Pigeonhole Sort

CSE211

By Group 25

- *Fardin Ahsan Shafi, 2020975 Sec-1*
- *Shanjidul Hasan Shajid 2022049 Sec-2*
- *Syed Niaz Mohtasim 2021607 Sec-1*

Introduction and History of the Algorithm

Pigeonhole Sorting is a linear time sorting algorithm. Pigeonhole sorting uses the idea of Pigeonhole Principle, If we have n number of pigeons and m number of holes and if n > m, then there must be at least one pigeonhole which contains more than one pigeon. German mathematician Peter Gustav Lejeune Dirichlet produced the concept of Pigeonhole Principle. He used this concept to produce mathematical proofs in his book called “Lectures on Number Theory” in 1863.

Like counting sort, Pigeonhole sort also uses the keys of an element to index into an auxiliary array The difference is that, in pigeonhole sorting, we copy over the elements itself and append it to a list that is contained within each index of the auxiliary array. In this way the auxiliary array with pigeonhole sort ends up containing all the elements that correspond to a particular index in the auxiliary array. And this means that once you have everything copied over all we do is concatenate together all these lists and then we will end up with the sorted array.

Pseudo-code

```
PigeonHoleSort(Array)
        min = Array.min
        max = Array.max
        range = max - min + 1
        holes = [0]*range

        for item in Array
                holes[item.key - min].append[item]

        i = 0
        for hole in holes
                for item in hole
                        Array[i++] = item
```

Pseudo-code

Complexity

```
PigeonHoleSort(Array)
        min = Array.min                          ……..O(n)
        max = Array.max                          ……..O(n)
        range = max - min + 1                    ……..O(1)
        holes = [0]*range                        ……..O(k)

        for item in Array                        ……..O(n)
                holes[item.key - min].append[item]    ……...O(1)

        i = 0                                    ……..O(1)
        for hole in holes                        ……..O(k)
                for item in hole                 ……..O(n)
                        Array[i++] = item        ……O(1)
                                                        O(n+k)
```

Complexity Analysis

Time Complexity

The total complexity of the algorithm will be a summation of the individual time complexity of each line which adds up to O(n+k). There are a total of n items that have to be iterated over so it's not possible to end up in a situation where there are n items that have to be copied over for each of the pigeonholes. We know that we are going to iterate over once for each of the pigeonholes and then we are going to do constant amount of work for each of the items so hence summation not multiplication. Time complexity is O(n+k) for all cases.

Space Complexity

Also O(n+k). We need k space for the pigeonholes because we have one pigeonhole for each of the range spaces and then we are also going to need an additional O(n) to contain all of these elements that we are copying over to the lists themselves. So k for the pigeonholes and n for the elements so O(n+k) just like the time.

Attributes of the Pigeonhole Sort

Adaptability	Online/Offline
Pigeonhole Sort is NON-ADAPTIVE. A sorting algorithm is non-adaptive when the runtime doesn’t depend on the element orientation of given list/array. In Pigeonhole Sorting, the runtime doesn’t depend on whether the list was initially sorted in ascending order or descending order. That’s why Pigeonhole Sorting is non-adaptive.	Pigeonhole sorting is Offline. An online algorithm can process its input in a serial fashion. So, without having the entire input available from the start a online algorithm can start working. In Pigeonhole Sorting, we need to calculate the min and max for the range of auxiliary array, so we can’t take new inputs in middle. That’s why Pigeonhole Sorting is Offline.

