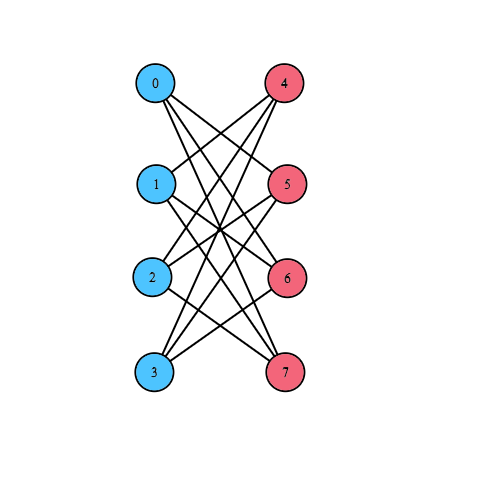
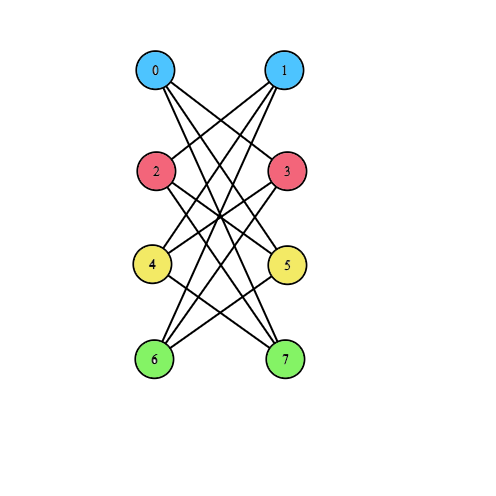
Text, letter

Description automatically generated

Let  the vertices with the largest degree  in the graph G. When we color , at most colors are forbidden to use if it happens in the case that all of its neighbors are colored differently. Therefore, one more color is required to color , requiring at most  colors. The greedy algorithm can arrive at this worst case when it randomly colors all of the neighbors of differently.

* The upper bound of the chromatic number is 

Example where the chromatic number is small but the greedy algorithm uses many colors is the bipartite graph. It is a graph with two disjoint and independent sets of vertices, called U and V. Every edge of the bipartite graph will always connect one vertex from U and one vertex from V. In this bipartite graph, if we colors all of the vertices in each set U and V subsequently, we only need two colors. However, if the greedy algorithm somehow alternate coloring on the vertices between set U and V, then the greedy algorithm unnecessarily needs at most n colors, where n is the size of the sets

Bipartite graph colored with optimum Bipartite graph colored by the greedy algorithm

Text, letter

Description automatically generated  
Improved greedy algorithm

(1) Order the vertices according to their degrees in non-increasing order. In other words, after sorting the vertices, we have:   
(2) Pick each vertex subsequently after they have been sorted

(3) Give it the smallest possible value in c

Prove of the upper bound of the improved greedy algorithm:

For each , we observe that he vertexhas at most  neighbors among the previous sorted vertices . According to part (a), we have  and  is determined by the vertex with the most neighbors

* 

Plugging into the formula, we have:   
(proven)

A screenshot of a computer

Description automatically generated with medium confidence

Greedy algorithm for interval graphs

(1) Sort the intervals in decreasing order according to their starting time:   
(2) for i in 1:n

Assign to  the smallest color that has not been assigned to the previous starting vertices that intersect the duration of (works like the First Greedy Algorithm)

