eC Academy

Realize Your Dreams

AP Computer Science A Review

Week 12: Algorithm V Traversal

DR. ERIC CHOU
IEEE SENIOR MEMBER

SECTION 1

String, ArrayList<String>, char[] Conversion

Questions

- AP2016 – Q1(b)

String, Array of Characters and ArrayList of Characters

```
static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
static char[] alpha = getChar(alphabet);
static ArrayList<String> alphaList = new ArrayList<String>();
```

- Java does not have direct conversion between these data types.
- Use character level operators effectively.

String to array of characters

```
public static char[] getChars(String x){
    char[] a = new char[x.length()];
    for (int i=0; i<x.length(); i++) a[i]= x.charAt(i);
    return a;
  }
```

Array of Characters to String

```
public static String reverseToString(char[] x){
    String str = "";
    for (int i=x.length-1; i>=0; i--){
        str += ""+x[i];
    }
    return str;
}
```

String to ArrayList of Strings

```
public static ArrayList<String> getList(String x){
    ArrayList<String> a = new ArrayList<String>();
    for (int i=0; i<x.length(); i++){
        a.add(x.substring(i, i+1));
      }
    return a;
  }
```

ArrayList of Strings to String

```
public static String reverseToString(ArrayList<String> x){
    String str = "";
    for (int i=x.size()-1; i>=0; i--){
        str += ""+x.get(i);
     }
     return str;
  }
```

