Week 12

Friday: Graph Coloring*

12.1 Vertex Coloring

Definitions:

A k-coloring of graph G is a labeling $f : V(G) -> S$ where $S = \{c_1, c_2, c_3, ..., c_k\}$ is a set of k-colors. A coloring is proper if adjacent vertices have different labels.
Example: Given a set of colors, S where $S = \{R, G, B, Y\}$, color a graph G given below with suitable colors so that no adjacent vertices are colored with the same color. χ_G

Figure 12.1: Color the graph G

Coloring-1 :

Figure 12.2: Graph G colored with 2 colors

*Lecturer: Dr. Anand Mishra. Scribe: Tejaswee A (M21CS064).

Coloring-2 :

Figure 12.3: Graph G colored with 2 colors

Chromatic Number ($\chi(G)$): The least value of k (minimum number of colors) such that G is proper k-colorable.

12.2 Greedy Coloring Algorithm

The Greedy Coloring algorithm was devised to cater the need of proper coloring of graphs relative to vertex ordering $V(G) = \{v_1, v_2, v_3..., v_n\}$.

Steps involved in Greedy Coloring algorithm for graph coloring are:

1. Select any arbitrary vertex and color it.
2. Iterate the process as mentioned for all the $V - 1$ vertices
   (a) select any vertex from the pool of non-colored vertices
   (b) color it with the least numbered color that has not been considered or used on any previously colored vertices adjacent to it
   (c) if the adjacent vertices to vertex v are colored with the previously used colors then assign a new color to it.

Example: Consider a graph given in Figure-4 having 7 vertices and a set of colors $S = \{R, G, Y, C, B, W\}$ where the colors are indexed at 1, 2, 3, 4, 5, 6 and 7 index values respectively starting from $index = 1$.

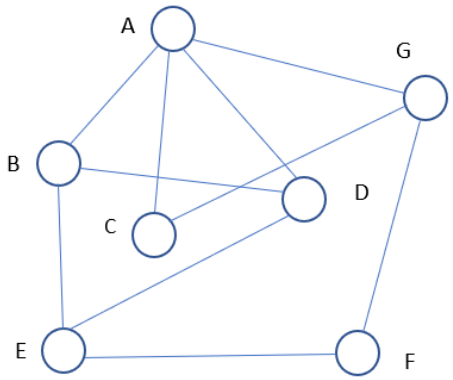


Figure 12.4: Color G using Greedy Coloring Algorithm

Step-wise implementation of the algorithm:

1. Let us select an arbitrary vertex A and color it with the color of the least numbered index which is Red.

2. Iterate the process as mentioned for all 6 vertices

   Iteration 1:

   (a) Now, let us select vertex F in random fashion out of the pool of non-colored vertices.

   (b) It is not adjacent to vertex A hence, can be colored with Red (as shown in Figure-5).

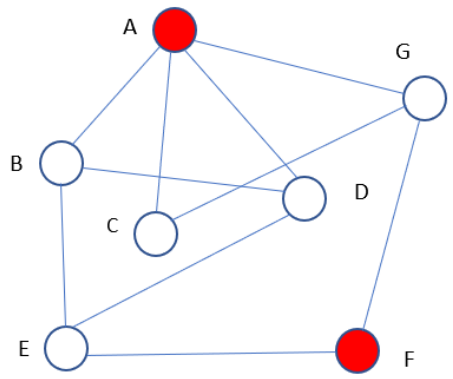


Figure 12.5: Color A and F vertices with Red color

Iteration 2 :

(c) Select B and color it with the next least numbered index i.e G (Green) as Red can not be used being the vertex adjacent to A which is already colored with Red.
Further iterations:
item Select C, D, G and F in the order and color with G, Y, Y and B respectively which will yield the solution as shown in Figure-6.