Proof of the optimality

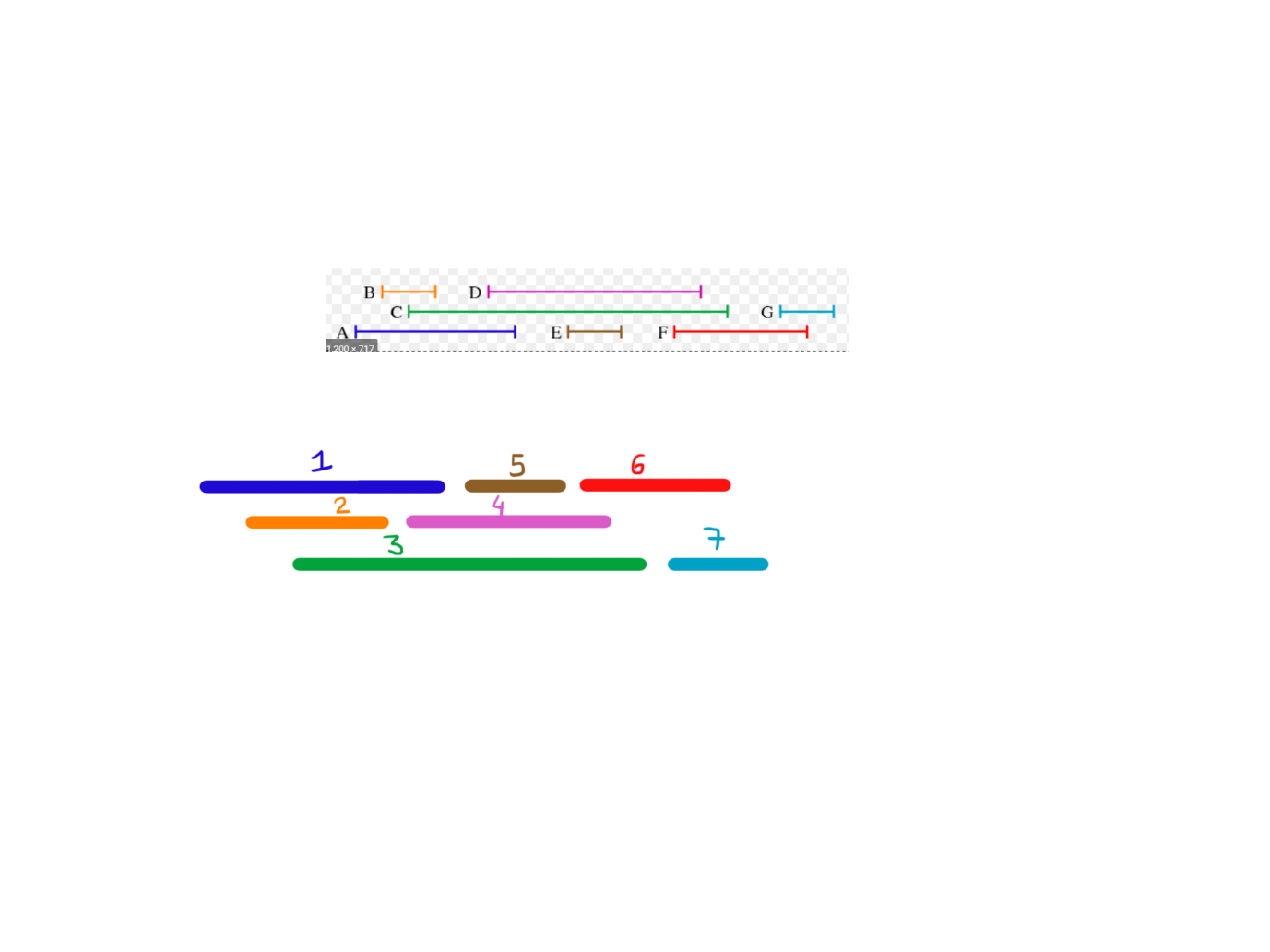
Let k be the chromatic number at the end of the algorithm. At the moment when the color k is used, we know that all previous vertices use colors from 1 to k -1. In the interval, the algorithm only allows a vertex to occupy a color from its start to finish time, which indicates that the set of vertices in the schedule contains k mutually intersecting vertices. This means that k is a lower bound of the chromatic number and since our algorithm uses exactly k colors, it uses the least number of colors

* This algorithm is optimum for interval graphs

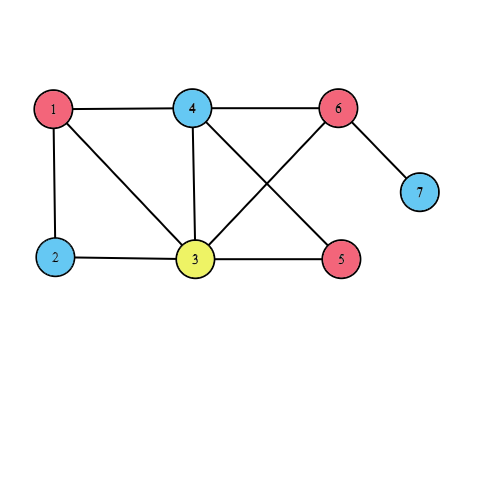
A screenshot of a computer

Description automatically generated with medium confidence

Suppose that the durations in the schedule are given as follows:

