Week 12

Lecture 23 and Lecture 24*

Graphs are diagrammatic representations composed of edges and vertices. The study of the relationships between the edges and vertices is the purpose of graph theory. To be able to understand the various applications of graphs, a thorough understanding of the structural properties and nature of different types of graphs is crucial.

Graph coloring and Interval graphs are different ways to solve real world problems where scheduling of different events in real time are required. The find a wide range of applications and are useful in solving a variety of optimization problems which aim to get the best possible results.

12.1 Graph Coloring

In map coloring, labels are assigned to vertices-here the numerical values of the labels are not of any importance. They are referred to as colors to indicate that they might be elements belonging to any set. Thus, we can say that graph coloring takes its name from map-coloring.

In k-coloring of graphs, the coloring is said to be proper if adjacent vertices have different colors. Thus, the number of colours required to color adjacent vertices differently tells us how many sets the vertices can be divided into, on the basis of color. So, based on this we can say that a given graph is 2-colorable only if it is bipartite. Thus, we can say that k-colorable and k-partite have the same meaning.

For a given graph, the chromatic number of the graph is the least k for which the graph is k-colorable.

12.2 Greedy Coloring

The greedy coloring relative to a vertex ordering V(G) is obtained by coloring vertices in the order where the the smallest-indexed color not already used is assigned to its lower-indexed

*Lecturer: Anand Mishra. Scribe: Ritwika Das.

neighbors.

The Greedy Algorithm for graph coloring consists of the following steps:

- Choose any vertex and color it
- For the remaining vertices, do the following:
  - Choose any non-colored vertex
  - Color it with the lowest numbered color that has not been used on any previously colored vertex adjacent to it
  - If the case is such that all the previously used colours appear on the adjacent vertex, then assign a new color to it.

This algorithm does not guarantee the perfect solution. It provides the global solution, or what could be the closest possible to the perfect solution.

12.3 Clique Size

As we know, a clique is a fully connected sub-graph. There exists a relation between the clique size and the chromatic number of a given clique. We see that a clique of size 4 will have chromatic number 4. A clique of size 3 will have a chromatic number of 3 and so on for cliques of various sizes. Also, for every graph, the chromatic number of the graph is always either greater than or equal to the clique size.

Proof: The chromatic number of a graph G is the maximum of the chromatic numbers of the components of the graph - G1, G2, G3, ... , Gk. If there are k components in a graph, each having a different chromatic number, then the chromatic number of the graph shall be equal to the highest of the chromatic number of its components. This is so because, it is possible that the other components having lesser chromatic number contain colours which are already included in the component having largest chromatic number , that is all the colours have been covered in the component having largest chromatic number. Thus, we can say that the chromatic number of a graph is equal to the largest chromatic number of its components.

12.4 Bounds on Chromatic Number

Trivial Upper Bound The chromatic number of a given graph will always be less than or equal to the number of vertices, never more.

In a complete graph, the chromatic number will be equal to the number of vertices in the graph.

Trivial Lower Bound The chromatic number of a given graph will be grater than zero.

In a null graph, where there are no edges, the chromatic number shall be 1 as all the disconnected vertices can be colored with the same color. Thus, we can say that the minimum possible chromatic number shall be 1.

Thus, it can be stated that the chromatic number of a graph lies between 1 and the number of vertices in a graph.

Proposition: The chromatic number of a graph is less than or equal to 1 grater than the maximum degree of the graph This situation arises when in a graph, all neigh ours of a node having degree d have different colours. In that situation, the total colors required to color the nodes shall be equal to the number of neighbours plus one for the central node which is connected to the surrounding nodes.

So, in a star graph having n nodes, the maximum degree of the graph is n-1. Similarly, in a wheel graph, the maximum degree is n-1. For or a cycle, the maximum degree is 2 and for a complete graph it is equal to n. Following the proposition above, we can say that the chromatic number of a star graph is less than or equal to n. The same applies to a wheel graph. For a cycle, the chromatic number is less than or equal to 3, and that of a complete graph is less than or equal to n+1.

To verify the above proposition, we compare the deductions from the proposition above to the actual values. We see that as proposed, the chromatic number of a star graph is equal to 2, which is less than n. That of a complete graph is n and for a wheel graph it is either 3 or 4 and for a cycle it is 2 if the cycle is even and 3 for an odd cycle. Thus, the above proposition is verified.

Welsh-Powell Bound In 1967 Welsh and Powell Algorithm introduced in an upper bound to the chromatic number of a graph . It provides a greedy algorithm that runs on a static graph. The vertices are ordered according to their degrees, the resulting greedy coloring uses at most $max_i mind(x_i) + 1, i$ colors, at most one more than the graph's maximum degree. This heuristic is called the Welsh–Powell algorithm.

The Welsh Powell Algorithm follows the following steps:

- Find the degree of each vertex.
- List the vertices in order of descending valence i.e.valence degree(v(i)) is greater than or equal to degree(v(i+1)).
- Colour the first vertex in the list.
- Go down the sorted list and color every vertex not connected to the colored vertices above the same color then cross out all colored vertices in the list.
- Repeat the process on the uncolored vertices with a new color-always working in descending order of degree until all in descending order of degree until all vertices are colored.

