CHROMATIC NUMBER IN FLAGCALC

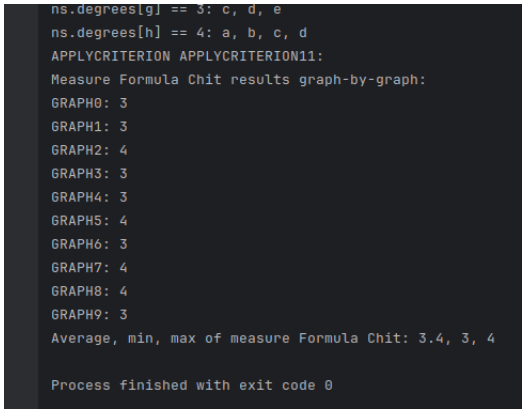
PETER GLENN

A *graph* is a set of vertices V and a symmetric anti-reflexive relation called "edges" $E \subseteq V \times V$. The *chromatic number* "χ" of a graph is the minimum number of "colors" in a set C and a map $V \to C$ that "colors" each vertex with a distinct color, such that no two adjacent vertices have the same color. See, for example, Lovasz, Pelikan, Vesztergombi "Discrete Mathematics: Elementary and Beyond" (2003) section 13.3 and following for an exposition of chromatic number.

1. Chromatic Number and FlagCalc

To compute the chromatic number χ of 10 random graph, run flagcalc with the following command-line argument (see the file "How to Install FlagCalc" for examples of invoking the tool for chromatic number):

```
-r 8 14 10 -a a=Chit all -v graphs allmeas crit
```



Verify each of the ten graphs' chromatic number by hand (one needs the full output, not just this screenshot, to see all ten graph's adjacency matrices).

2. Algorithms for Chromatic Number

Compute the "greedy" chromatic number, run flagcalc with the following command-line argument:

```
-r 8 14 10 -a a=Chigreedyt all -v graphs allmeas crit
```

Date: November 24, 2025.

```
ns.degrees[g] == 3: a, b, c
ns.degrees[h] == 5: a, b, c, e, f
APPLYCRITERION APPLYCRITERION11:
Measure Formula Chigreedyt results graph-by-graph:
GRAPH0: 4
GRAPH1: 3
GRAPH2: 4
GRAPH3: 4
GRAPH4: 3
GRAPH5: 3
GRAPH6: 5
GRAPH7: 3
GRAPH8: 4
GRAPH9: 5
Average, min, max of measure Formula Chigreedyt: 3.8, 3, 5

Process finished with exit code 0
```

Verify each of the ten graphs' chromatic number via the "greedy" algorithm by hand.

3. Compare the two algorithms

Ask the tool for examples of graphs with a different value from the greedy algorithm in comparison to the precise algorithm:

```
-r 8 14 10 -a s="Chigreedyt > Chit" a2=Chigreedyt a2=Chit all -v graphs allmeas crit
```

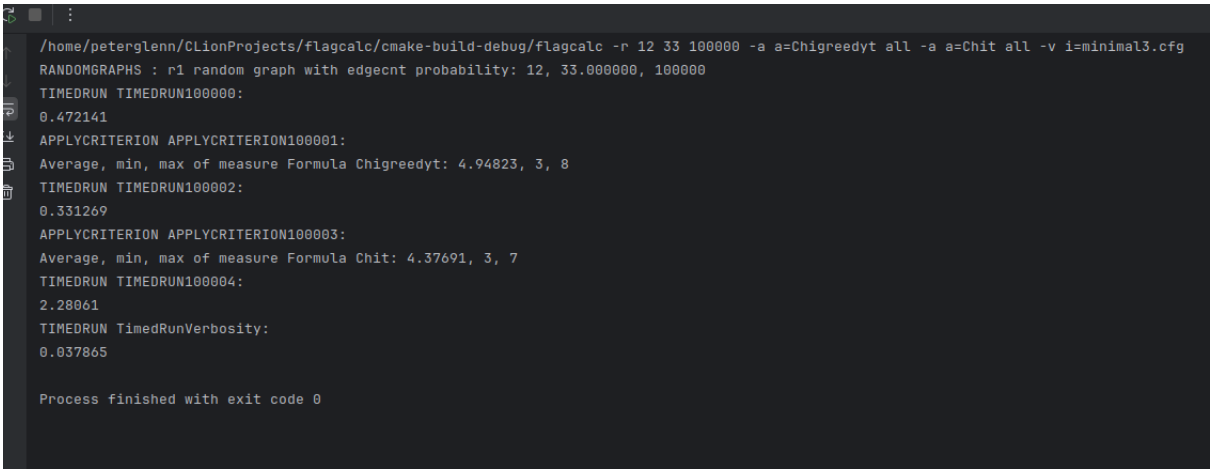
```
f 0 0 0 0 0 0 0 1
g 1 1 0 1 1 0 0 0
h 1 0 1 1 1 1 0 0
[a,b], [a,c], [a,e], [a,g], [a,h]
[b,d], [b,g]
[c,d], [c,h]
[d,g], [d,h]
[e,g], [e,h]
[f,h]
ns.degrees[a] == 5: b, c, e, g, h
ns.degrees[b] == 3: a, d, g
ns.degrees[c] == 3: a, d, h
ns.degrees[d] == 4: b, c, g, h
ns.degrees[e] == 3: a, g, h
ns.degrees[f] == 1: h
ns.degrees[g] == 4: a, b, d, e
ns.degrees[h] == 5: a, c, d, e, f
APPLYBOOLEANCRITERION APPLYBOOLEANCRITERION11:
Criterion Sentence Chigreedyt - Chit > 0 results of graphs:
GRAPH0, number 1 out of 10: 0
GRAPH1, number 2 out of 10: 0
GRAPH2, number 3 out of 10: 0
GRAPH3, number 4 out of 10: 1
GRAPH4, number 5 out of 10: 0
GRAPH5, number 6 out of 10: 0
GRAPH6, number 7 out of 10: 0
GRAPH7, number 8 out of 10: 0
GRAPH8, number 9 out of 10: 0
GRAPH9, number 10 out of 10: 0
result == 0: 9 out of 10, 0.9
result == 1: 1 out of 10, 0.1
APPLYCRITERION APPLYCRITERION12:
Measure Formula Chigreedyt results graph-by-graph:
GRAPH3: 4
Average, min, max of measure Formula Chigreedyt: 4, 4, 4
APPLYCRITERION APPLYCRITERION13:
Measure Formula Chit results graph-by-graph:
GRAPH3: 3
Average, min, max of measure Formula Chit: 3, 3, 3

Process finished with exit code 0
```

Can you explain how the greedy algorithm differs from the exact algorithm?

4. Justify using the greedy algorithm

In large datasets, see the speed difference between Chit and Chigreedyt:
`-r 12 33 100000 -a a=Chigreedyt all -a a=Chit all -v i=minimal3.cfg`



Please write out the respective algorithms in pseudo-code, and try to explain this seven-fold speedup!