DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 6651 Algorithm Design Techniques
Fall 2025
Project Description
Due: Monday, December 1

## Contents

## 1 Project Specification

In this project, you will implement some Greedy algorithms for colouring online graphs and study their performance.

**What is an online graph?** An online graph is a graph that reveals its vertices and edges in an online fashion. Specifically, at each time step (i.e., at each iteration), one vertex together with its edges with the *existing* vertices are revealed.

**The online graph colouring problem:** you are given an online graph presented to you in an online fashion as described above, and at each step, after the arrival of the next vertex together with its edges with existing vertices, you need to give that vertex a proper colour (i.e., no two vertices of an edge have the same colour) [1]. Note that once a colour is given to a vertex, it is permanent and can not be modified afterwards. Naturally, the goal here for an online colouring algorithm that uses the least number of colours possible for the entire revealed graph.

The two greedy-type algorithms you will implement are the FirstFit and CBIP (Coloring Based on Interval Partitioning) algorithms. We will demonstrate these two algorithms using the example online graph with 10 vertices, shown in Figure 1 with its optimal 2-colouring given.

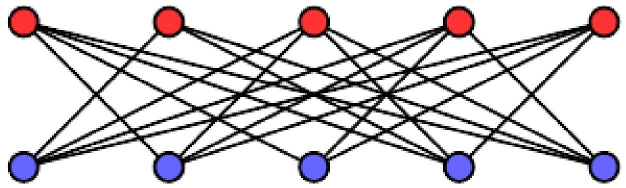

Figure 1: Example graph with optimal 2-colouring shown.

The **FirstFit algorithm** [2]: upon the arrival of a vertex, colour it by the smallest natural number as long as this number fits—i.e., the smallest number that has not been used to colour its existing neighbours.

Figure 2 gives a step-by-step demonstration of the FirstFit algorithm run on example graph. In this example, the FirstFit algorithm uses 5 colours, but this graph is actually two-colourable.

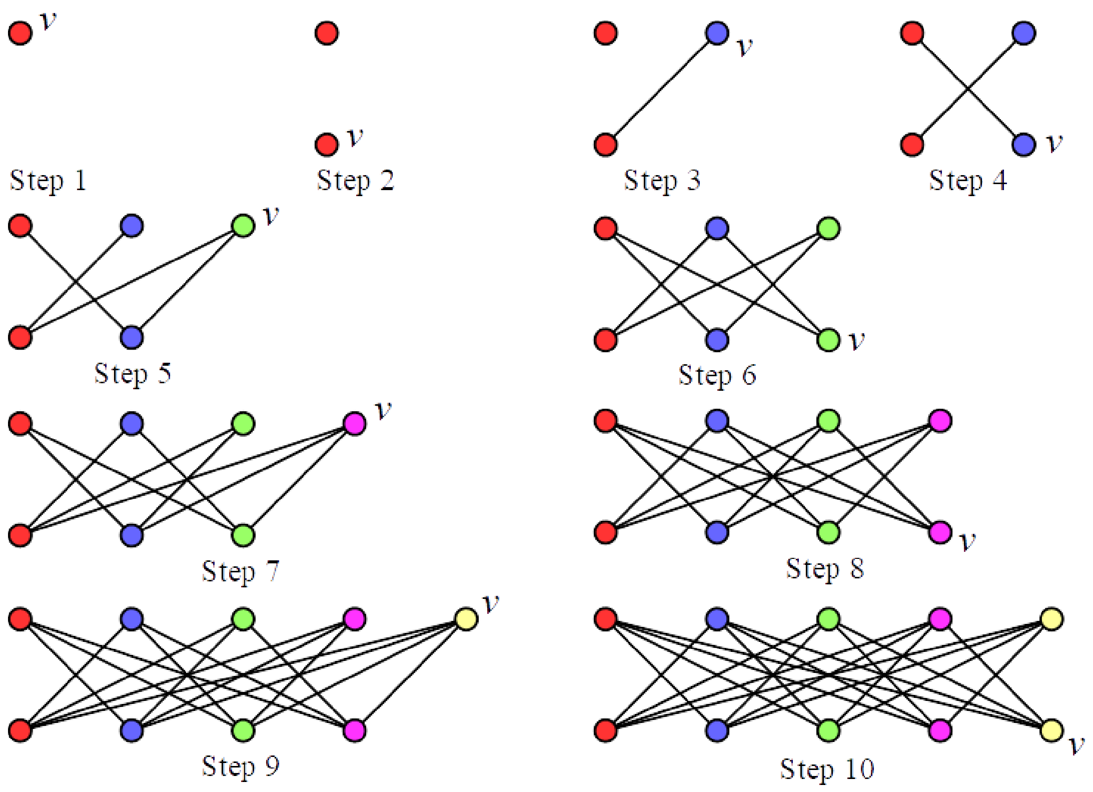

Figure 2: FirstFit: Sample execution on example online graph.