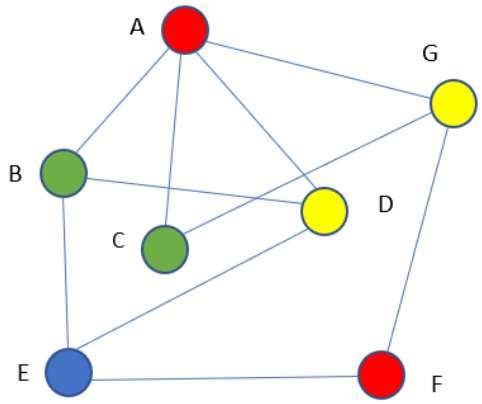


Figure 12.6: Graph G after applying Greedy Coloring algorithm

Definitions:

Clique: Cliques are the induced complete sub-graphs with a subset of vertices in graph G with every pair of distinct vertices being adjacent to each other.

Clique Number ($\omega(G)$): The number of vertices in a maximum clique in graph G is termed as Clique Number or Clique Size of G.

Relation between chromatic number ($\chi(G)$)and Clique Number ($\omega(G)$) is given as:

$$\boxed{\chi(G) >= \omega(G)}$$

Problem: Prove that $\chi(G)$ of a graph G is $max(\chi(G_1), \chi(G_2), \chi(G_3), ..., \chi(G_k))$ where $G_1, G_2, G_3, ..., G_k$ are k components of the graph.

Proof:
Consider a graph G having 4 different components with the chromatic number of each component being 8, 4, 3 and 2 for the components G_1, G_2, G_3 and G_4 respectively as shown below.

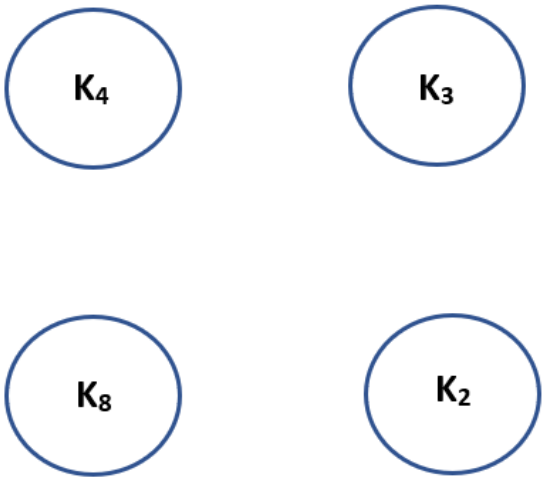


Figure 12.7: Graph G containing 4 components

Here, the chromatic number of the graph G becomes 8 since the colors required to color K_8 may be used by K_4, K_3 and K_2 which form a subset of the colors used to color K_8. Hence, the maximum of all chromatic numbers of components of graph G is the chromatic number of the entire graph.

12.3 Bounds on Chromatic Number

Trivial Upper Bound: $\chi(G) <= |V(G)|$. This situation arises when the graph is fully connected. A complete graph has all the vertices adjacent to each other making the chromatic number equal to the number of vertices in the graph.

Trivial Lower Bound: $\chi(G) > 0$ It is applicable for all the graphs wherein every graph has atleast one vertex leading to satisfying the above condition.

Therefore, in terms we have, $1 <= \chi(G) <= |V(G)|$ which can be more specifically formulated as

$$\boxed{1 <= \chi(G) <= \Delta(G) + 1}$$

where $\Delta(G)$ represents the maximum degree a graph has.

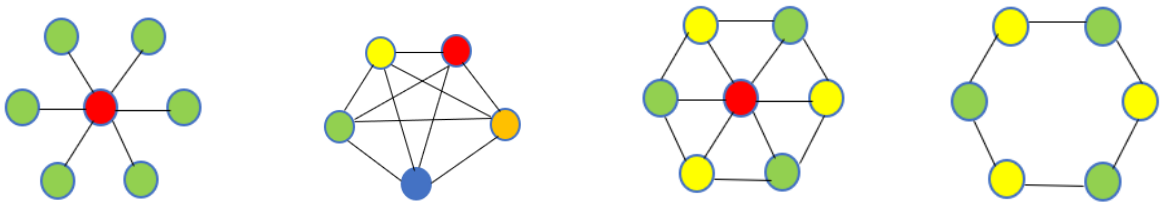
As mentioned, chromatic number of graph $\chi(G)$ will be less than or equal to the total number of vertices. Degree of a vertex signifies how many other vertices the node is connected to i.e. the number of adjacent vertices. Hence, for a complete graph where $\chi(G) <= |V(G)|$, we have $|V(G)| = \Delta(G) + 1$ which gives birth to the above equation.

