



eC Academy

Realize Your Dreams

AP Computer Science A Review

Week 13: Algorithm VI Data Operations

DR. ERIC CHOU
IEEE SENIOR MEMBER

SECTION 1

Reader Head (Iterator)



Questions

- AP2017 – Q2



Hard Disk Reading Head

Class ReaderHead

- Build a ReaderHead class
- Data field: **max** (number of reading addresses)
right (move right?)
index (current reading address)
- Methods: `getIndex()`, `isFacingRight()`,
`changeDirection()`, `reset()`, `next()`, `move(n)`

Methods

- **getIndex():** get the current reading head location
- **isFacingRight():** check if the reading head is facing right.
- **changeDirection():** change the direction of the reading head.
- **reset():** reset to the index==0 and facing right.
- **next():** move one location (right or left).
- **move(n):** move n location (right or left but will stop at ends)

```

1 public class ReaderHead{
2     private int max = 0;
3     private int index = 0;
4     private boolean right = true;
5     ReaderHead(){}
6     ReaderHead(int max){
7         this.max = max;
8         index = 0;
9         right = true;
10    }
11    public int getIndex(){ return index; }
12    public void changeDirection(){
13        right = !right;
14    }
15    public boolean isFacingRight(){
16        return right;
17    }
18    public void move(int n){
19        if (isFacingRight()){
20            while (index < max-1 && n > 0) { index++; n--; }
21        }
22        else{
23            while (index > 0 && n > 0) { index--; n--; }
24        }
25    }

```

```

31 public void next(){
32     if (isFacingRight() && getIndex() == max-1){
33         changeDirection();
34         move(1);
35         return;
36     }
37     if (!isFacingRight() && getIndex() == 0){
38         changeDirection();
39         move(1);
40         return;
41     }
42     move(1);
43 }

```



Class TestReaderHead

- Using both next(), reset() and move(n) to control the reader head.


```
2 public class TestReaderHead
3 {
4     public static int[] data = {10, 20, 30, 40, 50};
5     public static void main(String[] args){
6         ReaderHead r = new ReaderHead(data.length);
7         System.out.print("\f");
8         System.out.println("B1 Part(a):");
9         for (int i=0; i<data.length*3; i++){
10             System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
11             r.next();
12             if (i%data.length==data.length-1) System.out.println();
13         }
14         System.out.println("B1 Part(b):");
15         r.reset();
16         r.move(3);
17         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
18         r.move(3);
19         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
20         r.changeDirection();
21         r.move(3);
22         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
23         r.move(3);
24         System.out.println("Index "+r.getIndex()+" : "+data[r.getIndex()]);
25     }
26 }
```

B1 Part(a):

Index 0: 10

Index 1: 20

Index 2: 30

Index 3: 40

Index 4: 50

Index 3: 40

Index 2: 30

Index 1: 20

Index 0: 10

Index 1: 20

Index 2: 30

Index 3: 40

Index 4: 50

Index 3: 40

Index 2: 30

B1 Part(b):

Index 3: 40

Index 4: 50

Index 1: 20

Index 0: 10

Iterator()

- Serialize the access of a data structure.
- Access sequential access devices.

SECTION 2

Available List



Questions

- AP2016 – Q1



Selection Sort

- Selection Sort using available list.
- Mark the number out if a number is picked.

```

public static ArrayList<Integer> selectionSort(int[] data){
    ArrayList<Integer> a = new ArrayList<Integer>();
    boolean[] b = new boolean[data.length];
    for (int i=0; i<b.length; i++) b[i] = true;
    for (int i=0; i<data.length; i++){
        int min = Integer.MAX_VALUE;
        int index = -1;
        for (int j=0; j<data.length; j++){
            if (b[j]){
                if (data[j] < min)
                { min = data[j];
                  index = j;
                }
            }
        }
        b[index] = false;
        a.add(min);
    }
    return a;
}