The **CBIP algorithm** [3] (assuming the online graph has (offline) chromatic number k): upon the arrival of a vertex $v$, find a $k$-partition (equivalently, a $k$-colouring) of vertices of the current partial graph into $k$ independent sets. See Figure 3 for an example of a 3-partition of a graph. Let $A$ denote the set where $v$ belongs to, then, colour $v$ by the smallest natural number that has not been used by vertices not in $A$ (i.e., choose the smallest numbered colour that is not being used by a vertex *outside* of $A$).

For example, Figure 4 gives a step-by-step demonstration of the CBIP algorithm run on the same

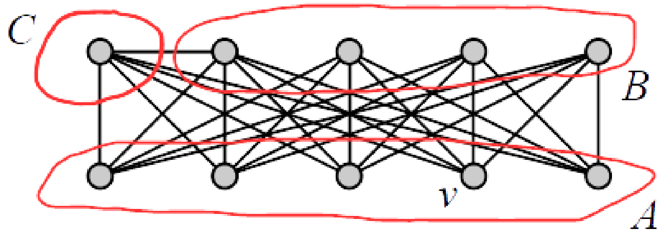

Figure 3: CBIP: partitioning current graph into independent sets

two-colourable online graph as in Figure 1. The difference starts appearing at Step 6: the CBIP chooses to use the colour magenta (colour number 4), instead of green (colour 3) as chosen by FirstFit, to colour the 6th vertex. Let us examine the details of CBIP; what happens is as follows: upon the arrival of the 6th vertex $v$, see Figure 3, the CBIP algorithm first computes a two-partition of the current partial graph into 2 independent sets. In this case, the upper part of the graph is one independent set, and the lower part the second independent set. Since $v$ belongs to the lower independent set, the set $A$, CBIP chooses the smallest number colour that is not present in the upper part. Hence, CBIP colours $v$ as magenta (colour number 4). Similarly, if the current partial graph is 3-colourable, CBIP will first compute a three-partition into 3 independent sets, such as the partition shown in Figure 3, and then CBIP will colour vertex $v$ by the smallest number that is not present in $B$ and $C$.