Let us verify whether the equation holds true for various types of graphs.

Table 12.1: Chromatic number Table

Sr. No.	**Type of graph**	$\Delta(G)$	$\chi(G)$ **wrt.** $\Delta(G)$	**Actual** $\chi(G)$
1	Star Graph	$n-1$	$<= n$	2
2	Complete Graph	$n-1$	$<= n$	n
3	Wheel Graph	$n-1$	$<= n-1$	$3; n = odd$
				$4; n = even$
4	Cycle Graph	2	$<= 3$	$3; n = odd$
				$2; n = even$

Respective figures for the graphs mentioned in Table 12.1:



((a)) Star Graph ((b)) Complete Graph ((c)) Wheel Graph ((d)) Cycle Graph

Figure 12.8: Graphs for Table 12.1

12.3.1 Proposition

Welsh-Powell Bound: If a graph has degree sequence $d_1 >= d_2 >= d_3 \ldots >= d_n$ then, $\chi(G) <= 1 + max_i\ min\{d_i, i-1\})$

Proof: Greedy algorithm is applied on the vertices arranged in non-increasing order of degrees. By the time we color vertex v_i, it has at most $min\{d_i, i-1\}$ neighbours which are colored. Therefore, v_i can be colored with is at most $1+ min\{d_i, i-1\}$. To obtain the upper bound on maximum colors used we maximize over i

The upper bound given in proposition 12.3.1 is always at least as good as $\chi(G)<=\Delta(G)+1$ since the bound specified in the mentioned proposition is at most $\Delta(G) + 1$.

Week 12

Saturday: Interval Graphs*

12.1 Introduction

Definitions:

Interval Graph: A family of intervals when assigned to the vertices so that the vertices are adjacent if and only if the intervals corresponding to the nodes intersect is called interval representation of graph G. An **interval graph** is a graph having interval representation.

((a)) Interval representation ((b)) Corresponding Interval Graph

Figure 12.1: Graphs for Table 12.1

Example: Let us consider a situation where there are a set of activities happening in the lecture halls of Lecture Hall Block at the campus. Each activity has a start and finish time stamps as given below in Table 12.2.

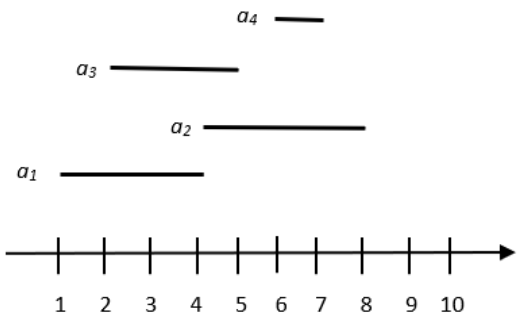
*Lecturer: Dr. Anand Mishra. Scribe: Tejaswee A (M21CS064).

Table 12.1: Activities' Interval Timestamps

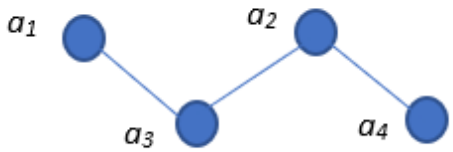
Activity	Start Time	Finish Time
a_1	1	4
a_2	4	8
a_3	2	5
a_4	6	7

Ponder upon:

Two questions arise: *a*) How many activities can be conducted in one room with no overlapping timestamps? *b*) How many minimum number of rooms is required to organize all the activities?