```

BlueJ: Terminal Window - Week10

Options

[0, 1, 2, 3, 5, 6, 7, 8, 9]



Hotel class

Build a Hotel Class

Data fields:

- num: number of rooms;
- available: number of available rooms;
- alist[]: available list;

Methods:

- isAvailable(int n): check if a room is available or not.
- isFull(): check if the hotel is full;
- checkIn(int n): check in a room if it is available, make it unavailable. And decrease the number of available rooms.
- checkOut(int n): check out a room if it is not available, make it available. And increase the number of available rooms.


```

1 public class Hotel{
2     private int num;
3     private int available;
4     private boolean[] alist;
5     Hotel(int num){
6         this.num = num;
7         available = num;
8         alist = new boolean[num];
9         for (int i=0; i<alist.length; i++) alist[i]=true;
10    }
11    public boolean isAvailable(int n){
12        if (n<num && alist[n]) return true;
13        return false;
14    }
15    public boolean isFull(){ return available <=0; }
16    public void checkIn(int n){
17        if (isAvailable(n)) {
18            alist[n] = false;
19            available--;
20        }
21    }

```

```

22    public void checkOut(int n){
23        if (!isAvailable(n)) {
24            alist[n] = true;
25            available++;
26        }
27    }
28    public String toString(){
29        String str = "[";
30        for (int i=0; i<alist.length; i++){
31            if (i==0) if (alist[i]) str += "T"; else str += "F";
32            if (i!=0) if (alist[i]) str += " T"; else str += " F";
33            if (i== alist.length-1) str += "];";
34            if (i%10== 9 || i== alist.length-1) str += "\n";
35        }
36        return str;
37    }
38 }

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

```
public class TestHotel
{
    public static void main(String[] args){
        Hotel h = new Hotel(20);
        System.out.print("\f");
        System.out.println("Initialized");
        System.out.print(h);
        h.checkIn(3);
        h.checkIn(6);
        h.checkIn(7);
        System.out.println("After check-ins");
        System.out.print(h);
        h.checkOut(15);
        h.checkOut(6);
        h.checkOut(3);
        System.out.println("After check-outs");
        System.out.print(h);
    }
}
```

```
Initialized
[T T T T T T T T T T
 T T T T T T T T T T]
After check-ins
[T T T F T T F F T T
 T T T T T T T T T T]
After check-outs
[T T T T T T T F T T
 T T T T T T T T T T]
```

SECTION 3

Non-Recurring Set and Occurrence List

Generation of Non-Recurring Set

- Add a word into a word list if the word is not contained by the word list.
- Use the occurrence list to compare and generate the occurrence list.

Non-Recurring Set

```
ArrayList<String> wlist = new ArrayList<String>();  
String[] tokens = str.split(" ");  
for (int i=0; i<tokens.length; i++) {  
    tokens[i] = tokens[i].trim();  
    if (tokens[i].length() !=0 && !wlist.contains(tokens[i])) wlist.add(tokens[i]);  
}
```

You may try to write your own contains() function.

Generation of Occurrence Array (or List)

```
int[] wCount = new int[wlist.size()];  
for (int i=0; i<tokens.length; i++){  
    for (int j=0; j<wlist.size(); j++){  
        if (wlist.get(j).equals(tokens[i])) wCount[j]++;  
    }  
}
```