Simulation of the Algorithm

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

```
19  void pigeonHoleSort(int arr[], int size)
20  {
21      int minimum = arr[0];
22      int maximum = arr[0];
23      for (int i = 0; i < size; i++)
24      {
25          if (arr[i] > maximum)
26              maximum = arr[i];
27          if (arr[i] < minimum)
28              minimum = arr[i];
29      }
30
31      int range = maximum - minimum + 1;
32      list<int> holes[range];
33      for (int i = 0; i < size; i++)
34      {
35          holes[arr[i] - minimum].push_back(arr[i]);
36      }
37
38      int mainArrayIterator = 0;
39      for (int i = 0; i < range; i++)
40      {
41          for (int x : holes[i])
42          {
43              arr[mainArrayIterator++] = x;
44          }
45      }
46  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 10

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 10
max = 10

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 6
max = 20

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 6
max = 20
range = 15

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 6
max = 20
range = 15

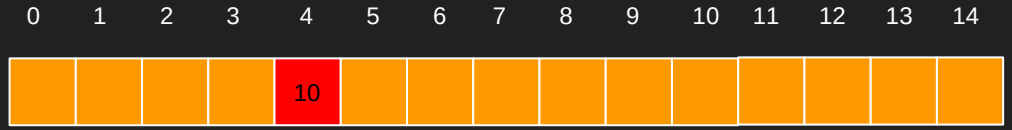
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
        list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=0
holes[4].push_back(10)

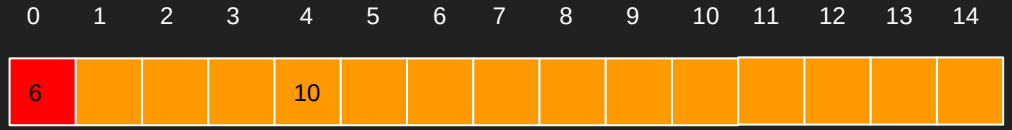



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=1
holes[0].push_back(6)

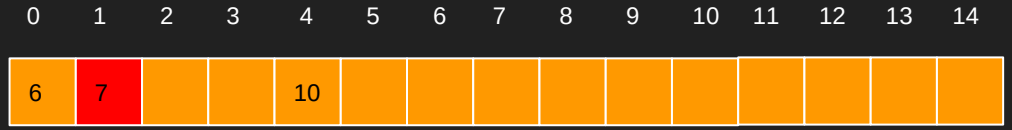



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=2
holes[1].push_back(7)

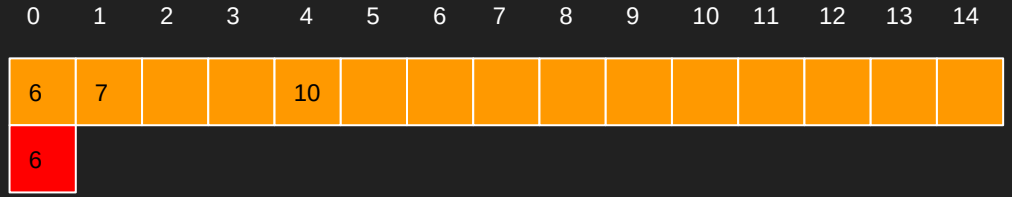



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=3
holes[0].push_back(6)

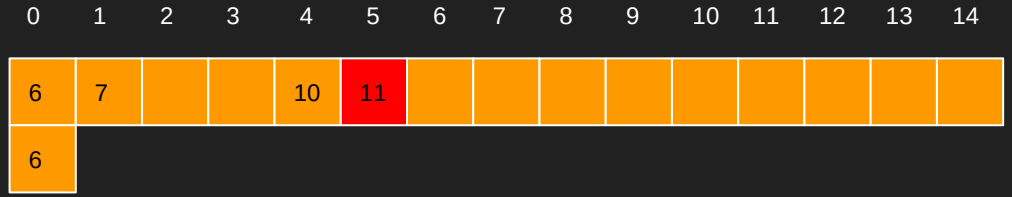



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=4
holes[5].push_back(11)




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

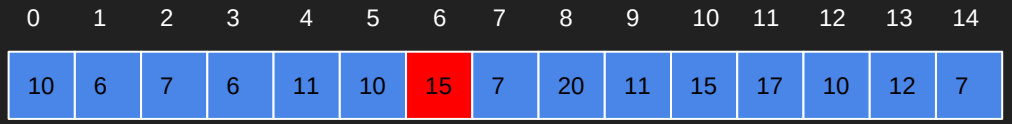
min = 6
max = 20
range = 15

i=5
holes[4].push_back(10)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11									
6				10										

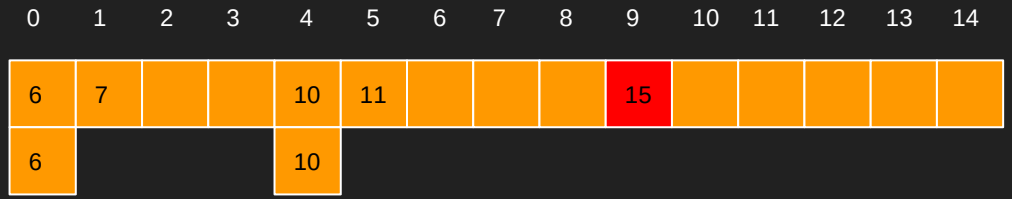
```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm



