eC Academy

Realize Your Dreams

AP Computer Science A Review

Week 10: Algorithm III Computation

DR. ERIC CHOU
IEEE SENIOR MEMBER

SECTION 1

Number Systems

Questions

- AP2017 – Q1(a)

C1(a) Reverse of Digits

```
public static int reverse(int x){
    if (x==0) return 0;
    int r=0;
    while (x>0){
        int d = x % 10;
        r = r*10 + d;
        x /= 10;
    }
    return r;
}
```

C1(b) Number of Zero Digits

```
public static int countZero(int x){
    if (x==0) return 1;
    int c=0;
    while (x>0){
        int d = x % 10;
        if(d==0) c++;
        x /= 10;
    }
    return c;
}
```

C1(c) Digit List

```
public static ArrayList<Integer> makeList(int x){
    ArrayList<Integer> alist = new ArrayList<Integer>();
    if (x==0) { alist.add(0); return alist; }
    while (x>0){
        int d = x % 10;
        alist.add(d);
        x /= 10;
      }
    return alist;
  }
```

C1(d) Sum of Digits

```
public static int sum(int x){
    if (x==0) return 0;
    int sum =0;
    while (x>0){
        int d = x % 10;
        sum += d;
        x /= 10;
    }
    return sum;
}
```

```
C1 Part (a):
 reverse(0)=0
 reverse(12345)=54321
 reverse(1010101)=1010101
C1 Part (b):
 countZero(0)=1
 countZero(12345)=0
 countZero(1010101)=3
C1 Part (c):
 makeList(0)=[0]
 makeList(12345)=[5, 4, 3, 2, 1]
 makeList(1010101)=[1, 0, 1, 0, 1, 0, 1]
C1 Part (d):
 sum(0)=0
 sum(12345)=15
 sum(1010101)=4
```

Key Points

- Take good care of **Zeros**! Zeros may need special care.
- Be aware of the direction (ascending or descending)

SECTION 1

ContainsExactWord()

Questions

- AP 2016 – Q2(b)

Observation

- str.equals(pattern); // exact the same
- str.indexOf(pattern+" ") == 0;
- (str.indexOf(" "+pattern)+1)+pattern.length() == str.length(); // pattern at end
- str.indexOf(pattern)>0; // pattern at middle

```java
public class ContainsExactWord
{
  public static boolean containsExactWord(String str, String pattern){
      return str.equals(pattern) ||   // exact the same
             str.indexOf(pattern+" ") == 0 ||
            (str.indexOf(" "+pattern)+1)+pattern.length() == str.length() || // pattern at end
             str.indexOf(" "+pattern+" ")>=0;  // pattern at middle
    }

  public static void main(String[] args){
      System.out.print("\f");
      System.out.println(containsExactWord("A", "A"));          true
      System.out.println(containsExactWord(" A", "A"));         true
      System.out.println(containsExactWord("A ", "A"));         true
      System.out.println(containsExactWord(" A ", "A"));        true
      System.out.println(containsExactWord("ABAB", "A"));       false
      System.out.println(containsExactWord(" AB", "A"));        false
      System.out.println(containsExactWord("BA ", "A"));        false
      System.out.println(containsExactWord(" ABA ", "A"));      false
      System.out.println(containsExactWord("AB A AB", "A"));    true
      System.out.println(containsExactWord("ABAB", "A"));       false
    }
}
```

SECTION 1

String Formatting

Left Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Center Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Right Aligned Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Justified Paragraph Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Given a formatLength

Given a line of text which is shorter than the formatLength (line length).