12.5 Interval Graphs

If we consider a scenario where there are multiple events which overlap in the time space. Then, a graph representing this relationship between the events, where an edge exists between the events (nodes) only if the overlap in the time space, is known as an interval graph.

All graphs are not interval graphs. A graph which has no induced sub-graphs (such as in a cycle) and its complement graph has transitive orientation, is an interval graph.

Based on the above, we see that a Cycle cannot be an interval graph as it does not satisfy the above condition. However a star graph as well as a complete graph do form interval graphs as they do satisfy the above statement.

12.6 Scheduling problems

We consider various heuristics when addressing a scheduling problem. The question arises that what would be the best heuristic to ensure that more events which are scheduled close to each other or overlapping with each other can be fulfilled. Some of the heuristics are as follows:

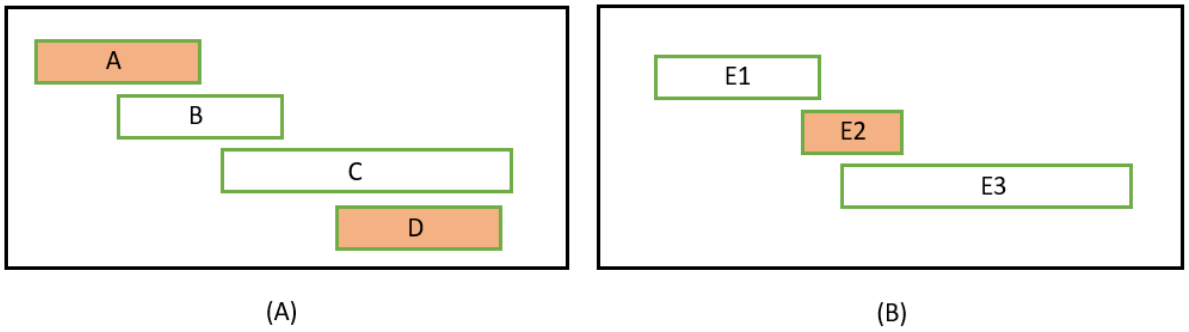

Figure 12.1: Heuristic 1: Select the shortest events first .

- Heuristic 1: Select the shortest events first. In the Figure 12.1(a), we see that if we follow this heuristic, then ideally event A and event D can be fulfilled. However, this does not work best for ever case of events. In the counter example in Figure 12.1(b), we see that only 1 event can be fulfilled as the other two are overlapping with this event (shortest event).

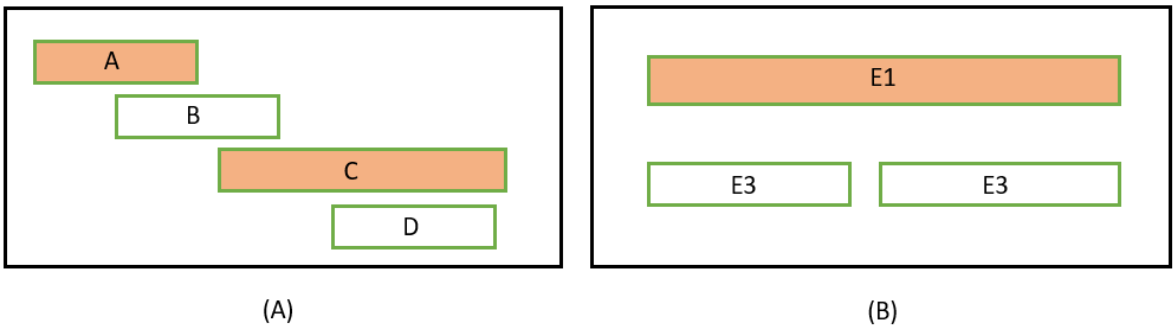

Figure 12.2: Heuristic 2: Start with the first event and then proceed with the next event which starts as early as possible.

- Heuristic 2: Start with the first event and then proceed with the next event which starts as early as possible. In the Figure 12.2(a), we see that if we follow this heuristic, then ideally event A and event D can be fulfilled. However, once again, in the counter example provided in Figure 12.2(b), we see that only one such event can be fulfilled. So this heuristic too does not stand good for all events.
- Heuristic 3: Select earliest finish time first. Here as shown in Figure 12.3(a) we see that the heuristic provides satisfactory results as compared with the previous two heuristics as is visible in Figure 12.3(b). Even for the counter examples in the previous two cases, Heuristic 3 provides better results.

Figure 12.3: Heuristic 3: Select earliest finish time first .

Final note: When we are given a series of events, then two questions arise - If provided only one room, how can we maximise the number of events which can be held, and, how

many rooms would be required to ensure that all events take place. The firmer question can be solved by using Interval graphs as has been discussed. By selecting the right heuristic we can try to make more events take place. For the latter question, the solution is a graph colouring problem. By following the right steps we can find out how many colours, and equivalently how many rooms would be required to make sure all the events take place.

Some applications of graph coloring include

- Map Coloring
- Register Allocation
- Mobile Radio Frequency Assignment
- Making time table, etc.
- Bipartite Graph Checking

****