

RET Computing 2018

Tom Falcone

La Lumiere School

tfalcone@lalumiere.org

La Porte, Indiana

7450

(219) 326

RET Impact on My Classes

- **AP Computer Science A**
 - **Programming projects**
 - **Applications of Computer Science**
- **AP Statistics**
 - **Data Acquisition and Analysis**
- **Intro to Computer Science**
 - **Python / Applications**
 - **RUR (Reeborg)**

Overview of my work...

- Researched Graph Coloring problem relating to my work in scheduling students with classes at my school.
- Overarching theme in all my studies: **Networks!**

My Work

Schedule Allocation:

- The Graph Coloring Problem
 - Graph Theory
 - Brelaz / DSatur Heuristics
 - Apache POI
 - Anonymizing data
 - Limitations



Schedule Allocation

```
1 import java.util.*;
2 import java.io.*;
3 import org.apache.poi.xssf.usermodel.*; // Note: must add ApachePOI jars to required libraries package class
4 import org.apache.poi.ss.usermodel.*;
5
6 /*
7  * This class generates the edge list (currently 'falc.col') in the proper format to match the actual
8  * classes that teachers and students are taking.
9  * After running, recompile Constants.java, GraphReader.java, and GraphColoring.java, then run GraphColoring.java
10 */
11
12 public class EdgeGenerator
13 {
14     public static void main(String[] args) throws Exception
15     {
16         ArrayList<ArrayList<Integer>> nums = new ArrayList<ArrayList<Integer>>();
17
18         ArrayList<Integer> sets = new ArrayList<Integer>();
19         final int STUDENTS = 174; // number of students
20         final int NUMCLASSES = 100; // total classes offered
21         final int PERIODS = 7; // number of class periods per day (must be >= CLASSES)
22         final int TEACHERS = 29; // total number of teachers
23
24         // import data from worksheet
25         FileInputStream file = new FileInputStream(new File("C:\\Users\\TJ\\OneDrive\\Tom's Stuff\\RET Computing Lessons\\graphcoloring-master\\Schedule Anony.xlsx"));
26         XSSFWorkbook wb = new XSSFWorkbook(file);
27
28         XSSFSheet sheet = wb.getSheet("Student Registrations");
29         XSSFSheet sheet2 = wb.getSheetAt(1);
30
31         //XSSFRow row, newRow;
32         XSSFCell cell, cell2;
33
34         // generate periods and add to arraylist
35         for(int i = NUMCLASSES+1; i<NUMCLASSES+PERIODS+1; i++)
36         {
37             sets.add(i);
38         }
39         nums.add(sets);
40
41         // iterate through students and add each Fall class to list, then add each student's list to a list
42         // note: rows and columns are 0 based
43         int count = 2;
44         sets = new ArrayList<Integer>();
45         while(count<964)//sheet.getLastRowNum()
46         {
47             //rows.next();
48             cell = sheet.getRow(count).getCell(4); // student name column
49             //cell2 = sheet.getRow(count).getCell(10);//room assignment
50
51             if(count==2) // this is the first class for the next student, if it's not spring, add it
52             {
53                 //checkVal(cell2,count,sets,sheet);
54                 sets.add(Integer.parseInt(sheet.getRow(count).getCell(7).getRawValue()));
55             }
56
57             else
58             {
59                 if(cell.getRawValue().equals(sheet.getRow(count-1).getCell(4).getRawValue()))
60                 {
61                     //checkVal(cell2 count sets sheet);
62                 }
63             }
64         }
65     }
66 }
67
```

Schedule Allocation

Reading Graph...

Computing Clique...

Maximum Clique Size Found: 9

Vertices in the Clique:

54 48 22 84 2 0 21 7 81

100 milliseconds (excluding I/O).

16 coloring found using DSatur.

Applying Iterated Greedy

Improvement...

Found Better Coloring - 15

Found Better Coloring - 14

Applying Local Search...

Final Coloring of graph possible
with 14 colors.

Colors of Vertices:

3 9 2 12 3 10 8 13 6 7 3 9 5 12 7
1 12 4 14 10 3 14 8 4 5 13 11 6 3
1 14 8 2 11 1 2 7 13 8 10 4 6 3 10
4 1 4 8 1 10 5 14 11 8 7 11 4 10 3
2 13 7 10 6 1 3 6 3 10 14 1 7 4 2
5 4 3 2 4 11 1 12 5 4 6 10 5 11 1
14 1 13 9 14 2 4 1 12 8 3 8 1 4 7
5 6 2 3

Reading Graph...

Computing Clique...

Maximum Clique Size Found: 7

Vertices in the Clique:

27 30 26 25 24 29 31

78 milliseconds (excluding I/O).

7 coloring found using DSatur.

Applying Iterated Greedy

Improvement...

Applying Local Search...

Final Coloring of graph possible with 7
colors.

Colors of Vertices:

1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
1 1 1 5 4 3 1 1 6 2 7 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 5 1
1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 7 6 5 4 3
2 1

Process completed.

My Work

Sorting Algorithms

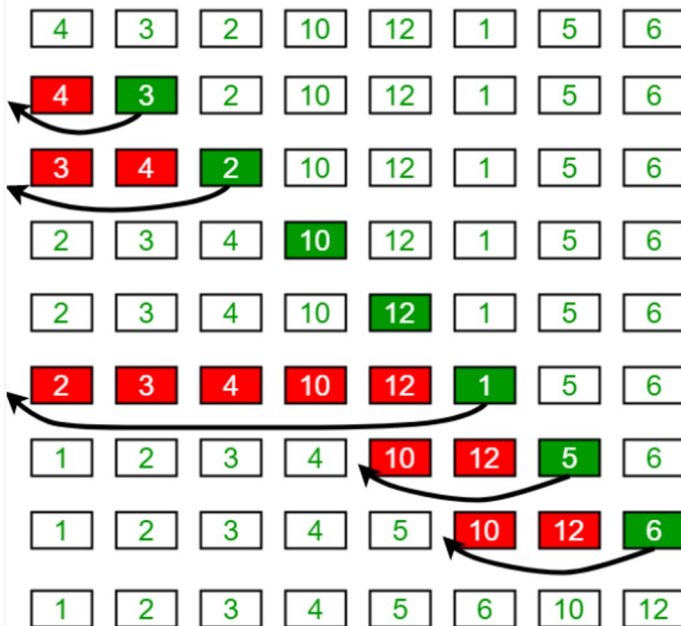
- Review the problem from the computer's perspective
- Analyze Insertion, Selection, Quicksort
 - Speed



Sorting Algorithms



Insertion Sort Execution Example



Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	$O(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
Mergesort	$O(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Timsort	$O(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$
Heapsort	$O(n \log(n))$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(1)$
Bubble Sort	$O(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$O(n)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Selection Sort	$O(n^2)$	$\Theta(n^2)$	$O(n^2)$	$O(1)$
Tree Sort	$O(n \log(n))$	$\Theta(n \log(n))$	$O(n^2)$	$O(n)$
Shell Sort	$O(n \log(n))$	$\Theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
Bucket Sort	$O(n+k)$	$\Theta(n+k)$	$O(n^2)$	$O(n)$
Radix Sort	$O(nk)$	$\Theta(nk)$	$O(nk)$	$O(n+k)$
Counting Sort	$O(n+k)$	$\Theta(n+k)$	$O(n+k)$	$O(k)$
Cubesort	$O(n)$	$\Theta(n \log(n))$	$O(n \log(n))$	$O(n)$