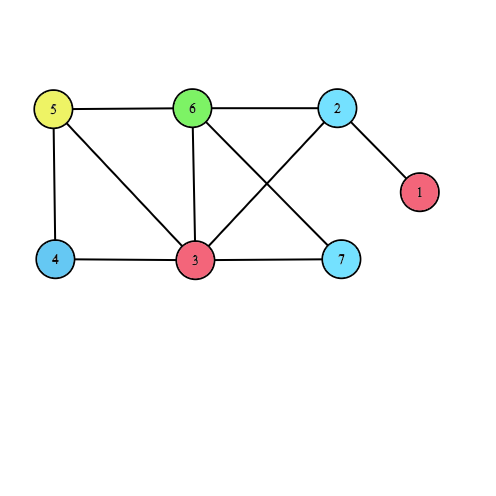


The numbers are given according to their starting time. From the interval graph, we sort the vertices based on their starting time. Then we proceed to color like the first greedy algorithm:



First, vertex 1 starts first so it is colored red. At vertex 2, it intersects vertex 1 so it needs blue. For vertex 3, it intersects both vertex 1 and 2 so it needs yellow. For vertex 4, it only intersects vertex 1 and 3 so the smallest color is red again. This works until we color all of the graph.

* This interval graph needs only 3 colors

However, if we only use the first greedy algorithm without sorting, this color ordering yields 4 colors, which is not optimum for this graph => First greedy algorithm is not optimum for interval graphs  


Text

Description automatically generated

We can observe that an edge can only connect one vertex from B and one from R. In order to create a cycle, first we have go from a vertex in B to one in R, then from R to another vertex in B. Next, we go from B to another vertex in R and finally return back to the original vertex in B, taking 4 edges. The cycle can only be lengthened by alternating between the two groups in a pair of edges. Therefore, the bipartite can only have even length cycles. An even-length cycle can be colored with only 2 colors by alternation. If it is not a cycle, then it is a simple line with even length which also requires 2 colors. Therefore, for every bipartite graph, , where one color only colors vertices in B and another only colors vertices in R.

Correctness of the algorithm: We can use the induction proof to verify that the algorithm returns the optimum, which is 2 colors.

1. Base case:   
   From the start, we know that all color-degree is 0. Suppose we pick one vertex from B and color it with color #1. When we go to R, we update the color-degree of all vertices connected to the one in B to 1, because there is only one colored neighbor. We color one vertex from R with degree 1 with the a new color #2. Now we have one vertex from B with color #1 and one vertex from R with color #2
2. Induction property:   
   At step number k, we assume that all previously colored vertices in B are exclusively colored #1 and all vertices in R colored #2 => The base case is correct
3. Induction step

At step number k + 1, let the vertex in consideration be  and we need to color it. Due to the greedy algorithm, we know that the vertex  has at least one colored neighbor. As we know, the neighbors of a vertex only belong to a different group than the current vertex. According to the induction property, all vertices are colored uniformly

* All neighbors of vertex  are colored with one color, so it means the vertex only needs to use the other alternating color. As we know, this alternating color is the same as the one used for the neighbors of neighbors of vertex , so this implies that vertex  is colored the same as its current group it belongs to.   
  At stage k + 1, the induction property still holds, concluding the proof

Therefore, this algorithm is optimum for bipartite graph

Text

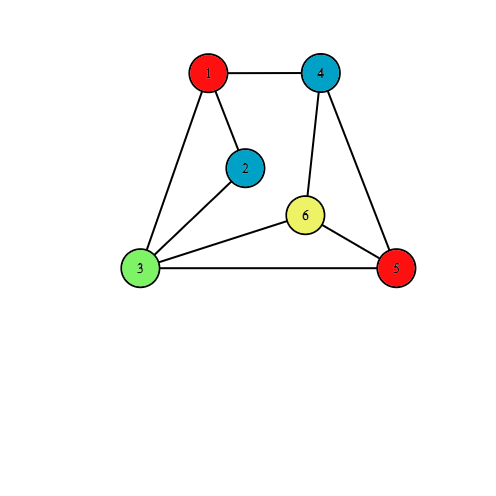
Description automatically generated

In this graph, if we colors by this order with the First Greedy Algorithm, we will only need 3 colors, which is the chromatic number of this graph

Diagram

Description automatically generated with low confidence

However, if we use the greedy algorithm for the Bipartite, the graph can be colored as follows:



First, all nodes color-degree are 0. We color the first vertex as red. The color degree of vertices 2, 3, 4 are updated as 1. We randomly choose vertex 2 and color it blue. Now color degree of vertex 3 is 2, which is highest, so we proceed to color it as green and update the color degree of vertex 5 and 6 as 1. Now all vertices 4,5,6 have equal color degree of 1, so we randomly pick 4 and color it as blue. Now the color degree of 5 and 6 are 2. We randomly pick vertex 5 and colors it as red. Finally, due to unfortunate coloring configuration, the last vertex 6 has to resort to a fourth color which is yellow

* Greedy algorithm for the bipartite graph is not optimum for general graphs