Testing

```
public static void main(String[] args){
    alphaList = getList(alphabet);
    System.out.print("\f");
    System.out.println(alphabet);
    System.out.println(Arrays.toString(alpha));
    System.out.println(reverseToString(alpha));
    System.out.println(alphaList);
    System.out.println(reverseToString(alphaList));
  }
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]
ZYXWVUTSRQPONMLKJIHGFEDCBA
[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z]
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

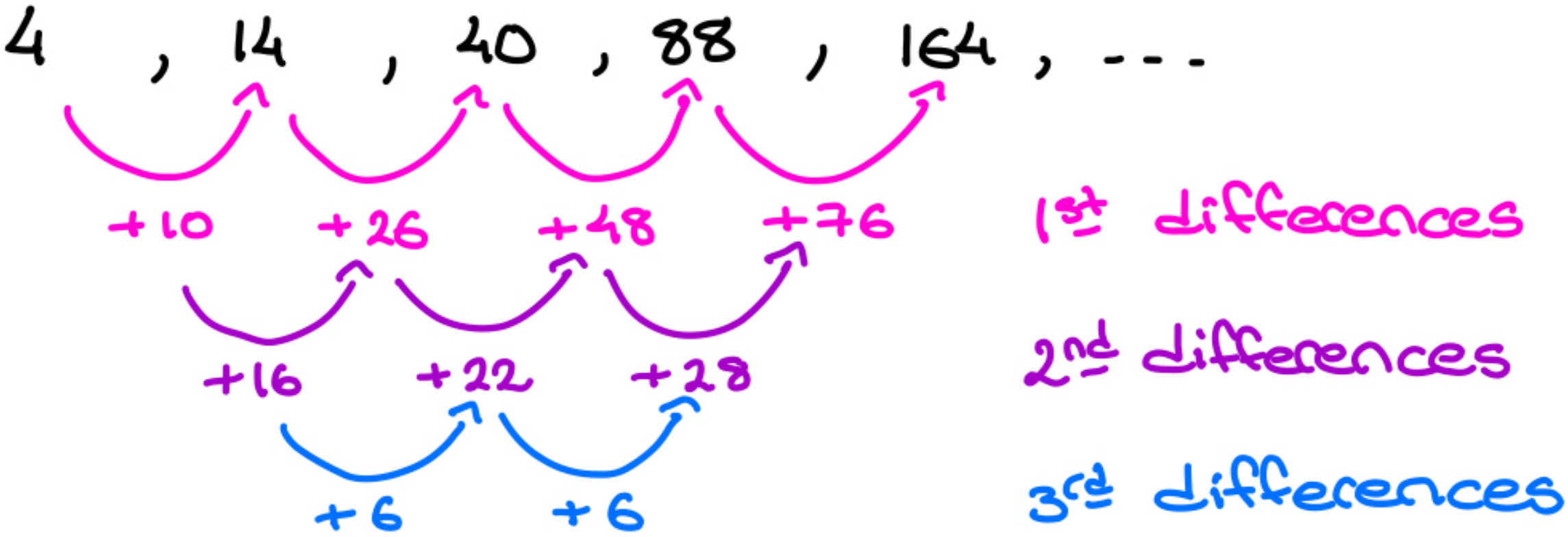
SECTION 1

Difference of a Sequence

Common Difference

The common difference is the constant amount of change between numbers in an arithmetic sequence.

common difference: 2

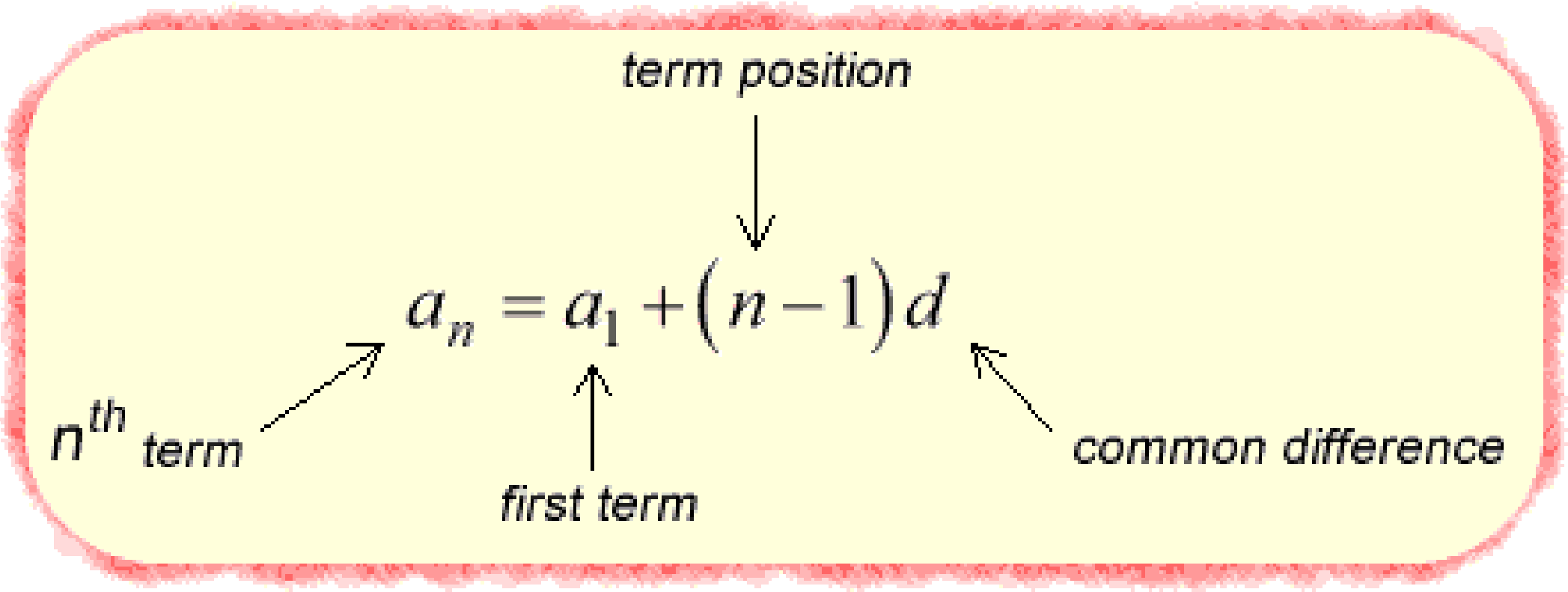

4 , 14 , 40 , 88 , 164 , ---
+10 +26 +48 +76
+16 +22 +28
+6 +6
1st differences
2nd differences
3rd differences

Arithmetic Sequence