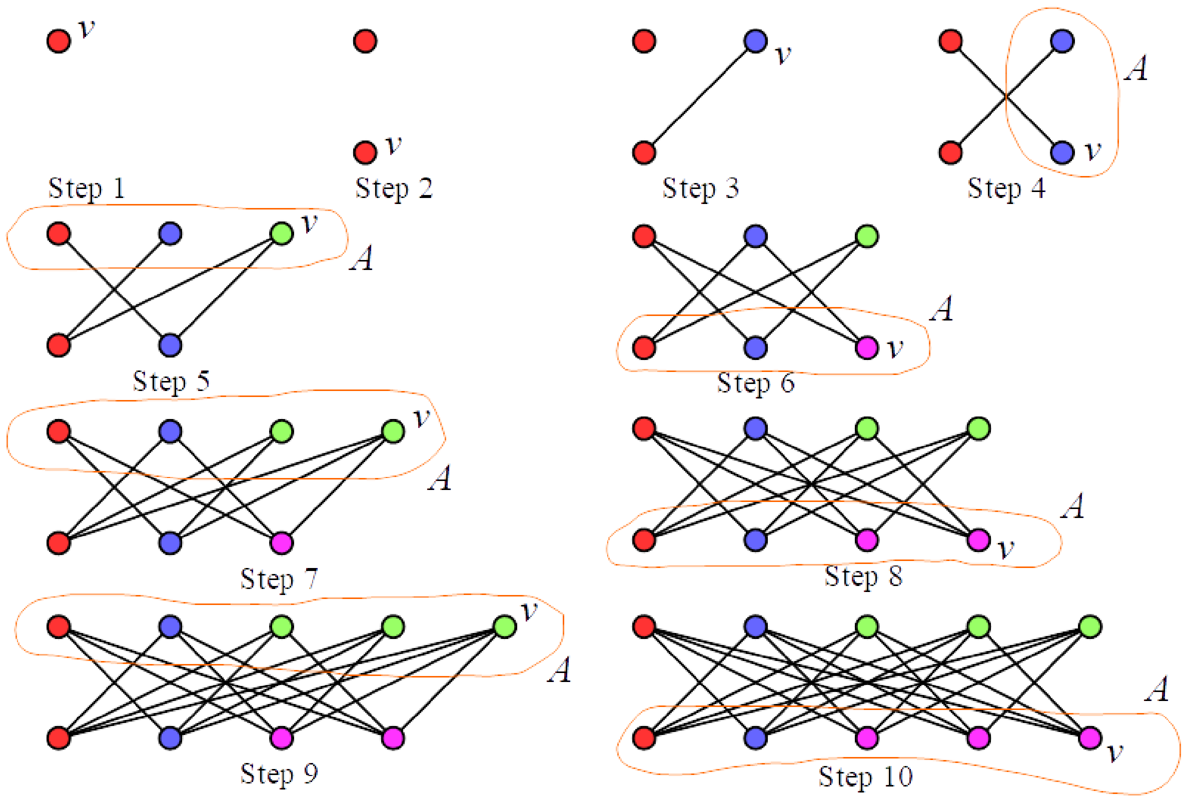

Figure 4: CBIP: Sample execution on example online graph.

In this project, as a team, you will do the following:

1. You will implement a graph generator to create random $k$-colourable online graphs, which will be stored in an external file using the ASCII-based EDGES format.
2. You will implement the FirstFit and CBIP algorithms.
3. You will perform an empirical study of the greedy FirstFit on $k$-colourable online graphs for $k \in \{2, 3, 4\}$ and CBIP algorithms on 2-colourable online graphs.
4. You will implement your choice of a heuristic to attempt to improve the FirstFit algorithm.
5. The characteristics of the randomly generated graphs and the metrics for all these algorithms

will be entered into tables, as shown below.

You may use a programming language of your choice, such as Python, Java, or C++, to create a functional implementation of the required algorithms.

## 1.1 Required implementations:

- Random Online $k$-colourable Graph Generator

  For this project, you will need to randomly generate online $k$-colourable graphs.

  A resource you can access is:

  https://pypi.org/project/kcol-graph-gen/

  For example, the following pseudocode based on the above resource generates an undirected graph $G = (V, E)$ of $n$ vertices to form an online $k$-colourable graph where $n >> k$.

```
GENERATE_ONLINE_KCOLOURABLE_GRAPH($n, k, p$)
  // $n$ is number of vertices, $k$ is number of independent sets in generated graph,
  // $p$ is the probability that an undirected edge $(u, v)$ is added
  Define a set $S$ of $k$ sets: $S = \{S_1, S_2, \ldots, S_k\}$
  // assign vertices to sets in $S$ such that no set $S_i$ is empty
  for each vertex $u \in V$ do
      $counter \leftarrow 1$
      if $counter < k$ then                    ▷ Put each of the first $k$ vertices in its own set $S_i$
          Assign $u$ to set $S_{counter}$
          $counter \leftarrow counter + 1$
      else                    ▷ Distribute the rest of the vertices randomly to sets in $S$
          $rInt \leftarrow$ a uniform random integer between 1 and $k$ (i.e., $rInt \in [1, k]$)
          Assign $u$ to set $S_{rInt}$
  // randomly assign edges between independent sets in $S$
  for each $S_i \in S$ do
      for each $v \in S_i$ do
          for each $S_j \in S$ do
              if $(i \neq j)$ then
                  // Add at least one edge between each vertex in $S_i$ to a vertex in $S_j$
                  $u \leftarrow$ a uniform random vertex from $S_j$
                  if $((u, v) \notin E)$ && $((v, u) \notin E)$ then
                      $E \leftarrow E \cup \{(u, v), (v, u)\}$
          for each $S_j \in S$ do
              if $(i \neq j)$ then // Add rest of edges between $S_i$ and $S_j$ according to $p$
                  for each $u \in S_j$ do
                      $rand \leftarrow$ a uniform random number in $[0..1]$
                      if $rand < p$ then
                          if $((u, v) \notin E)$ && $((v, u) \notin E)$ then
                              $E \leftarrow E \cup \{(u, v), (v, u)\}$
```

  Generate $N$ graphs for each set of $n$ and $k$ values (detailed instructions below on how to choose these parameters) and store the graphs in an EDGES format file. This is an ASCII readable edge list format storing a graph as node pairs with no data, e.g., the directed edge $(1,2)$ is stored on one line of the file as:

```
1 2
```

- Greedy algorithms for colouring online $k$-colourable graphs

  Below are the pseudocode for the algorithms that you need to implement to find a colouring of online $k$-colourable graphs. Some implementation details are not given (e.g., finding the connected component of a graph) that you will need to figure out yourself.

  The colours will be represented by natural numbers, $\mathbb{N}$. E.g., 1 represents red, 2 represents blue, etc.