Adjust the text the following 4 formats:

- Left-Alignment
- Right-Alignment
- Centered
- Justified

Questions

- AP 2016-Q4

Left Alignment

```
public static String leftAlignment(String str, int formatLength){
    String s = str;
    for (int i=0; i<formatLength-str.length(); i++) { s += " ";}
    return s;
 }
```

```
                    //01234567890123456789012345678901
static String a = "It is a wonderful day!";
```

Right Alignment

```
public static String rightAlignment(String str, int formatLength){
    String s = "";
    for (int i=0; i<formatLength-str.length(); i++) { s += " ";}
    s += str;
    return s;
 }
```

Centered

```
public static String centered(String str, int formatLength){
    String s = "";
    int left = (formatLength -str.length())/2;
    int right = (formatLength -str.length())-left;
    for (int i=0; i<left; i++) { s += " "; }
    s += str;
    for (int i=0; i<right; i++) { s += " ";}
    return s;
 }
```

```
25  public static String justified(String str, int formatLength){
26      String s="";
27      String[] tokens = str.split(" ");
28      if (tokens.length == 0) return leftAlignment(s, formatLength);
29      if (tokens.length == 1) return centered(s, formatLength);
30      int total=0;
31      for (int i=0; i<tokens.length; i++){
32          tokens[i] = tokens[i].trim();
33          total += tokens[i].length();
34       }
35      int gap = (formatLength-total)/(tokens.length-1);
36      int leftover = (formatLength-total)%(tokens.length-1);
37      s = tokens[0];
38      for (int i=1; i<tokens.length; i++){
39          if (leftover >0) {s+= " "; leftover--; }
40          for (int j=0; j<gap; j++) s+= " ";
41          s += tokens[i];
42       }
43      return s;
44   }
```

```
public static void main(String[] args){
    System.out.print("\f");
    System.out.println("          1         2         3         4");
    System.out.println("01234567890123456789012345678901234567890");
    System.out.println(leftAlignment(a,30)+"$");
    System.out.println(rightAlignment(a,30)+"$");
    System.out.println(centered(a,30)+"$");
    System.out.println(justified(a,30)+"$");
 }
```

```
          1         2         3         4
01234567890123456789012345678901234567890
It is a wonderful day!         $
        It is a wonderful day!$
    It is a wonderful day!     $
It   is   a   wonderful   day!$
```

SECTION 1

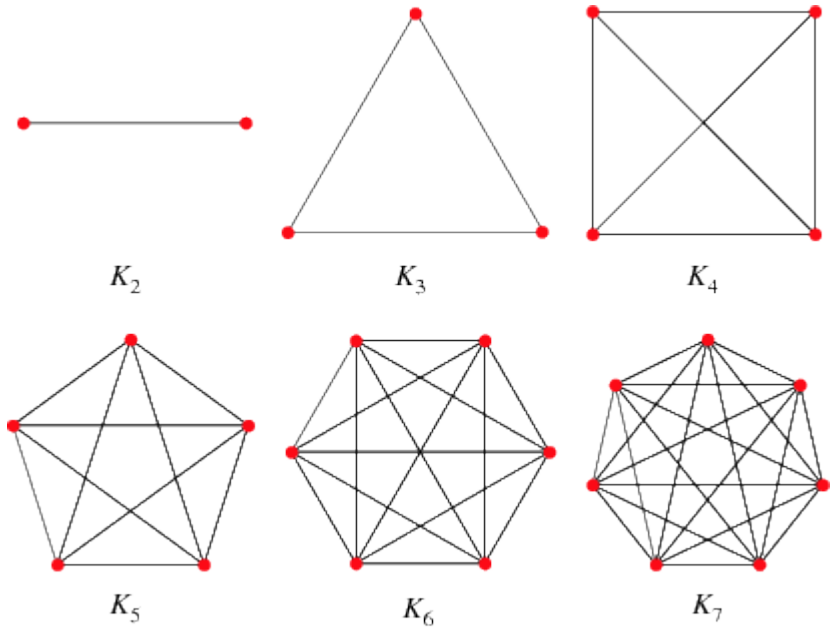
K-Map

Questions

- AP2015-Q1(a, b, c)