A pattern of numbers that increase or decrease at a constant amount

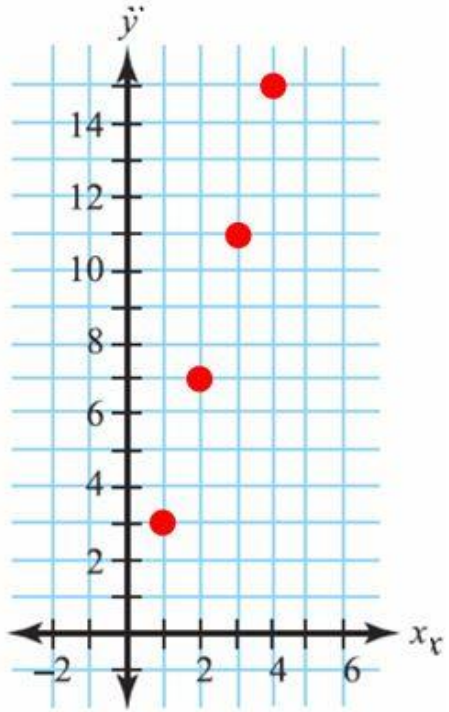
2, 4, 6, 8, 10, 12 . . .

common difference: 2


term position
$a_n = a_1 + (n-1)d$
n^{th} term
first term
common difference

Complete the table, and graph the corresponding points.

	n	$t_n = 3 + (n - 1)4$	t_n
1.	1	$t_1 = 3 + (1 - 1)4$	3
2.	2	$t_2 = 3 + (2 - 1)4$	7
3.	3	$t_3 = 3 + (3 - 1)4$	11
4.	4	$t_4 = 3 + (4 - 1)4$	15



5. Write a linear function for the relationship between t and n.

$t = 3 + 4n$

6. What does the number 4 represent in both the graph and the sequence?

The slope and the common difference

SITUATION 5 (money in wallet paying for sodas)

John has $ 10 in his wallet. Each soda he buys costs $ 2.00.

SLOPE (rate of change) **- 2**

Y-intercept (initial value) **10**

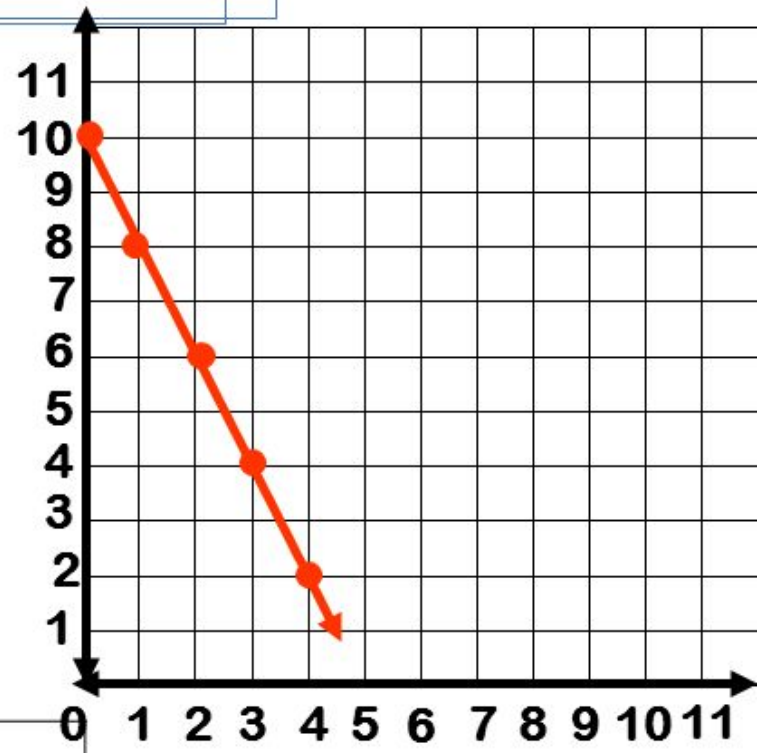
TABLE

x Number of sodas bought	y Amount left
0	10
1	8
2	6
3	4
4	2

EQUATION

$$y = -2x + 10$$