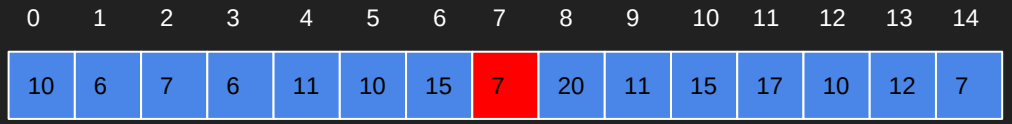
min = 6
max = 20
range = 15

i=6
holes[9].push_back(15)



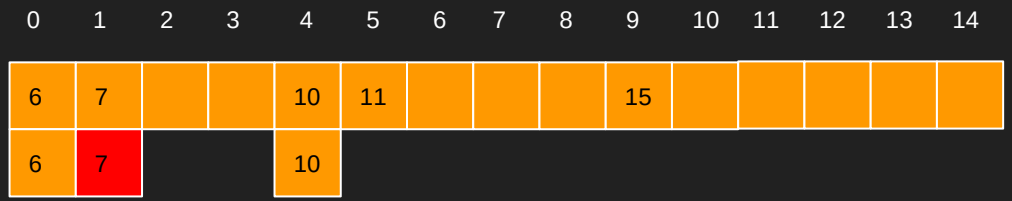

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm



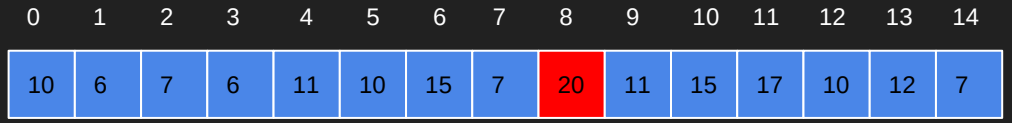
min = 6
max = 20
range = 15

i=7
holes[1].push_back(7)



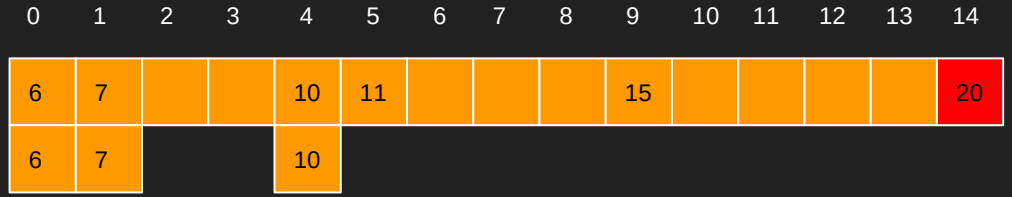

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm



min = 6
max = 20
range = 15

i=8
holes[14].push_back(20)

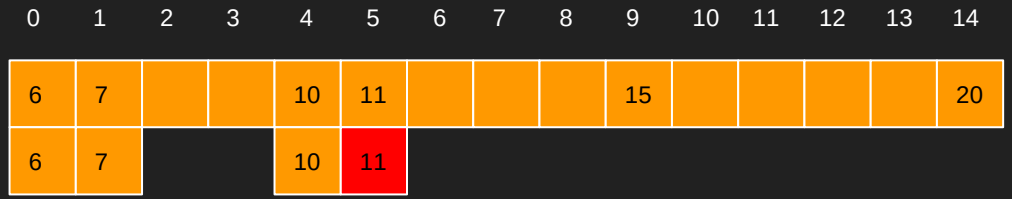



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=9
holes[5].push_back(11)

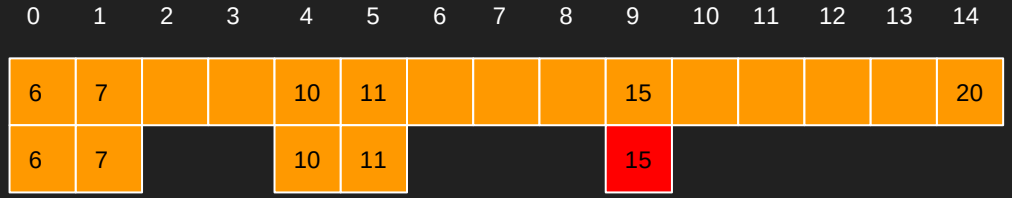



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=10
holes[9].push_back(15)

