Greedy Color Completion

- Given
  - Intervals I = { I1 ⊏ … ⊏ I2k },
  - ω(I) ⩽ k,
  - c′ : prefix⊏ I(Ik) → [k] be a proper k-coloring on prefix ⊏ I(Ik)

- Task is to assign color to next k intervals

- Observation
  - Let I = {I1 ⊏ … ⊏ Im} be a multiset of unit intervals. If ω(I) < m, then the extremal intervals are disjoint, i.e., I1 < Im.

Greedy Color Completion

- Algorithm
  - Let J = { Il,Il+1,...Ik }, intervals that intersects Ik
  - Number of elements in J = k-(l-1)
  - Coloring of J requires k-(l-1) colors.
  - Now color next l-1 intervals with the other (l-1) colors, i.e. infix ⊏ I(Ik+1, Ik+l−1)
  - Now set c(Ii) = c(Ii−k) for i ∈ {k + l,...,2k} because by observation 1 intervals Ii and Ii−k do not intersect.

- Hence we colored 2k intervals from given k coloring.

```
PS I:\My Drive\GithubBTP\implementation\Color Completion> g++ .\Greedy.cpp
PS I:\My Drive\GithubBTP\implementation\Color Completion> .\a.exe
Enter the chromatic number: 9
Enter the intervals: 1 2 3 4 4.2 4.4 4.6 4.8 4.9 5 5.05 5.06 5.07 5.5 6 7 8 9
Enter the colors of first k intervals: 5 4 6 7 8 2 3 9 1
Colors assigned to the intervals:
1:      5
2:      4
3:      6
4:      7
4.2:    8
4.4:    2
4.6:    3
4.8:    9
4.9:    1
5:      4
5.05:   5
5.06:   6
5.07:   7
5.5:    8
6:      2
7:      3
8:      9
9:      1
```

Modulo Color Completion

- Given
  - n Intervals
  - $\omega(I) \leqslant k$,
  - Coloring of starting K intervals
  - K intervals assumes proper coloring

- Task is to assign color to every interval

- Algorithm
  - Apply Greedy Color Completion Algorithm on next K intervals recursively till no intervals are left.

```
PS I:\My Drive\GithubBTP\implementation\Color Completion> g++ .\Modulo.cpp
PS I:\My Drive\GithubBTP\implementation\Color Completion> .\a.exe
Enter the number of intervals: 13
Enter the chromatic number: 5
Enter the intervals: 1 2 3 3.2 3.4 3.6 3.8 4 4.5 4.8 6 7 8
Enter the colors of first k intervals: 4 1 2 3 5
Colors assigned to the intervals:
1:      4
2:      1
3:      2
3.2:    3
3.4:    5
3.6:    1
3.8:    4
4:      2
4.5:    3
4.8:    5
6:      1
7:      4
8:      2
```