Mutual Relationship

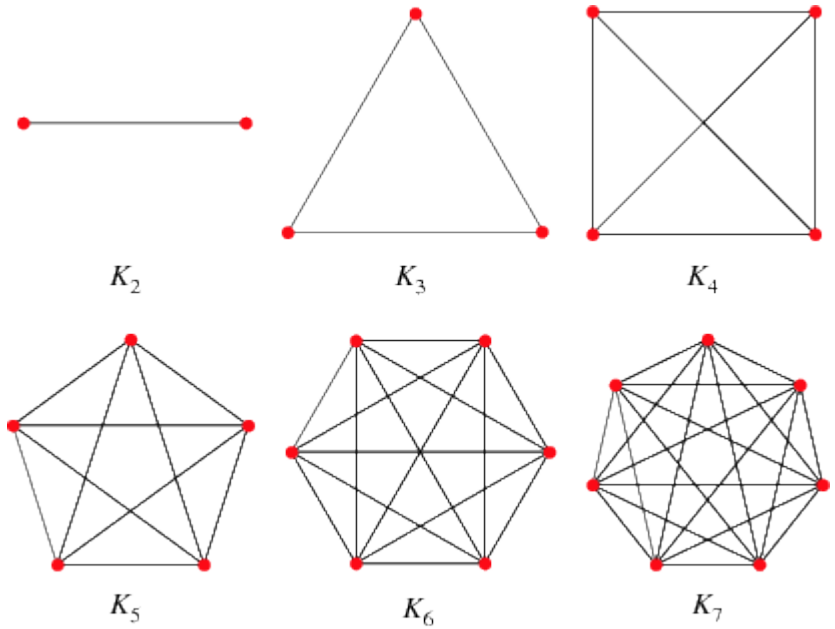
- Equality (Equality Check)
- Shortest Distance (Finding Max/Min)
- Inverse Order Count (Comparison)
- Friendships (Two-way checking)
- Matrix Transpose (Swap)




```
for (int i=1; i< arr.length; i++){
        for (int j=0; j<i; j++){
            cell[i][j] = '*';
            /* for all solution space
               (i, j) */
        }
    }
```

K-graph is for C(n, 2).

	0	1	2
0			
1	*		
2	*	*	




```
for (int i=0; i< arr.length-1; i++){
        for (int j=i+1; j<arr.length; j++){
            cell[i][j] = '*';
            /* for all solution space
               (i, j) */
        }
    }
```

K-graph is for C(n, 2).

	0	1	2
0		*	*
1			*
2			

Diversity

Check if a group of n numbers are diverse

- A group of number are called diverse if none of its members repeats.
- Use partial 2-D traversal for all one-to-one mapping from one member to another. If there is any equality, then return false. Otherwise, if there is no identical matching members, we return true.

```java
public class Diverse
{
    static int[] a = {1, 13, 6, 3, 7, 3, 7, 9, 8, 4, 32};
    static int[] b = {9, 8, 11, 7, 4, 3};

    public static boolean isDiverse(int[] x){

        for (int i=1; i<x.length; i++){
            for (int j=0; j<i; j++){
                if (x[i]==x[j]) return false;
              }
         }
        return true;
     }

    public static void main(String[] args){
        System.out.println("isDiverse(a)="+isDiverse(a));
        System.out.println("isDiverse(b)="+isDiverse(b));
     }
}
```

```
isDiverse(a)=false
isDiverse(b)=true
```

Matrix Transpose

- Each pair of elements will be swapped only once.
- Use K-map traversal. (upper half or lower half)
- Swap arr[r][c] with arr[c][r]

```java
public class MatrixTranspose{
    static int[][] m={
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}
    };
    public static void transpose(int[][] m){
        for (int i=1; i<m.length; i++){
          for (int j=0; j<i; j++){
              int tmp = m[i][j];
              m[i][j] = m[j][i];
              m[j][i] = tmp;
            }
        }
    }
    public static void display(int[][] m){
       for (int i=0; i<m.length; i++){
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args){
      System.out.println("\fBefore Transpose:");
      display(m);
      transpose(m);
      System.out.println("\nAfter Transpose:");
      display(m);
    }
}
```

```
Before Transpose:
1 2 3
4 5 6
7 8 9

After Transpose:
1 4 7
2 5 8
3 6 9
```