((a)) Interval representation



((b)) Corresponding Interval Graph

Figure 12.2: Graphs for Table 12.1

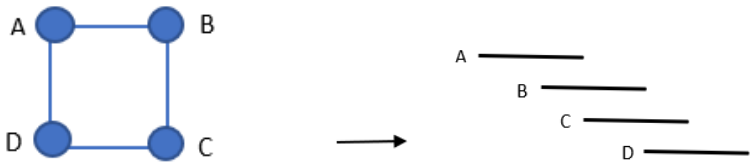
We will come across the answers to these questions in further sections.

NOTE: Graph coloring process is different in Interval graphs since **not every graph is an interval graph.**

Let us see some of the examples.

1. Is C_4 an interval graph?

   Ans: No



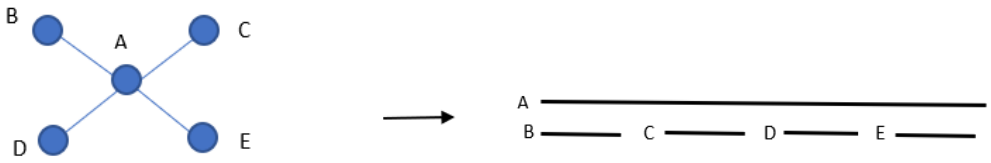
((a)) C_4 graph ((b)) Interval Representation of C_4

Figure 12.3: Interval graph of C_4

   Explanation: C_4 is not an interval graph there exists an edge between vertex A and vertex D but there is no overlapping of edges in the interval representation of the same graph. It is not possible to construct C_4 satisfying all the conditions of interval representation.

2. Is star graph an interval graph?

   Ans: Yes



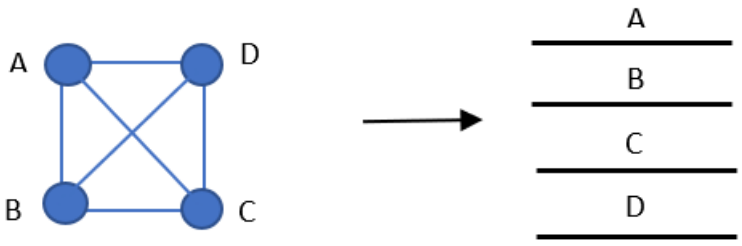
((a)) Star graph ((b)) Interval Representation of star graph

Figure 12.4: Interval graph of star graph

   Explanation: Given star graph has equivalent interval representation which manifests a longer edge of vertex A and 4 other non-overlapping line segments which have edges from A and not connected to any other vertex in the graph.

3. Is complete graph an interval graph?

   Ans: Yes



((a)) K_4 graph ((b)) Interval Representation of K_4

Figure 12.5: Interval graph of K_4 graph

Explanation: K_4 graph proves to be an interval graph since its equivalent interval representation can be drawn with overlapping edges (line segments) indicating all the vertices are adjacent to each other.

In general, we can say that **a graph having no induced subgraph as C_4 and its complement graph having transitive orientation is known to be an interval graph. Example:**

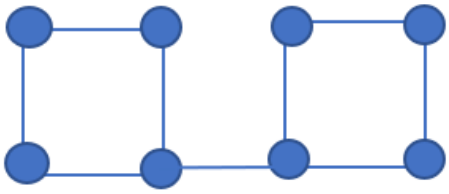


Figure 12.6: Color G using Greedy Coloring Algorithm

12.2 Graph coloring of Interval graphs

Interval graphs are colored in a different way than how the other graphs are colored because as mentioned, not all graphs are interval graphs. Hence, coloring of interval graphs is done in greedy manner where there are different heuristics to approach the graph coloring.

For the problems such as scheduling maximum number of events in a room such that every event is organised properly with no overlapping timestamps, there are a few approaches

used in greedy algorithm.

Three significant and popularly used heuristics are:

1. **Shortest Events First:** Events or activities which are scheduled for a shorter period of time are given priority over other events. The event scheduled for the shortest duration is preferred, then the second shortest and so on with no overlapping timestamps.