  Algorithm 1: FirstFit

```
FirstFit($G$)
  // Input: $G = (V, E)$ - Current online graph
  define $c(V)$                                    ▷ stores colouring of $V$
  initialize $c(V)$ to $nil$                       ▷ no colours set yet
  define $\sigma([1..n]) \leftarrow V$             ▷ presentation order of vertices, $\sigma : [1..|V|] \to V$
  uniformly randomize order of $\sigma$            ▷ randomize online order of vertices
  $i \leftarrow 1$
  define vertex $v$, set of vertices $N$
  while $i \leq |V|$ do
      $v \leftarrow \sigma(i)$
      $N \leftarrow$ all neighbours of $v$ in $V \cap \sigma[1..i-1]$   ▷ include only current revealed vertices
      $c(v) \leftarrow \min(\mathbb{N} \setminus c(N))$              ▷ $c(N)$ the set of colours of vertices in $N$
      $i \leftarrow i + 1$
  return $c$
```

  Algorithm 2: CBIP

```
CBIP($G$)
  // Input: $G = (V, E)$ - Current online graph
  define $c(V)$                                    ▷ stores colouring of $V$
  initialize $c(V)$ to $nil$                       ▷ no colours set yet
  define $\sigma([1..n]) \leftarrow V$             ▷ presentation order of vertices, $\sigma : [1..|V|] \to V$
  uniformly randomize order of $\sigma$            ▷ randomize online order of vertices
  $i \leftarrow 1$
  define vertex $v$, set of vertices $N$
  while $i \leq |V|$ do
      $v \leftarrow \sigma(i)$
      $CC \leftarrow$ connected component of $v$ in $G \cap \sigma[1..i-1]$ ▷ include only current revealed vertices
      Partition vertices of $CC$ into $A$ and $B$ such that all edges go between $A$ and $B$ only
              and $v \in A$
      $c(v) \leftarrow \min(\mathbb{N} \setminus c(B))$              ▷ $c(B)$ the set of colours of vertices in $B$
      $i \leftarrow i + 1$
  return $c$
```

## 1.2 Simulations and Results

*Simulations I*

Define the following metrics to be computed (to be presented in tables; see below):

- $n$: number of nodes in graph $G = (V, E)$, or $n = |V|$
- $k$: the value used to generate the online $k$-colourable graphs
- $N$: number of generated graphs created for computing average competitive ratio.
- Competitive ratio: The competitive ratio of an algorithm $Alg$ on an online graph $G$ is defined as follows:
$$\rho(Alg, G) = \frac{\text{number of colors used by } Alg \text{ on } G}{\chi(G)}$$
where $\chi(G)$ denotes the chromatic number of $G$
- Average Competitive ratio: The average competitive ratio of an algorithm $Alg$ on $N$ online graphs of $G$ with the same $n$ and $k$ is defined as follows:
$$\overline{\rho(Alg)} = \frac{\sum_{N \text{ graphs in } G} \rho(Alg, G)}{N}$$
- Standard Deviation of Competitive ratio: The standard deviation of the competitive ratios of an algorithm $Alg$ on $N$ online graphs of $G$ with the same $n$ and $k$ is defined as follows:
$$SD(\rho(Alg)) = \text{standard deviation of } \rho(Alg, G) \text{ over } N \text{ graphs of } G$$

Perform an empirical study on the above parameters and metrics of the FirstFit algorithm on $k$-colourable online graphs for $k \in \{2, 3, 4\}$. And perform a similar empirical study of the CBIP algorithm on 2-colourable online graphs. While performing theses studies, consider the choice of any parameters (e.g., $p$ for the online $k$-colourable graph generator).

Fill in a table similar to Table 1. Table 1 is only for demonstration purposes, you should choose your own parameters that work. Also, in this table, there are only 6 data points (i.e., values of $n$) for each $k$—of course you want to have as many as possible.

*Analysis*

Consider the results you found in the *Simulations I* part. You may try to plot your data and try to fit the data by some simple function (such as: polynomial functions of low degree, $\sqrt{x}$, $\log x$, etc.); or if no simple function fits the data, plot it and draw some observation.

It would also be nice if one could do an empirical study on the average competitive ratio of CBIP on $k \in \{3, 4\}$, however, this would be computationally impractical. Do some research to understand this, and briefly explain why one can do an empirical study for $k = 2$ but cannot for $k = 3, 4$.

