**Shell Sort**

**Group7 Chaoling Lu,Sheng Ming Yan,Mengdan Liu**

Shell Sort is a variation of Insertion Sort. It allows us to compare the biggest number and smallest number in the same gap range and swap them.

**Shell Sort in pseudocode:**

for(int i=1; i<len; i++){

for(int j=i-1; j>=0&&arr[j]>arr[j+1]; j--){

swap(arr,j,j+1);

}

}

1. Array list [19,2,6,1,4,3 ] with gap of 3,

Array length n=6;

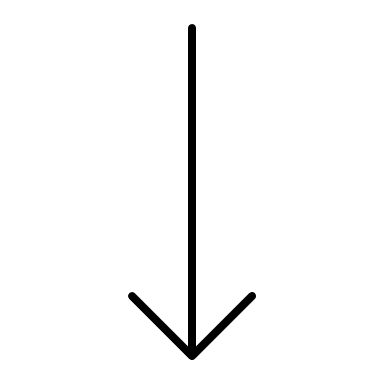
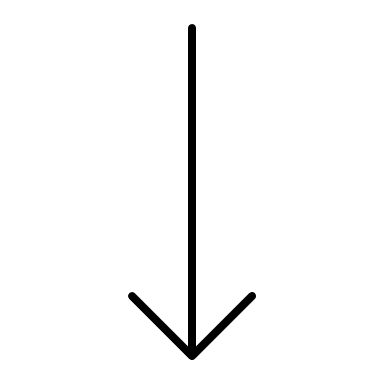
gap = n/3, i=gap, j=0;

compare array[j] and array [j+gap]:

19 > 1,then swap

j= j-gaplength = -4; loop end,i++;

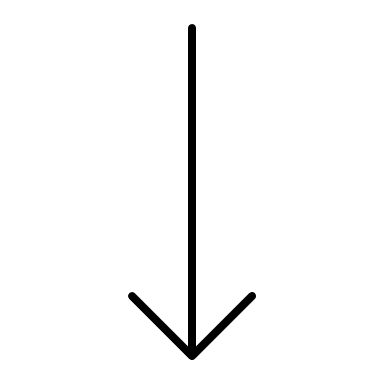
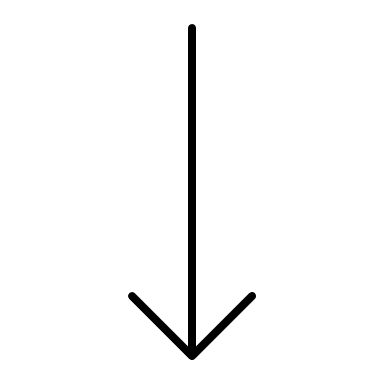
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 19 | 2 | 6 | 1 | 4 | 3 |

 swap 

i=4, j=i-gap =1, compare array[1] =2 < array[4] =4

j=j-gaplenght =-3 ,loop end,i++,j++

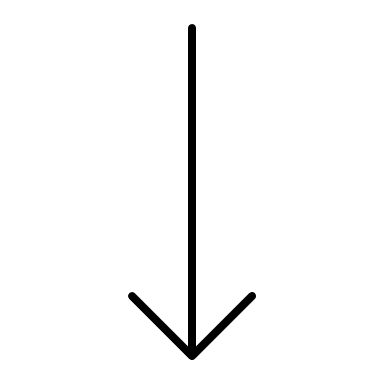
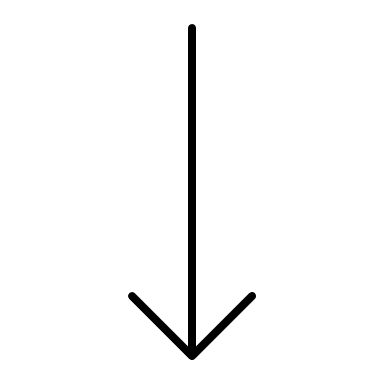
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 6 | 19 | 4 | 3 |

 no swap as 2<4 

i= 5,j= i-gap =2,compare array [2] =6 > array[5] =3

swap

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 6 | 19 | 4 | 3 |

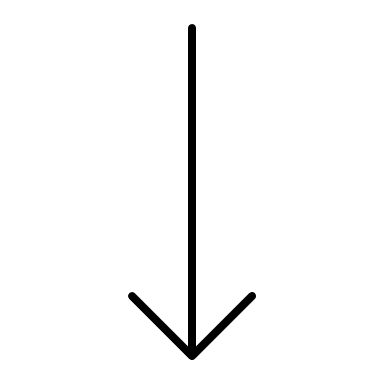
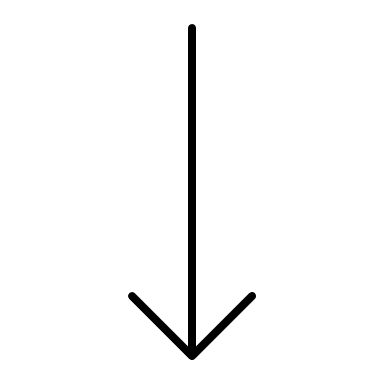
 swap 

1. Gap = 2

When i=2,j = i-gap =0,compare array[0] =1< array[2]=3

No swap;