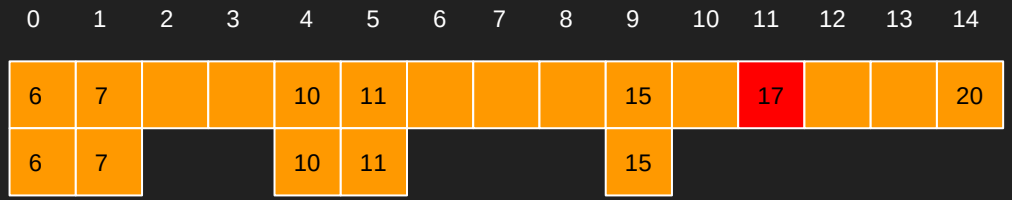



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=11
holes[11].push_back(17)

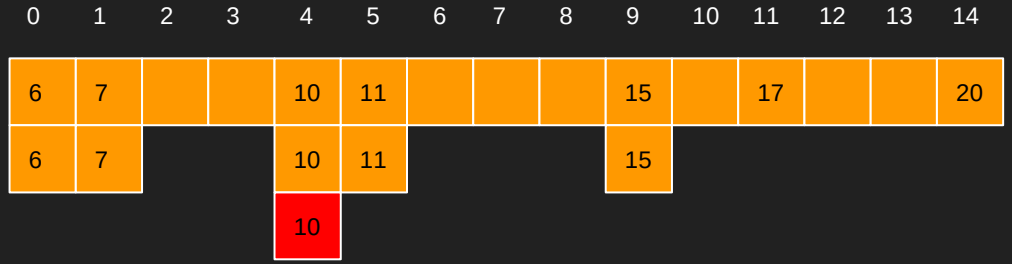



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=12
holes[4].push_back(10)




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

min = 6
max = 20
range = 15

i=13
holes[6].push_back(12)

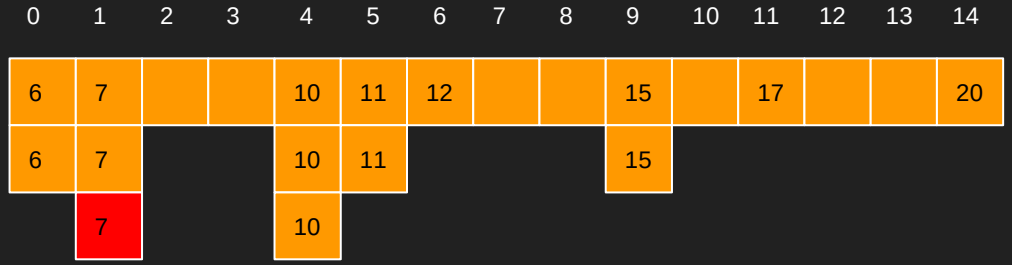
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
				10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15

i=14
holes[1].push_back(7)




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	6	7	6	11	10	15	7	20	11	15	17	10	12	7

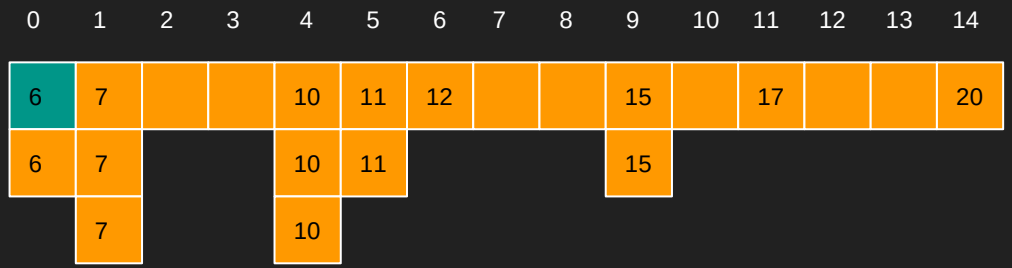
min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