Inverse Order Count

- Given an array, see how many inverse relationship among the numbers
- Example: a = {1, 4, 3, 2, 5};
- Totally 3 inverse.

```java
public class InverseCount
{
    static int[] a = {1,4, 3, 2, 5};
    public static int inverseCount(int[] x){
        int count = 0;
        for (int i=1; i<x.length; i++){
            for (int j=0; j<i; j++){
                if (x[i]<x[j]) count++;
              }
         }
        return count;
    }

    public static void main(String[] args){
        System.out.println("inverseCount(a)="+inverseCount(a));
    }
}
```

BlueJ: Terminal Window - Week10

Options

```
inverseCount(a)=3
```

BlueJ: Terminal Window - Week10

Options

```
inverseCountUpper(a)=3
```

```java
public class InverseCountUpper
{
    static int[] a = {1,4, 3, 2, 5};
    public static int inverseCountUpper(int[] x){
        int count = 0;
        for (int i=0; i<x.length-1; i++){
            for (int j=i+1; j<x.length; j++){
                if (x[i]>x[j]) count++;
            }
        }
        return count;
    }

    public static void main(String[] args){
        System.out.println("inverseCountUpper(a)="+inverseCountUpper(a));
    }
}
```

Shortest Distance

- Find the two points in a set of 8 points. These two points have the shortest distance between them.
- Using K-map upper-half 2-D Traversal.
- Compare and identify the pair of points which have the minimum distance.
- Use Pair(x, y) as the two tuples return value.
- Use Point(a, b) as the two tuples parameters.

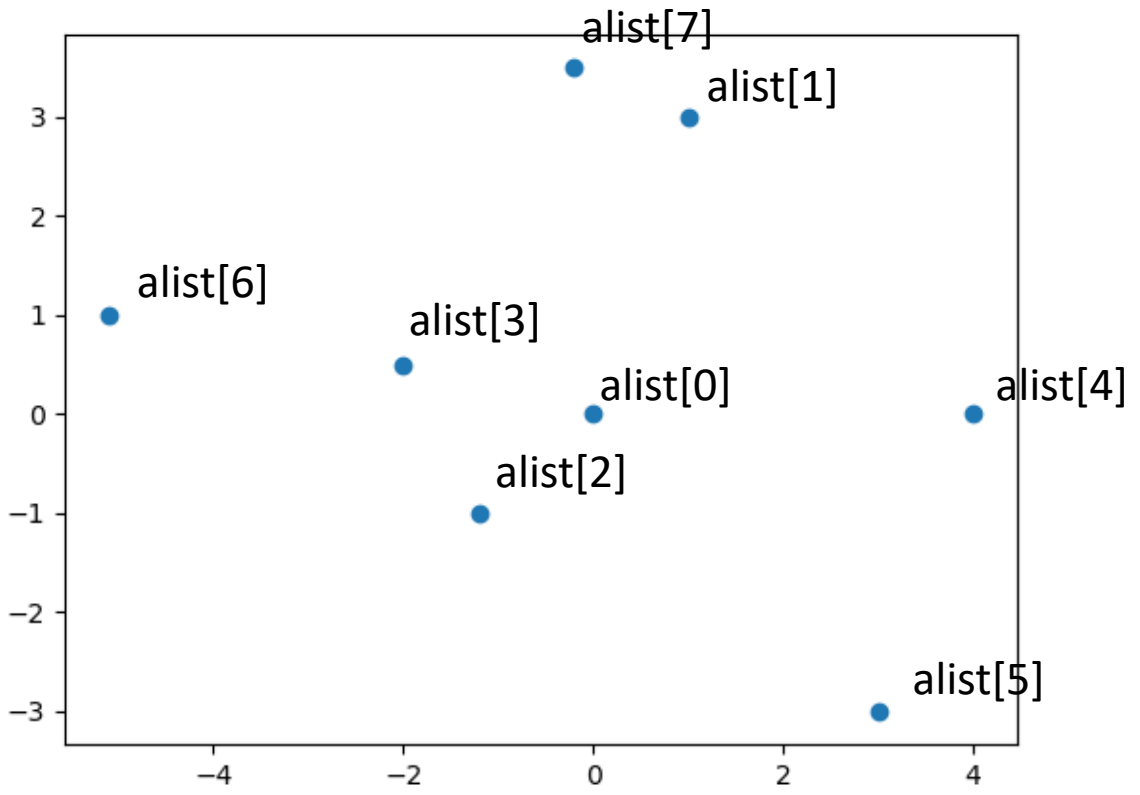
```
Minimum Distance=1.3 is between Point[1] and Point[7]
```

```java
public class ShortestDistance
{
    static Point[] alist = {
        new  Point(0.0, 0.0), new Point(1.0, 3.0),
        new  Point(-1.2, -1.0), new Point(-2.0, 0.5),
        new  Point(4.0, 0.0), new Point(3.0, -3.0),
        new  Point(-5.1, 1.0), new Point(-0.2, 3.5)
     };
    static public class Point{
       double x=0;
       double y=0;
       Point(double a, double b){
            x=a;
            y=b;
         }
     }
    static public class Pair{
       int p1=0;
       int p2=0;
       Pair(int a, int b){
            p1=a;
            p2=b;
         }
     }
    public static double distance(Point a, Point b){
        return Math.sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
     }

    public static void main(String[] args){
        Pair min = new Pair(0, 1);
        double min_distance = distance(alist[0], alist[1]);
        for (int i=0; i<alist.length-1; i++){
           for (int j=i+1; j<alist.length; j++){
                double dist = distance(alist[i], alist[j]);
                if (dist < min_distance){
                  min_distance = dist;
                  min = new Pair(i,j);
                }
            }
         }
        System.out.println("Minimum Distance="+min_distance+
              " is between Point["+min.p1+"] and Point["+min.p2+"]");
     }
}
```