```
1 import java.util.ArrayList;
2 public class WordList
3 {
4     static String str = "A B C A B C D E F F A B C W S Z K Z";
5
6     public static void main(String[] args){
7         ArrayList<String> wlist = new ArrayList<String>();
8         String[] tokens = str.split(" ");
9         for (int i=0; i<tokens.length; i++) {
10             tokens[i] = tokens[i].trim();
11             if (tokens[i].length() !=0 && !wlist.contains(tokens[i])) wlist.add(tokens[i]);
12         }
13         int[] wCount = new int[wlist.size()];
14         for (int i=0; i<tokens.length; i++){
15             for (int j=0; j<wlist.size(); j++){
16                 if (wlist.get(j).equals(tokens[i])) wCount[j]++;
17             }
18         }
19         System.out.print("\f");
20         System.out.println("Original String="+str);
21         for (int i=0; i<wlist.size(); i++){
22             System.out.println(String.format("Word(%d)=", i)+wlist.get(i)+" happens "+wCount[i]+" times.");
23         }
24     }
25 }
```

Original String=A B C A B C D E F F A B C W S Z K Z

Non-recurring Set=[A, B, C, D, E, F, W, S, Z, K]

Word(0)=A happens 3 times.

Word(1)=B happens 3 times.

Word(2)=C happens 3 times.

Word(3)=D happens 1 times.

Word(4)=E happens 1 times.

Word(5)=F happens 2 times.

Word(6)=W happens 1 times.

Word(7)=S happens 1 times.

Word(8)=Z happens 2 times.

Word(9)=K happens 1 times.

SECTION 4

Selected List



Questions

- AP 2016-Q2(c)



Failed Students


- Remove students who has GPA lower than 2.0 from a school student list and put them into a list for summer class.

```
1 public class Student
2 {
3     private double GPA;
4     private String name;
5     Student(String n, double g){
6         name = n;
7         GPA = g;
8     }
9     public double getGPA(){ return GPA; }
10    public String getName(){ return name; }
11    public String toString(){return "{"+name+", "+GPA+"}"; };
12 }
```

```
1 import java.util.ArrayList;
2 import java.util.Arrays;
3 public class WashingtonHigh
4 {
5     public static ArrayList<Student> getSummerClass(ArrayList<Student> alist){
6         ArrayList<Student> summer = new ArrayList<Student>();
7         for (int i=alist.size()-1; i>=0; i--){
8             if (alist.get(i).getGPA() < 2.0){
9                 summer.add(alist.remove(i));
10            }
11        }
12        return summer;
13    }
14    public static void main(String[] args){
15        ArrayList<Student> slist = new ArrayList<Student>(
16            Arrays.asList( new Student[]{
17                new Student("Amy", 2.85), new Student("Bryan", 3.6), new Student("Carol", 1.46),
18                new Student("David", 4.0), new Student("Ellie", 1.98), new Student("Fanny", 3.2),
19                new Student("Goerge", 1.65), new Student("Harry", 2.78), new Student("Ivan", 2.95),
20                new Student("Jack", 1.8), new Student("Kate", 3.3), new Student("Larry", 2.25),
21                new Student("Mary", 2.03), new Student("Nancy", 1.77), new Student("Omar", 3.6),
22                new Student("Peter", 2.66), new Student("Queen", 1.99), new Student("Robert", 3.98)
23            })
24        );
25        System.out.println(getSummerClass(slist));
26    }
27 }
```



Summer Class Listing

 BlueJ: Terminal Window - Week9

Options