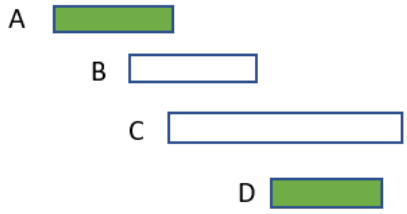


Figure 12.7: Using Shortest Events First heuristic (successful)

From the above figure, we can say that the maximum number of events that can be scheduled is only 2 and the Shortest Events First heuristic also gives the count as 2 with events A and D scheduled.

This works better for the events scheduled for shorter duration whereas other events are queued up. Shortest Events First approach fails or is inefficient when such a situation (mentioned below) arises.

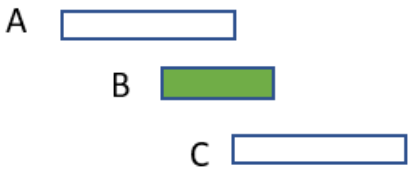


Figure 12.8: Using Shortest Events First heuristic (unsuccessful)

In the given Figure 12.8, event B is the shortest and so can be scheduled first but events A and C are non-overlapping and still can not be scheduled as they are overlapping with event B.
Therefore, only one event can be scheduled out of 3 whereas maximum number of events that can be scheduled is 2 (A and C).

2. **Early Start Events First:** This heuristic considers the first event at priority and selects next possible events that begin as early as possible.

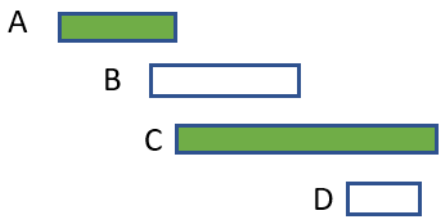


Figure 12.9: Using Early Start Events First heuristic (successful)

In Figure 12.9, 2 events namely, event A and event C are scheduled which is the maximum number of activities performed in a room with no clashing timestamps. Counter example:

Figure 12.10: Using Early Start Events First heuristic (unsuccessful)

In the above shown representation, events B and C can be scheduled marking a maximum count of the events that can be scheduled but only event A is scheduled using Early Start Events First approach. Hence, this heuristic doesn't work efficiently in every given situation.

3. **Earliest Finish Time First:** Using Earliest Finish Time First heuristic we schedule those events first which are completed early (with the least finish time) and takes up the next non-overlapping event having early finish time and continues until the conditions are satisfied.

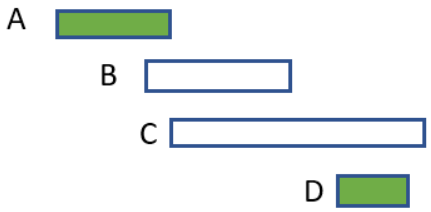


Figure 12.11: Using Earliest Finish Time First heuristic

In Figure 12.11, only 2 events can be scheduled in a room without clashing with any other events Earliest Finish Time First (EFT) heuristic selects events A and D to be scheduled making it a count of 2, and accounts for the maximum number.
This approach is very efficient and gives relevant results as compared to the other two heuristics. It solves the counter examples given in Figure 12. and 12. as well.

((a)) ((b))

Figure 12.12: Solution for the Figures 12.8 and 12.10 using EFT heuristic

Now, let us get back to the questions that were put forward in the beginning. Considering Figure 12.2 and Table 12.1, let us try to answer the two questions.

a) How many activities can be conducted in one room with no overlapping timestamps?
Ans: Using an efficient heuristic as discussed, Earliest Finish Time First, we can schedule a maximum of 2 events namely, event a_1 and event a_4 in a room.