Showing Minimum Distance of Points Using PyLab

```
1   from pylab import *
2
3   plist = [(0.0, 0.0), (1.0, 3.0), (-1.2, -1.0), (-2.0, 0.5),
4            (4.0, 0.0), (3.0, -3.0), (-5.1, 1.0), (-0.2, 3.5)]
5   x = [ a[0] for a in plist]
6   y = [ a[1] for a in plist]
7
8   figure()
9   scatter(x, y)
10  show()
```



Friendship Checking

- Mutual Checking for friendship relationship.
- We have an matrix of likes[][]. If a person who id is i likes another person j, then like[i][j] = true.
- If (likes[i][j] && likes[j][i]), then the function isFriends(i, j) will return true.
- Find out all the pair of the people who are friends. Using the same pair class from the previous example.

```java
import java.util.ArrayList;
public class Friends
{  static boolean[][] likes ={
        {false,  true,  true, false, false,  true},
        {false, false,  true,  true, false,  true},
        {true,   true, false,  true,  true, false},
        {true,  false,  true, false, false, false},
        {false, false,  true, false, false,  true},
        {true,  false,  true,  true, false, false}
    };
    static public class Pair{
       int p1=0;
       int p2=0;
       Pair(int a, int b){
            p1=a;
            p2=b;
         }
       public String toString(){ return "("+p1+", "+p2+")"; }
    }
```

```
static boolean[][] likes ={
      {false,  true,  true, false, false,  true},
      {false, false,  true,  true, false,  true},
      {true,   true, false,  true,  true, false},
      {true,  false,  true, false, false, false},
      {false, false,  true, false, false,  true},
      {true,  false,  true,  true, false, false}
  };
```

```
[(2, 0), (2, 1), (3, 2), (4, 2), (5, 0)]
```

```
20    public static void main(String[] args){
21        ArrayList<Pair> f = new ArrayList<Pair>();
22
23        for (int i=1; i<likes.length; i++){
24            for (int j=0; j<i; j++){
25                if (likes[i][j] && likes[j][i]) f.add(new Pair(i, j));
26
27             }
28         }
29
30        System.out.println(f);
31     }
32 }
```