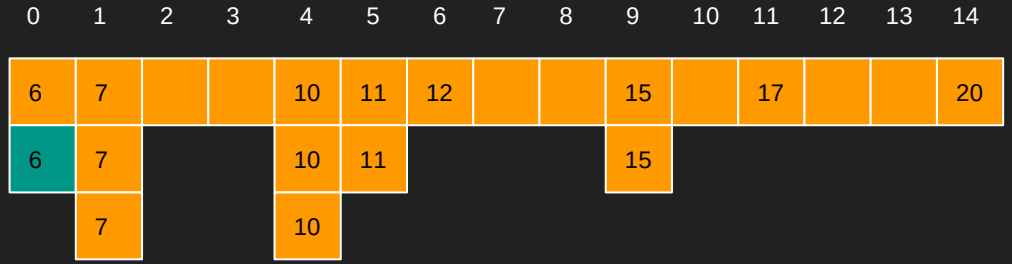
min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

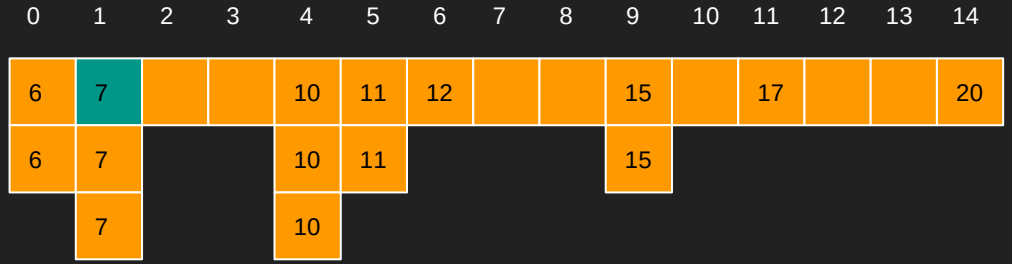
min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	11	10	15	7	20	11	15	17	10	12	7

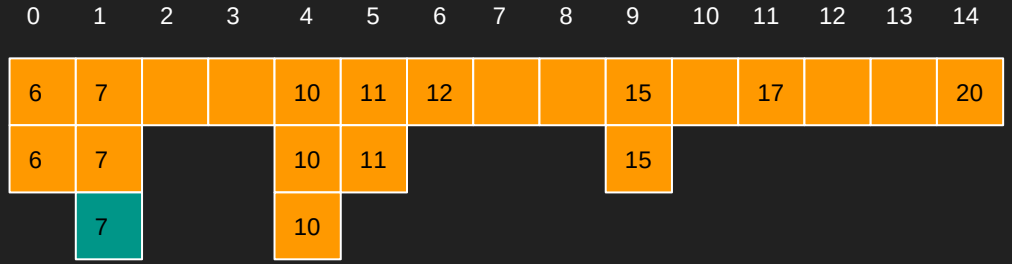
min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	15	7	20	11	15	17	10	12	7

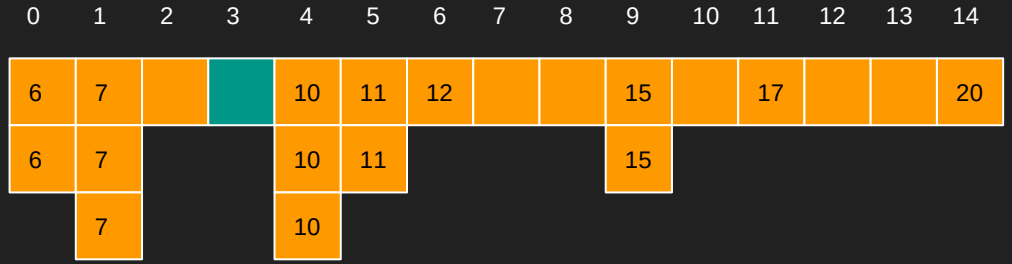
min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

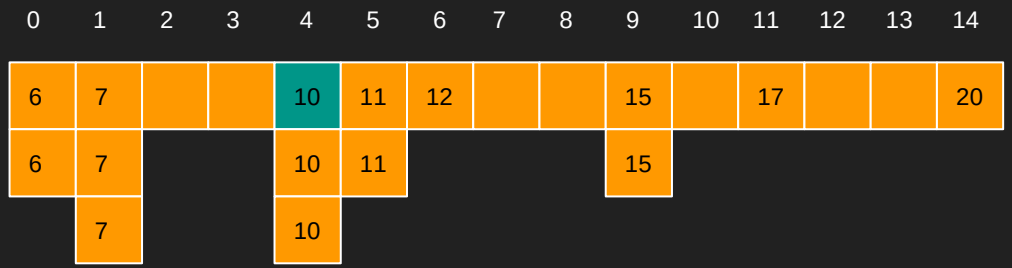
min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	7	20	11	15	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	20	11	15	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	15	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	15	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	17	10	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
            for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

