Evan Warner

Professor Poulson

Data Structures & Algorithms

5 February 2022

Sorting

Sorting is used in the coding world for a multitude of reasons. Sorting is used to arrange data in a certain order. If you wanted client names in alphabetical order you could code a sort program to take the data you have and arrange it to have the clients arranged alphabetically. In xava, there are multiple types of sorting algorithms for different things. The few I will explain will be merge and insertion. The merge sort is a very flexible type of sort. The strategy it uses is almost identical to divide and conquer. Ultimately it breaks an array down into smaller pieces and then sorts the smaller pieces which then leads to smaller pieces being merged to make larger segments and this process goes on and on until all are sorted.

class Sort

{

void merge(int arr[], int lt, int mid, int rt)

{

int low = mid - lt + 1;

int high = rt- mid;

int L[] = new int[low];

int R[] = new int[high];

int i = 0, x = 0;

for (i = 0; i < low; i++)

{

L[i] = arr[lt + i];

}

for (x = 0; x < high; x++)

{

R[x] = arr[mid + 1 + x];

}

int k = lt;

i = 0;

x = 0;

while (i < low && x < high)

{

if (L[i] <= R[x])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[x];

x++;

}

k++;

}

while (i < low)

{

arr[k] = L[i];

i++;

k++;

}

while (x < high)

{

arr[k] = R[x];

x++;

k++;

}

}

void mergeSort(int arr[], int lt, int rt)

{

int mid;

if (lt < rt) {

mid = (lt + rt) / 2;

mergeSort(arr, lt, mid);

mergeSort(arr, mid + 1, rt);

merge(arr, lt, mid, rt);

}

}

void display(int arr[])

{

for (int i=0; i<arr.length; ++i)

{

System.out.print(arr[i]+" ");

}

}

public static void main(String args[])

{

int arr[] = { 9, 3, 1, 5, 13, 12 };

Sort ob = new Sort();

ob.mergeSort(arr, 0, arr.length - 1);

ob.display(arr);

}

}

Merge Diagram

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | 4 | 3 | 8 | 5 | 1 |

1 3 4 5 8 9

1 5 8

3 4 9

1

5 8

3

4 9

3

1

8 5

9 4

9 4 3

1 8 5

Above is a diagram that gives a visual representation of how the merge sort works. Now let’s move onto the insertion sort method. This method of sorting is a bit simpler than the other. This method picks an array to be the key then compares it to the other arrays determining if it is smaller than the key or not and moves them around until they are in the wanted sorted order.

class Sort

{

static void insertionSort(int arr[], int n)

{

if (n <= 1)

{

return;

}

insertionSort( arr, n-1 );

int last = arr[n-1];

int x = n-2;

while (x >= 0 && arr[x] > last)

{

arr[x+1] = arr[x];

x--;

}

arr[x+1] = last;

}

void display(int arr[])

{

for (int i=0; i<arr.length; ++i)

{

System.out.print(arr[i]+" ");

}

}

public static void main(String[] args)

{

int arr[] = {222, 12, 110, 105, 26};

insertionSort(arr, arr.length);

Sort ob = new Sort();

ob.display(arr);

}

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 100 | 150 | 9 | 234 | 34 | 79 |
| 9 | 100 | 150 | 234 | 34 | 79 |
| 9 | 34 | 100 | 150 | 234 | 79 |
| 9 | 34 | 79 | 100 | 150 | 234 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Above is a diagram of how the insertion function works.