学习情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 于承志 | **学号** | 2021901678 |
| **学院** | 信息工程学院 | **专业** | 电子信息 |

（请在下面表格本周学习情况）

|  |
| --- |
| 学习情况简述 |
| 复习了快速排序，归并排序，KMP算法和二分查找  练习了相应代码 |
| 本周练习过的代码 |
| 代码一：  public class kmp {  private kmp(){}  public static int Kmp\_index(String S, String T , int pos){  int i = pos,j = 0;  int[] next = new int[255] ;  *get\_next*(T,next);  while( i <= S.length()-1 && j <= T.length()-1){  if (j == -1 || S.charAt(i) == T.charAt(j)){  i++;  j++;  }else {  j =next[j];  }  }  if (j > T.length()-1) {  return i-T.length();  }else return 0;  }  private static void get\_next(String T, int[] next){  int i = 0,k = -1;  next[0] = -1;  while(i < T.length()-1){  if (k == -1 || T.charAt(i) == T.charAt(k)){  k++;  i++;  next[i] = k;  }else{  k = next[k];  }  }  } }  public class main {  public static void main(String[] args) {  String t = "abababcccabcsbx";  String s = "abchjmcnmxcsXZcsabababcccabcsbxdsqwasc";  int l = kmp.*Kmp\_index*(s, t, 0);  System.*out*.println(l);  } } |
| 代码二：  public class BinarySearch {  private BinarySearch(){}  public static <E extends Comparable<E>> int search(E[] data , E target){  return *search*(data,0,data.length-1,target);  }  private static <E extends Comparable<E>> int search(E[] data ,int l ,int r, E target){  if (l > r) return -1;  int mid = l + (r-l)/2;  if (data[mid].compareTo(target) > 0){  return *search*(data,mid+1,r,target);  }else if (data[mid].compareTo(target) < 0){  return *search*(data,l,mid - 1,target);  }else return mid;   } } |
| 代码三：快排  import java.util.Arrays; import java.util.Random;  public class quicksort {  private quicksort(){}  public static <E extends Comparable<E>> void quicksort(E[] arr ){  *quicksort*(arr,0,arr.length-1);  }  private static<E extends Comparable<E>> void quicksort(E[] arr ,int l,int r){  if (l >= r ){  return;  }  int p = *partiton*(arr,l,r);  *quicksort*(arr,l,p-1);  *quicksort*(arr,p+1,r);   }  private static<E extends Comparable<E>> void sort(E[] arr ,int l,int r){   }  private static <E extends Comparable<E>> int partiton(E[] arr ,int l,int r){  Integer[] p = new Integer[5];  for (int i = 0; i < p.length; i++) {  p[i] = (new Random()).nextInt(0,r-l) + l;  }  mergesort.*sort*(p);   *swap*(arr,l,p[3]);  int j=l;  for (int i = l+1; i <= r; i++) {  if (arr[i].compareTo(arr[l]) < 0){  j++;  *swap*(arr,j,i);  }  }  *swap*(arr,j,l);  return j;  }   private static <E>void swap(E[] arr, int j, int i) {  E t = arr[j];  arr[j] = arr[i];  arr[i] = t;  }   } |
| 代码四：归并  import java.util.Arrays;  public class mergesort {   private mergesort(){}  public static<E extends Comparable<E>> void sort (E[] arr){  *sort*(arr, 0,arr.length-1);  }  private static<E extends Comparable<E>> void sort (E[] arr,int l ,int r){  if (l >= r) return;  int mid = (r + l)/2;  *sort*(arr,l,mid);  *sort*(arr,mid + 1,r);  *merge*(arr , l , mid , r);  }  public static<E extends Comparable<E>> void merge (E[] arr,int l, int mid ,int r){  E[] temp = Arrays.*copyOfRange*(arr,l,r+1);  int i = l , j = mid+1;  for (int k = l; k <= r ; k++) {  if ( i > mid ){  arr[k] = temp[j - l];  j++;  }else if ( j > r ){  arr[k] = temp[i-l];  i++;  } else if (temp[i -l].compareTo(temp[j-l]) <= 0) {  arr[k] = temp[i-l];  i++;  }else {  arr[k] = temp[j-l];  j++;  }   }   }  } |