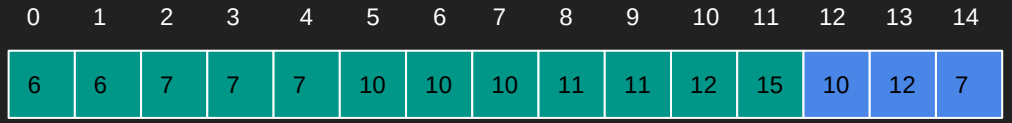
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	17	10	12	7

min = 6
max = 20
range = 15

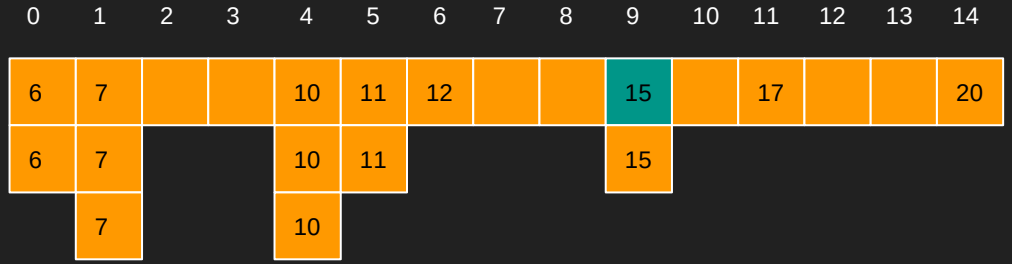
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm



min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	12	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

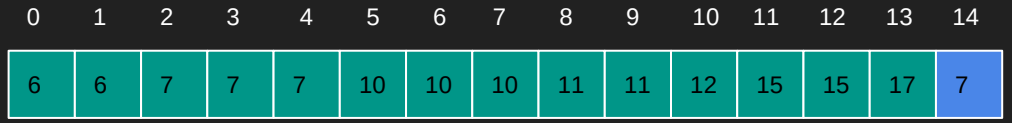
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	12	7

min = 6
max = 20
range = 15

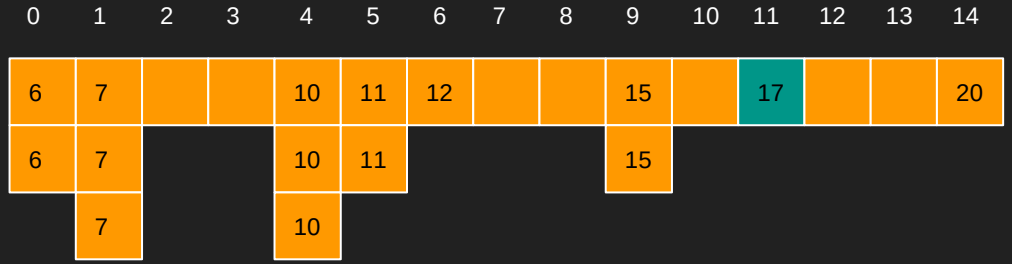
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

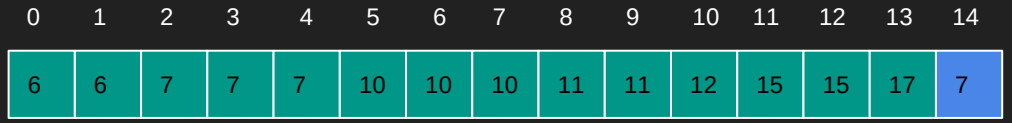


min = 6
max = 20
range = 15

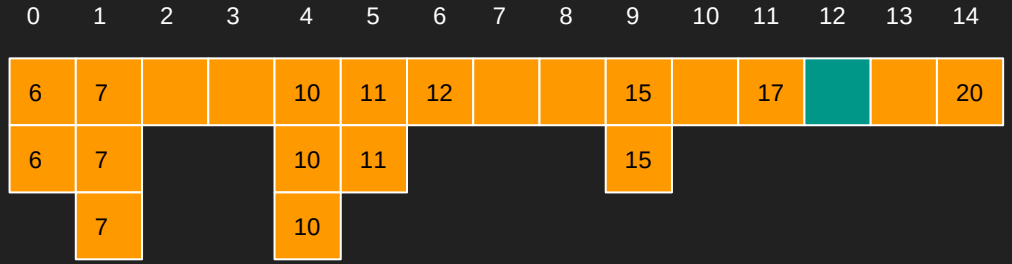



```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm



min = 6
max = 20
range = 15