GRAPH



SEQUENCE

Initial Value	1st	2nd	3rd	4th
10	8	6	4	2

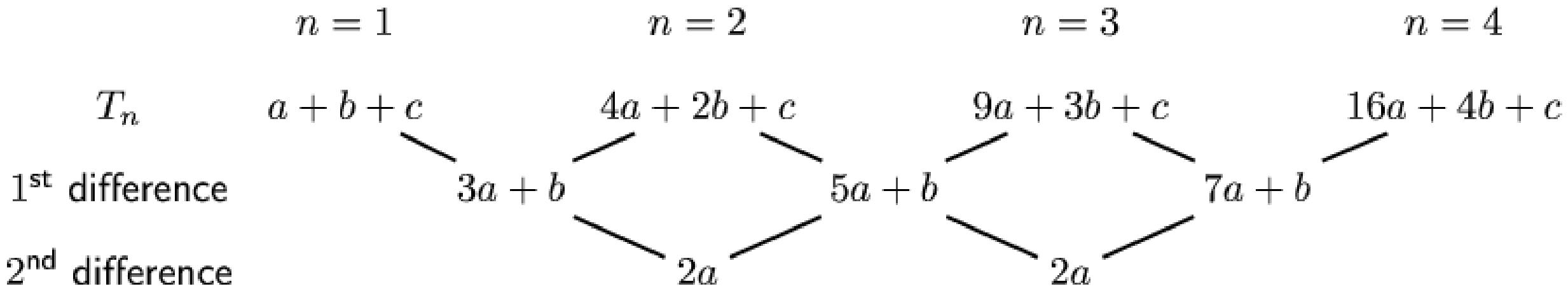
NEXT-NOW STATEMENT

NEXT = NOW – 2 ;
STARTING AT 10

Practical Domain $0 \le x \le 5$

Practical Range $0 \le y \le 10$

$$T_n = an^2 + bn + c$$

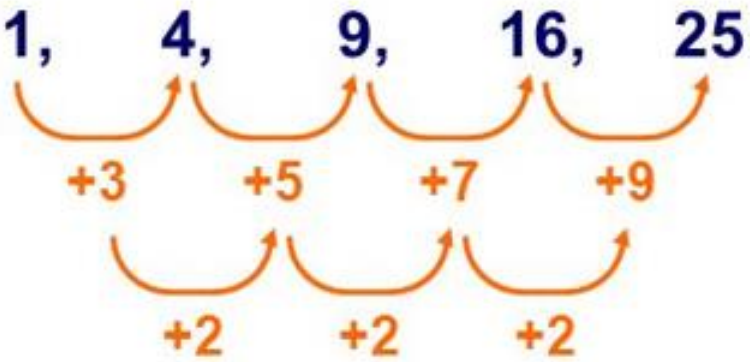

n = 1
n = 2
n = 3
n = 4
T_n
a + b + c
4a + 2b + c
9a + 3b + c
16a + 4b + c
1st difference
3a + b
5a + b
7a + b
2nd difference
2a
2a

Quadratic sequences

When the second row of differences produces a constant number the sequence is called a **quadratic sequence**.
This is because the rule for the n^{th} term of the sequence is a quadratic expression of the form

$$u_n = an^2 + bn + c$$

where a, b and c are constants and $a \neq 0$.
The simplest quadratic sequences is the sequence of **square numbers**.



The constant second difference is 2 and the n^{th} term is n^2.

Questions

- AP2017 – Q1(b)

A1(a) Strictly Increasing