```
[ {Queen,1.99}, {Nancy,1.77}, {Jack,1.8}, {Goerge,1.65}, {Ellie,1.98}, {Carol,1.46}]
```

SECTION 5

Custom Design String Split Function



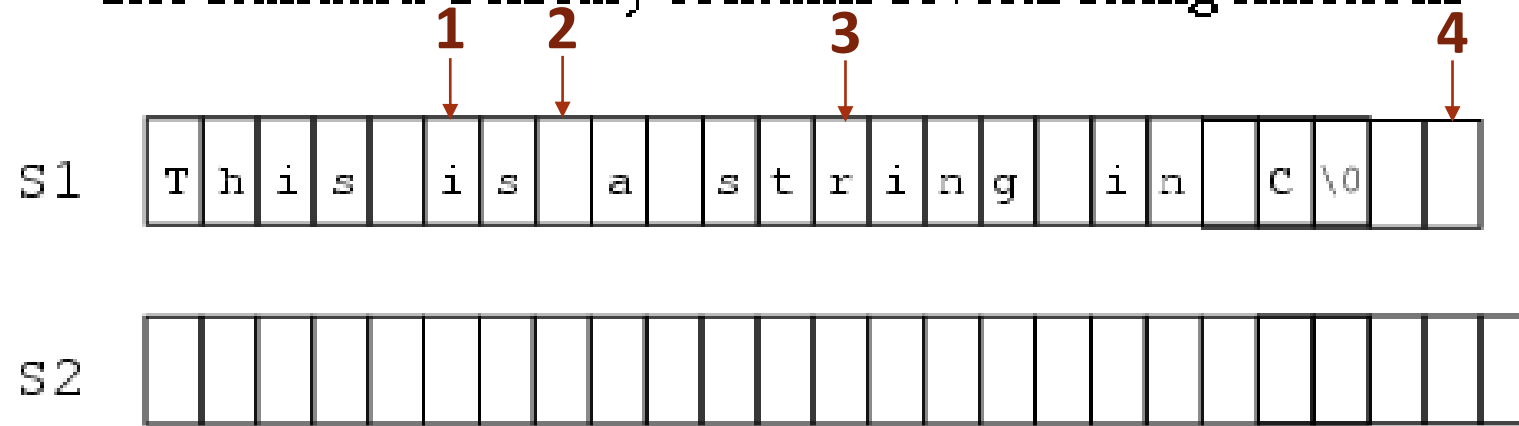
How to split string into tokens?

Detect the start of a token and the end of a string.

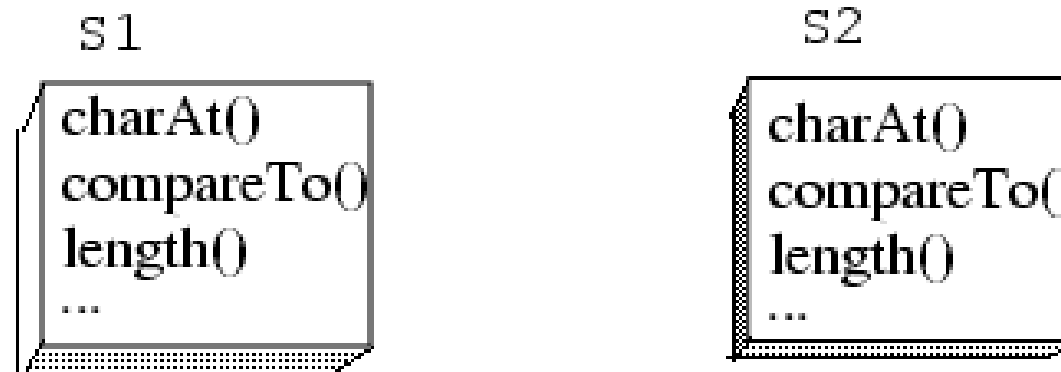
1. isSpace and the next character is not space -> start of a token
2. !isSpace and the next character is space -> end of a token
3. !isSpace and the next character is not a space -> in the middle of a token
4. isSpace and the next character is also a space -> do nothing and it is in space sequence.

A String in C is a nul-terminated sequence of chars

The standard C library contains several string functions



A String in Java is an object with several methods



```

13 public String[] split(){
14     ArrayList<String> a = new ArrayList<String>();
15     if (str.length()==0) return null;
16     if (str.length()==1) return new String[]{str};
17     str = str.trim();
18     int i=0;
19     boolean isSpace = true;
20     String words = "";
21     while (i<str.length()){
22         if (isSpace && !str.substring(i, i+1).equals(" ")) {
23             words+=str.substring(i, i+1);
24             isSpace = !isSpace;
25         }
26         else if (!isSpace && !str.substring(i,i+1).equals(" ")){
27             words+=str.substring(i, i+1);
28         }
29         else if (!isSpace && str.substring(i,i+1).equals(" ")){
30             a.add(words);
31             words="";
32             isSpace = !isSpace;
33         }
34         i++;
35         if (i==str.length() && words.length() !=0) a.add(words);
36     }
37     String[] b = new String[a.size()];
38     for (int k=0; k<a.size(); k++) b[k]=a.get(k);
39     return b;
40 }

```