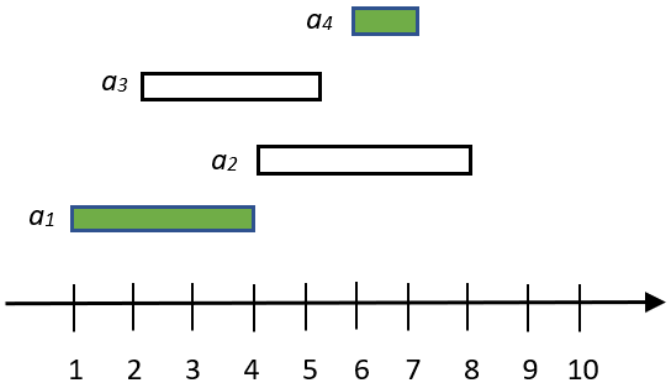


Figure 12.13: Applying EFT on Figure 12.1(a)

b) How many minimum number of rooms is required to organize all the events?
Ans: Proceeding with the Earliest Finish Time First heuristic, coloring the graph (b) of Figure 12.2 in greedy approach will require 3 distinct colors: a_1 is colored with a random color say, red, next in the queue is a_4 to be colored with red. Now, a_2 and a_3 are colored with different colors as they are adjacent to each and cannot use red as a_1 is adjacent to a_3 and a_4 is adjacent to a_2.

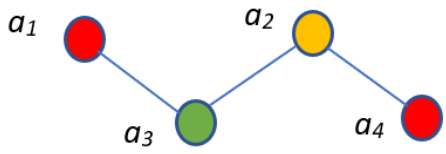


Figure 12.14: Graph coloring using EFT heuristic

In real, we can color the graph with 2 distinct colors as shown in Figure 12.15

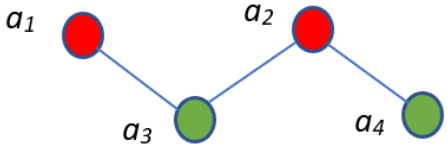


Figure 12.15: Graph coloring with usual approach

Hence, the chromatic number ($\chi(G)$) of the graph is 2 and the minimum number of rooms required to organize all the events is 2.

Therefore, we can conclude that the heuristic which gives better results to one scenario cannot necessarily provide appropriate results to another.

Exercise:

Solve the graph given in Figure 12.1 using EFT to find

a) how many activities can be conducted in one room with no overlapping timestamps. **Ans:** Using EFT heuristic, activities a, c and f are scheduled successfully in one room with no clashing of events.

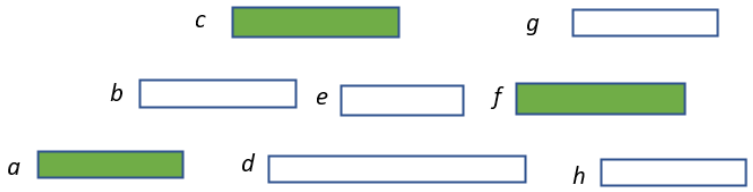


Figure 12.16: Using EFT heuristic to solve Figure 12.1

b) How many minimum number of rooms is required to organize all the events? **Ans:** Randomly assigning any color say, red to the nodes a, c and f, we can choose any color from the pool of unused colors. Let us color non-adjacent vertices b, e and h which are also non-overlapping with green and vertices d and g with orange. This is the minimum possible number to color the given graph hence, $(\chi(G)) = 3$.

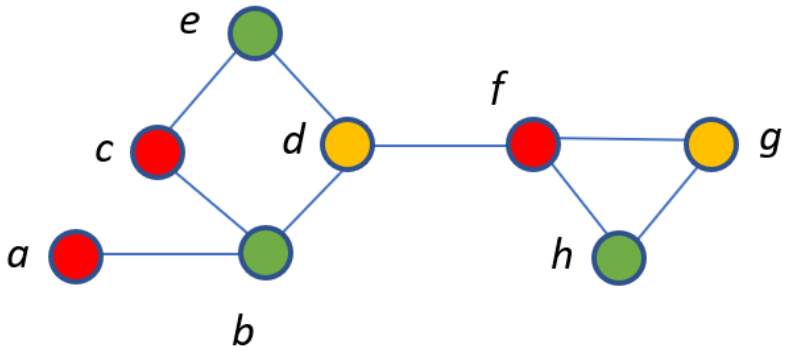


Figure 12.17: Using EFT to find $\chi(G)$