```
public static boolean isStrictlyIncreasing(int[] x){
    for (int i=0; i< x.length-1; i++){
        if (x[i+1]<=x[i]) return false;
    }
    return true;
}
```

A1(b) maximum Slope

```
public static int maxSlope(int[] x){
    int max = Integer.MIN_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (x[i+1]-x[i]> max) { max = x[i+1]-x[i]; }
    }
    return max;
}
```

A1(c) Minimum Slope

```
public static int minSlope(int[] x){
    int min = Integer.MAX_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (x[i+1]-x[i]< min) { min = x[i+1]-x[i]; }
    }
    return min;
}
```

A1(d) Maximum Delta

```
public static int maxDelta(int[] x){
    int max = Integer.MIN_VALUE;
    if (x.length<2) return 0;
    for (int i=0; i< x.length-1; i++){
        if (Math.abs(x[i+1]-x[i])> max) {
            max = (int) Math.abs(x[i+1]-x[i]);
        }
    }
    return max;
}
```

```
public static int[] a={
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
    };
public static int[] b={
        1, 4, 3, 5, 6, 7, 0, 9
    };
public static int[] c={0};
```

```
A1 Part(a):
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] is strictly increasing = true
[1, 4, 3, 5, 6, 7, 0, 9] is strictly increasing = false
[0] is strictly increasing = true
A1 Part(b):
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] maxSlope = 1
[1, 4, 3, 5, 6, 7, 0, 9] maxSlope = 9
[0] maxSlope = 0
A1 Part(c):
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] minSlope = 1
[1, 4, 3, 5, 6, 7, 0, 9] minSlope = -7
[0] minSlope = 0
A1 Part(d):
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] maxDelta = 1
[1, 4, 3, 5, 6, 7, 0, 9] maxDelta = 9
[0] maxDelta = 0
```

Key Points

- Boundary condition: null pointer, empty array (length==0), single element array.

SECTION 1

Find() function

Questions

- AP2017– Q3(a, b, c)

Find a pattern in a string

- find: Find the first occurrence.
- findN: Find the Nth occurrence without overlapping patterns.
- findL: Find the last occurrence without overlapping patterns.
- findAN: Find the Nth occurrence with overlapping patterns.
- findAL: Find the Last occurrence with overlapping patterns.

Find

This one is the same as indexOf

```
public static int find(String str, String pattern){
   if (str.indexOf(pattern) >=0) return str.indexOf(pattern);
   return -1;
}
```

findN

```
public static int findN(String str, String pattern, int n){
  int i=-1;
  int pos =0;
  int count = 0;
  while (str.indexOf(pattern, pos) >=0 && count != n) {
      i = str.indexOf(pattern, pos);
      pos = i+1;
      count++;
    }
  if (count != n) return -1;
  return i;
}
```

findL

```
public static int findL(String str, String pattern){
    int i = -1;
    int pos =0;
    while (str.indexOf(pattern, pos) >=0) {
        i = str.indexOf(pattern, pos);
        pos = i+1;
    }
    return i;
}
```

findAN

```
public static int findAN(String str, String pattern, int n){
  int i=-1;
  int pos =0;
  int count = 0;
  while (str.indexOf(pattern, pos) >=0 && count != n) {
      i = str.indexOf(pattern, pos);
      pos = i+pattern.length();
      count++;
    }
  if (count != n) return -1;
  return i;
}
```

findAL

```
public static int findAL(String str, String pattern){
    int i = -1;
    int pos =0;
    while (str.indexOf(pattern, pos) >=0) {
        i = str.indexOf(pattern, pos);
        pos = i+pattern.length();
    }
    return i;
}
```

```java
public static void main(){
    System.out.print("\f");
    System.out.println("A2 Part(a):");
    System.out.println("0         1         2         3         4         5         6         7");
    System.out.println("01234567890123456789012345678901234567890123456789012345678901234567890123456789");
    System.out.println(a);
    System.out.println(find(a.toLowerCase(), "ha"));
    System.out.println(findN(a.toLowerCase(), "ha", 6));  // Overlapping
    System.out.println(findL(a.toLowerCase(), "ha"));
    System.out.println(findAN(a.toLowerCase(), "ha", 6)); // Non-Overlapping
    System.out.println(findAL(a.toLowerCase(), "ha"));
    System.out.println("A2 Part(b):");
    String b = "bbb bbb bbbbb";
    System.out.println(b);
    System.out.println(find(b.toLowerCase(), "bb"));
    System.out.println(findN(b.toLowerCase(), "bb", 3));  // Overlapping
    System.out.println(findL(b.toLowerCase(), "bb"));
    System.out.println(findAN(b.toLowerCase(), "bb", 3)); // Non-Overlapping
    System.out.println(findAL(b.toLowerCase(), "bb"));

