272

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 $g * (4 * k) - g * (2 * k) + 1$

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

 Reverse the sequence

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 $array[i]$ to $array[j - gap]$

 Decrement j by gap

 Set $array[j]$ to temp

Display the sorted array

End the program

Gaire Ananta prasad
M24W0272

```

graph TD
    Start([Start]) --> Init[/Initialize array with n elements/]
    Init --> Gap1[Gap = 1]
    Gap1 --> Cond1{if < n}
    Cond1 -- Yes --> Gap2[Gap = 2 * Gap + 1]
    Gap2 --> Gap3[Gap = (Gap + 1) / 2]
    Cond1 -- No --> Gap3
    Gap3 --> Cond2{Gap >= 1}
    Cond2 -- Yes --> Sort[/Perform Sorting with current Gap/]
    Sort --> Cond3{i < n}
    Cond3 -- Yes --> Gap4[Gap = (Gap - 1) / 2]
    Gap4 --> Temp[Temp = array[i]]
    Temp --> j[i = i]
    j --> Cond4{j - Gap < 0 & array[i - Gap] > Temp}
    Cond4 -- Yes --> Assign[array[i] = array[i - Gap]]
    Assign --> j1[j = i - Gap]
    j1 --> Cond4
    Cond4 -- No --> Assign2[array[i] = Temp]
    Assign2 --> Inc[i = increment i]
    Inc --> Cond3
    Cond3 -- No --> Print[/Print the sorted array/]
    Print --> End([End])
    
```

GAIRE ANANTA PRASAD
M24W0272