```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	17	7

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	17	20

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
45  }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	17	20

min = 6
max = 20
range = 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	7			10	11	12			15		17			20
6	7			10	11				15					
	7			10										

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
    }
```

Simulation of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
6	6	7	7	7	10	10	10	11	11	12	15	15	17	20

```
18  void pigeonHoleSort(int arr[], int size)
19  {
20      int minimum = arr[0];
21      int maximum = arr[0];
22      for (int i = 0; i < size; i++)
23      {
24          if (arr[i] > maximum)
25              maximum = arr[i];
26          if (arr[i] < minimum)
27              minimum = arr[i];
28      }
29
30      int range = maximum - minimum + 1;
31      list<int> holes[range];
32      for (int i = 0; i < size; i++)
33      {
34          holes[arr[i] - minimum].push_back(arr[i]);
35      }
36
37      int mainArrayIterator = 0;
38      for (int i = 0; i < range; i++)
39      {
40          for (int x : holes[i])
41          {
42              arr[mainArrayIterator++] = x;
43          }
44      }
    }
```

Pros & Cons

Pros :

1. Can be used to count frequency of a given list.
2. Has a linear running time in all cases.
3. Stable sorting algorithm.
4. Non-comparison based sorting

Cons :

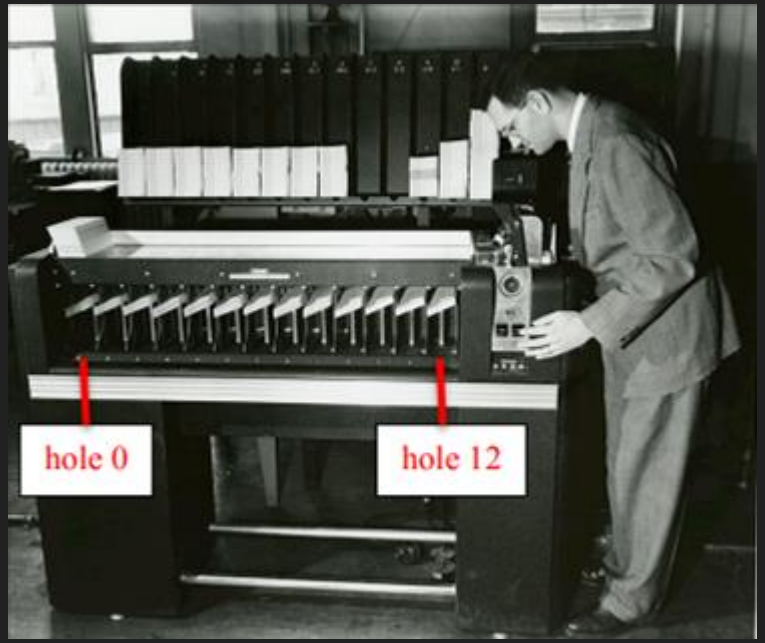
1. Can’t work with floating point numbers
2. If range(k) >> items(n), a lot of space gets wasted.
3. Can work with negative numbers only if the algorithm is modified to handle negative numbers.
4. Usually bucket sort, counting sort is preferred over pigeonhole sort in case of linear time sorting.

Practical Use

IBM card sorting
Depending on the holes punched in that column, each card is placed in one of the thirteen baskets, or pigeon holes.

Automated mail processing

Letter sorting for different destination



CONCLUSION

In short, Pigeonhole Sort works by indexing into an auxiliary array that contains a bunch of pigeonholes using the key of each particular element and copies over the elements on to the list in those pigeonholes by appending which ensures stability.Once all the elements are put in the lists, we iterate through the lists in order and copy the elements again in the original array which results in sorted original array

The time complexity is O(n+k) for all cases where ‘k’ is the range of values we can have. And this algorithm works efficiently when the number of element is approximately same as the range of elements.
The space complexity is also O(n+k) as discussed before.

It is similar to counting sort except we copy elements twice, once from original to auxiliary array and then back to original array.

This algorithm is generally not preferred as bucket sort is more capable in handling elements even if the range is significantly higher than the number of elements.