```

1 import java.util.ArrayList;
2 import java.util.Arrays;
3 public class MySplit
4 {
5     private static String str1 = "    AAA  BB  CC D  EEEEEEE FFFF ";
6     private static String str2 = "AAA  BB  CC D  EEEEEEE";
7     private static String str3 = "AAA  BB  CC D  EEEEEEE ";
8     private static String str4 = "    AAA  BB  CC D  EEEEEEE";
9     private static String str5 = "";
10    private static String str6 = "A";
11    private String str="";
12    MySplit(String s){str = s; }

```

```

42    public static void main(String[] args){
43        MySplit m1 = new MySplit(str1);
44        System.out.println(Arrays.toString(m1.split()));
45        MySplit m2 = new MySplit(str2);
46        System.out.println(Arrays.toString(m2.split()));
47        MySplit m3 = new MySplit(str3);
48        System.out.println(Arrays.toString(m3.split()));
49        MySplit m4 = new MySplit(str4);
50        System.out.println(Arrays.toString(m4.split()));
51        MySplit m5 = new MySplit(str5);
52        System.out.println(Arrays.toString(m5.split()));
53        MySplit m6 = new MySplit(str6);
54        System.out.println(Arrays.toString(m6.split()));
55    }

```

Options

```

[AAA, BB, CC, D, EEEEEEE, FFFF]
[AAA, BB, CC, D, EEEEEEE]
[AAA, BB, CC, D, EEEEEEE]
[AAA, BB, CC, D, EEEEEEE]
null
[A]

```

SECTION 6

Toggler

Binary Toggler

- Initial Condition
- BinaryToggler Kernel Function:

$t = !t;$

- BinaryToggler Transformation

```
BinaryToggler t = new BinaryToggler(true);
t.next();
```

1	public class BinaryToggler	
2	{	true
3	boolean t=false;	false
4	BinaryToggler(boolean x){t = x;}	
5	public void next(){	true
6	t = !t;	false
7	}	
8	public boolean get(){	true
9	return t;	false
10	}	
11	public String toString(){ return ""+t;}	true
12		
13	public static void main(String[] args){	false
14	BinaryToggler bt = new BinaryToggler(true);	true
15	for (int i=0; i<10; i++){	
16	System.out.println(bt);	false
17	bt.next();	
18	}	
19	}	
20	}	

Toggler Kernel

- Initial state of the toggler = -1
- Toggler Kernel


```
state = (state+1) % count; // count is number of states
```
- Toggler Transformation


```
Toggle t = new Toggle(count);
t.toggle()*step+base; // use toggle function like random()
```

```

8 public class Toggler
9 {
10     int state = -1;
11     int count = 2;
12     Toggler(int count){ this.count = count; }
13
14     public int toggle(){ state = (state+1) % count; return state; }
15
16
17     public static void main(String[] args){
18         System.out.println("\fPart (a) 3-way Toggle:");
19         Toggler t = new Toggler(3);
20         for (int i=0; i<10; i++){
21             System.out.println(t.toggle());
22         }
23         int count = 4;
24         int step = 2;
25         int base = 10;
26         System.out.println("\fPart(b) 4-way Toggle: 10, 12, 14, 16 Toggle.");
27         Toggler t1 = new Toggler(count);
28         for (int i=0; i<10; i++){
29             System.out.println((t1.toggle()*step+base));
30         }
31     }
32 }

```

Part (a) 3-way Toggle:

0

1

2

0

1

2

0

1

2

0

Part(b) 4-way Toggle:

10

12

14

16

10

12

14

16

10

12