My WalletHub

"Hello World"

I am Ajay Kavuri (pseudoaj), I am a graduate student here at West Virginia University. The solutions for the programming assignment are presented here.

Talk is cheap show me the code

All the code is hosted on github and here is the **link: https://github.com/PseudoAj/MyWalletHub.** Further, following section presents the important aspects of code, the snippets reflect my thought process and output for the respective problems is also presented.

Tools I have used for the assignment

1. **Operating system:** Ubuntu 14.04

2. **Editor:** Atom
3. **Java:** 1.8.0_91
4. Vagrant(for MySQL)
5. Scotch box(Lamp Stack)
6. Junit4
7. White board and caffine

Solutions

1. Java

1.1 Palindrome

Solution snippet:

```
//method to check for Palindrome
public boolean isPalindrome(){
  //if it is null string it is false
  if (this.checkMeStr.length()>0){//handles null(replaced as
    return this.checkMeStr.equals(new StringBuilder(this.chec
  }
  return false;
}
```

Tests:

1. Case sensitive strings have been verified; ignores character attributes
2. Null and empty strings have been checked, doesn't return them as a palindrome
3. Special characters are handled; they are replaced from the character sequence

How to run:

1. Change directory to `java/Task1/`
2. Compile(assuming junit is installed) and run:

```
javac -cp .:/usr/share/java/junit4.jar PalindromeTest.jav
```

```
java -cp .:/usr/share/java/junit4.jar org.junit.runner.JU
```

Performance: The solution works in a linear time i.e. `O(n)` and depends on the native calls

Notes:

1. All the tests have been verified
2. The null string is replaced with the empty string while initialization

1.2 K-Complimentary

Solution snippet: Two different solutions have been tested:

Iterative:

```
//method to actually check for the k-complimentary pairs
//This is a brute force implementation
//performance: O(n^2); as we are traversing the array twice
public int isKComplimentary(){
  int pairsCount=0;
  System.out.println("k-complimentary pairs are:");
  for(int i=0; i<this.checkAr.length; i++){
    for(int j=0; j<this.checkAr.length; j++){
      if(i<j){//avoids generating interchanged pairs and same
        if(this.checkAr[i]+checkAr[j]==kVal && this.checkAr[i
          //System.out.println(this.checkAr[i]+","+checkAr[j]
          pairsCount++;
        }
      }
    }
  }
  return pairsCount;
}
```

Bidirectional HashMap:

```
  //An efficient method to check for the k-complimentary pairs
  //This uses a hashmap to optimize the performance
  //performance: O(n); as we are traversing only once and retri
  public int isKComplimentaryHashMap(){
    int pairsCount=0;
    System.out.println("k-complimentary pairs using hashing are
    for(int i=0;i<this.checkAr.length;i++){
      int res=this.kVal-this.checkAr[i];
      if(myMap.getKey(res)!=null && res!=this.checkAr[i]){//che
        //System.out.println(this.checkAr[i]+","+this.checkAr[r
        pairsCount++;
      }
    }
    return pairsCount/2;//removing count for interchanged pairs
  }
```

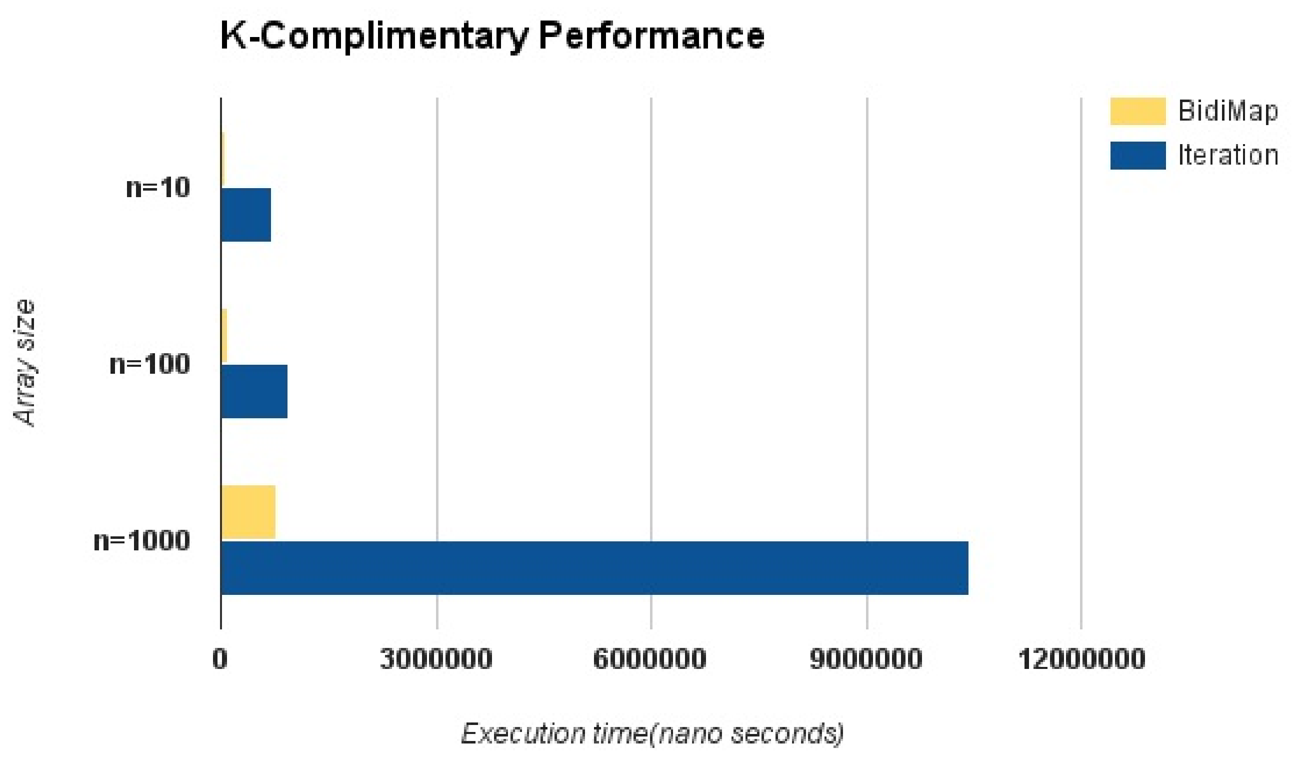
How to run:

Compiling and run:

```
  javac -cp /home/pseudoaj/Downloads/commons-collections4-4.1/c
```

```
java -cp /home/pseudoaj/Downloads/commons-collections4-4.1/c
```

Performance: Following image depicts a simple analytics for varying size of array:



Notes:

1. use of apache commons bi-directional HashMap
2. clean code is presented under java/taks2
3. Full implementation is included in Other too