}
```

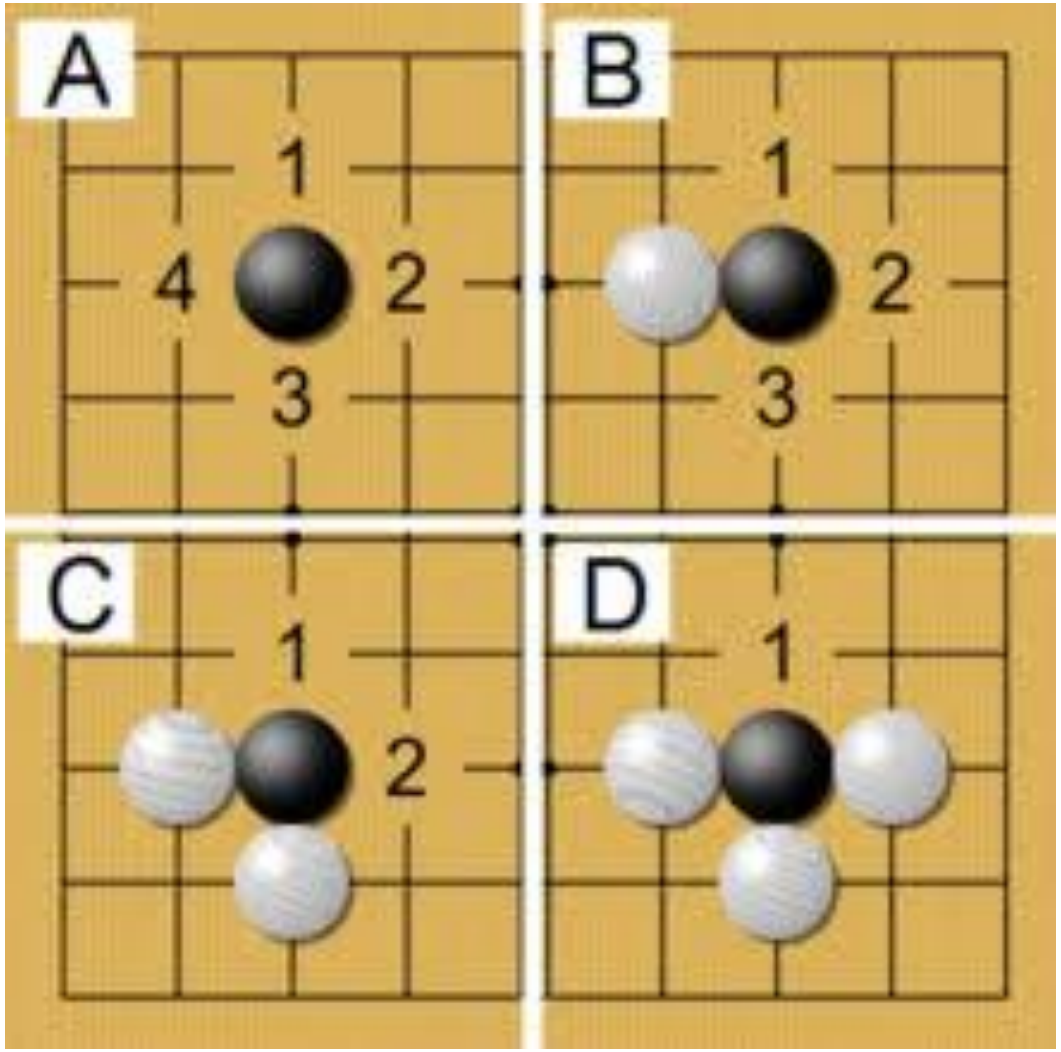
```
A2 Part(a):
0         1         2         3         4         5         6         7
01234567890123456789012345678901234567890123456789012345678901234567890123456789
Ha Ha Ha Ha! HaHaHa! I am a happy person, HaHaHa! I have a million dollar.
0
15
52
15
52
A2 Part(b):
bbb bbb bbbbb
0
4
11
8
10
```

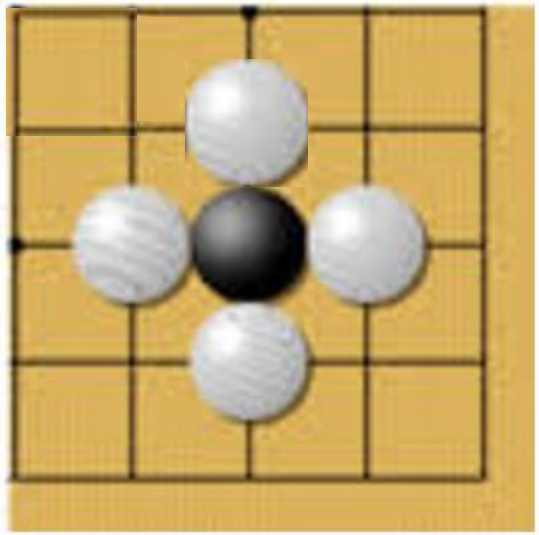
SECTION 1

2-D Traversal Dead Go-Game Stone Detection

Questions

- AP2016– Q3(a, b, c)





Stone Colors:

String B=“B”;

String W=“W”;

String S=“ “;

Single Dead Stone:

1. A stone is dead if it is surrounded by stones of other color.
2. Stones on a side is dead if surrounded by 3 sides.
3. Stones at a corner is dead if surrounded by 2 sides.

We don’t detect the death of connected stones.

Generate a Scanned Map for the Dead Stones

- Game Board Size 9x9.
- public static boolean isDead(String[][] gameboard, int r, int c);
- Input Parameter: a 9x9 Go-Game game board.
- Return: if a stone is dead. (0 <= r < 9, 0 <= c < 9)