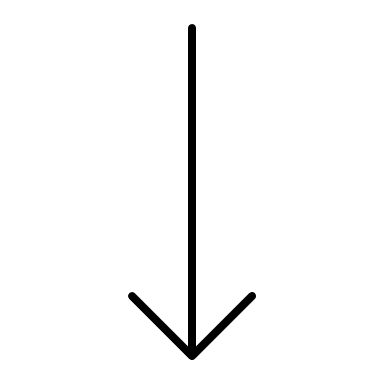
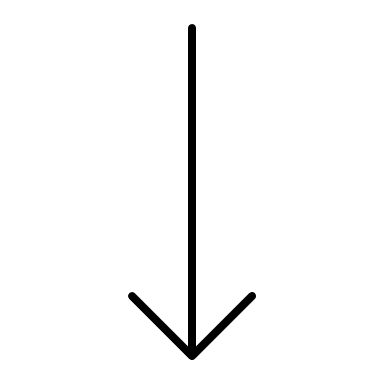
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 19 | 4 | 6 |

 no swap 

When i=3,j = i-gap =1,compare array[1] =2< array[3]=19

No swap;

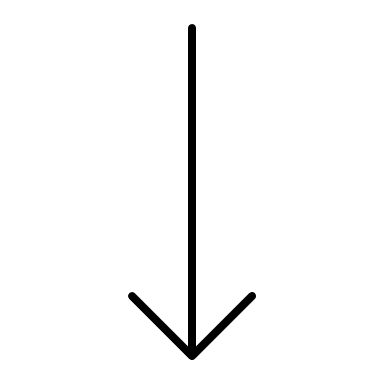
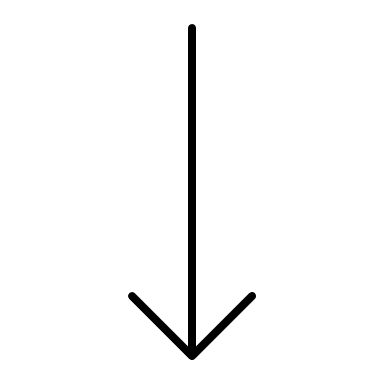
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 2 | 3 | 19 | 4 | 6 |

 no swap 

When i=4,j = i-gap =2,compare array[2] =6< array[4]=4

swap;

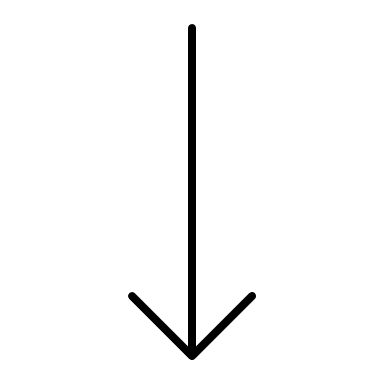
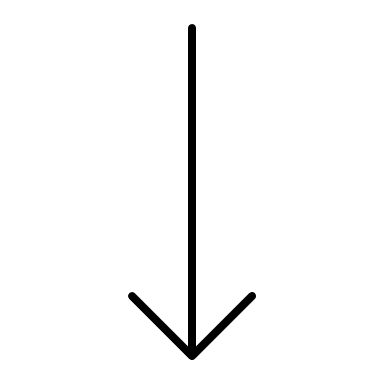
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 6 | 22 | 4 | 19 |

 swap 

When i=5,j = i-gap =3,compare array[3] =22< array[5]=19

swap;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 22 | 6 | 19 |

 swap 

1. With gap of 1 then Start Insertion sort.

for(int i=1; i<len; i++){

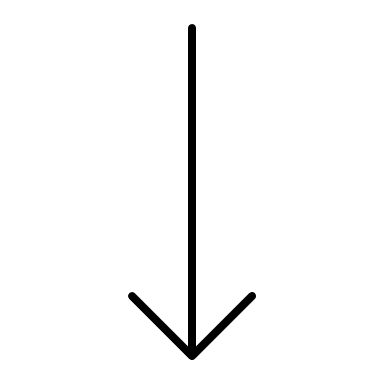
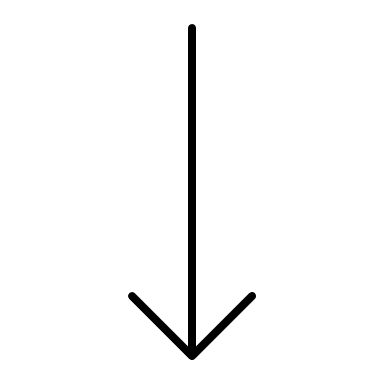
for(int j=i-1; j>=0&&arr[j]>arr[j+1]; j=j-1){

swap(arr,j,j+1);

}

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 19 | 6 | 22 |

 swap 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 6 | 19 | 22 |

**Shell Sort complexity:**

Shell sort adopts the array uses skip grouping as its strategy. The array elements are divided into several groups by a certain increment, and the groups are then sorted for insertion, and then the increment is gradually reduced, and the insertion sorting operation is repeated by groups until the increment is 1. Hill sort uses this strategy to make the entire array orderly in the initial stage, with the smaller ones in the front and the larger ones in the back. Then reduce the increment until the increment is 1.

However, in most cases, only fine-tuning is needed, and there will not be much data movement involved.

Time complexity: In the worst case, every two numbers must be compared and exchanged once, then the time complexity in the worst case is O(n2), in the best case, the array is ordered and does not need to be exchanged, only need to compare, then the time complexity in the best case is O(n).