```java
import java.util.Scanner;
public class Go{
    public static String B = "B";
    public static String W = "W";
    public static String S = " ";
    public static String[][] gameboard ={
      {S, S, S, S, W, B, W, S, S},
      {S, S, S, S, B, W, S, S, S},
      {S, S, S, S, W, S, S, S, S},
      {S, B, B, S, S, S, S, S, S},
      {S, B, W, B, S, S, S, S, S},
      {W, W, B, S, S, S, S, W, S},
      {W, B, S, S, S, S, W, B, W},
      {B, W, S, S, S, S, B, W, B},
      {W, S, S, S, S, S, S, B, W}
    };
    public static void display(String[][] m){
       System.out.println("  0 1 2 3 4 5 6 7 8");
       for (int i=0; i<m.length; i++){
            System.out.print(i+" ");
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
```

```
  0 1 2 3 4 5 6 7 8
0         W B W
1         B W
2         W
3   B B
4   B W B
5 W W B         W
6 W B         W B W
7 B W         B W B
8 W             B W

Enter row number (999 to exit):6
Enter column number:0
Stone[6,0]=false
Enter row number (999 to exit):999
```

isDead() logic

- The stone itself is not S.
- The stone's any neighbor is not S, and not the same as the stone.
- If a side is board boundary, that side is considered "Surrounded" status.

isDead

```
public static boolean isDead(String[][] board, int r, int c){
    boolean top=false, left=false, bottom=false, right=false;
    String stoneColor = board[r][c];
    boolean isSpace = board[r][c].equals(S);
    if (r==0) top = true;
    else      top = !isSpace && !board[r-1][c].equals(S) && !stoneColor.equals(board[r-1][c]);
    if (c==0) left = true;
    else      left = !isSpace && !board[r][c-1].equals(S) && !stoneColor.equals(board[r][c-1]);
    if (r==board.length-1) bottom = true;
    else                   bottom = !isSpace && !board[r+1][c].equals(S) && !stoneColor.equals(board[r+1][c]);
    if (c==board.length-1) right  = true;
    else                   right  = !isSpace && !board[r][c+1].equals(S) && !stoneColor.equals(board[r][c+1]);
    return top && bottom && left && right;
}
```

Demo Program:

Go.java

Go BlueJ!!!

```java
public class TestGo
{
    public static String B = Go.B;
    public static String W = Go.W;
    public static String S = Go.S;
    public static String[][] gameboard = Go.gameboard;
    public static void display(String[][] m){
       System.out.println("  0 1 2 3 4 5 6 7 8");
       for (int i=0; i<m.length; i++){
            System.out.print(i+" ");
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static void display(int[][] m){
       System.out.println("  0 1 2 3 4 5 6 7 8");
       for (int i=0; i<m.length; i++){
            System.out.print(i+" ");
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static void display(int[][] m){
       System.out.println("  0 1 2 3 4 5 6 7 8");
       for (int i=0; i<m.length; i++){
            System.out.print(i+" ");
            for (int j=0; j<m[i].length; j++){
               System.out.print(m[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static int[][] getDeadMap(String[][] m){
        int[][] n= new int[m.length][m.length];
        for (int i=0; i<m.length; i++){
          for (int j=0; j<m.length; j++){
              if (Go.isDead(m, i, j)) n[i][j] = 1;
              else n[i][j]=0;
          }
        }
        return n;
    }
```

```java
public static void main(String[] args){
  System.out.println("\fGame Board:");
  display(gameboard);
  System.out.println();
  System.out.println("Dead Map:");
  int[][] n = getDeadMap(gameboard);
  display(n);
}
```

```
Game Board:
  0 1 2 3 4 5 6 7 8
0           W B W
1           B W
2           W
3     B B
4     B W B
5 W W B             W
6 W B             W B W
7 B W             B W B
8 W                 B W

Dead Map:
  0 1 2 3 4 5 6 7 8
0 0 0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0
3 0 0 0 0 0 0 0 0 0
4 0 0 1 0 0 0 0 0 0
5 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 1 0
7 1 0 0 0 0 0 0 1 1
8 0 0 0 0 0 0 0 0 1
```

Key Points

- Use Boolean variable to simplify the Boolean expression.
- Setting boundary condition wisely. Be careful about index out of bound exceptions.