*Simulations II*

For this part of the project, you are to propose the use of a heuristic to (possibly) improve the performance of FirstFit. The heuristic can address the online ordering of the vertices. E.g., instead of a random ordering, the vertices can be sorted by degree (number of neighbours) in a non-increasing order (this modifies the definition of "online"). Alternatively, the heuristic can modify the FirstFit greedy algorithm itself.

| Algorithm | k | $n$ | $N$ | $\overline{\rho(Alg)}$ | $SD(\rho(Alg))$ |
|---|---|---|---|---|---|
| FirstFit | 2 | 50 | 100 | | |
| FirstFit | 2 | 100 | 100 | | |
| FirstFit | 2 | 200 | 100 | | |
| FirstFit | 2 | 400 | 100 | | |
| FirstFit | 2 | 800 | 100 | | |
| FirstFit | 2 | 1600 | 100 | | |
| FirstFit | 3 | 50 | 100 | | |
| FirstFit | 3 | 100 | 100 | | |
| FirstFit | 3 | 200 | 100 | | |
| FirstFit | 3 | 400 | 100 | | |
| FirstFit | 3 | 800 | 100 | | |
| FirstFit | 3 | 1600 | 100 | | |
| FirstFit | 4 | 50 | 100 | | |
| FirstFit | 4 | 100 | 100 | | |
| FirstFit | 4 | 200 | 100 | | |
| FirstFit | 4 | 400 | 100 | | |
| FirstFit | 4 | 800 | 100 | | |
| FirstFit | 4 | 1600 | 100 | | |
| CBIP | 2 | 50 | 100 | | |
| CBIP | 2 | 100 | 100 | | |
| CBIP | 2 | 200 | 100 | | |
| CBIP | 2 | 400 | 100 | | |
| CBIP | 2 | 800 | 100 | | |
| CBIP | 2 | 1600 | 100 | | |

Table 1: Example results table for each algorithm.

Perform an empirical study on the above parameters and metrics of the heuristic version of FirstFit algorithm on $k$-colourable online graphs for $k \in \{2, 3, 4\}$. Again, while performing theses studies, consider the choice of any parameters (e.g., $p$ for the online $k$-colourable graph generator—the behaviour of different heuristics may change according to the type of graph (dense or sparse)).

Fill in a table similar to Table 1 for your choice of heuristic. Again, give an analysis of your results.

## 2 Project Deliverables

1. A project report, submitted in PDF format. This report should be concise and clearly written and include the following sections:
   - Problem description
     In your words, what is the purpose of this project?
   - Implementation details

Give details of any of the main function implementations **including pseudocode**. Repeating back the instruction sheet will only get part marks.

- Implementation correctness
  How your tested the correctness of your implementations, both in terms of software correctness (e.g., boundary testing, etc.) but also that your graph algorithms are correct. For example, demonstrate the correctness of minimum-cost flow found, assuming $d = f_{\text{max}}$, on relatively small (not too small!) example graphs where the output can be verified independently.
- Results
  Complete the tables as requested and including explanation of any choices you made during your simulations and how they affected the results. For the choice of heuristic, less interesting solutions will result in lower marks.
- Analysis and Conclusions
  What overall (non-trivial) conclusions can you draw from the results of your simulations?
- Team work distribution
  In this part, clearly write down how the work was distributed to each member for evaluation purposes.
- References
  Any references used to implement the algorithms in your project. You must not "cut'n'paste" code from any of sources (including LLMs).

2. A zip file containing:
   - ASCII encodings of all your online $k$-colourable graphs as input to your simulation code.
   - Your implementation code including a README to explain how to compile and run.

## 3 Marking Details

It should be clear that the results of your simulations were derived from running your implementation code. The marker needs to be able to verify this. If this is not easily done, the following penalties may be applied:

- A deduction of at least 50% if your code does not run
- A deduction of at least 20% if it is very difficult to verify that the results of your simulations presented in your report were derived from running your code

## 4 Other Specifications

- Team size: either 3 or 4. You are responsible to form your own team, although the team from the group project is suggested. On the first page of your report, *clearly* list the members of your team along with their student numbers. Only one team member needs to submit their team's project submission.
- You may **not** use external libraries such as the Python libraries NetworkX and igraph for any part of your project. You must implement all aspects of your solution and can only use the built-in libraries that come with your chosen programming language.

## References

[1] Y. Li, V. Narayan, and D. Pankratov, “Online coloring and a new type of adversary for online graph problems,” *Algorithmica*, vol. 84, pp. 1232–1251, 2022.

[2] A. Gyárfás and J. Lehel, “On-line and first fit colorings of graphs,” *Journal of Graph Theory*, vol. 12, no. 2, pp. 217–227, 1988.

[3] L. Lovász, M. Saks, and W. Trotter, “An on-line graph coloring algorithm with sublinear performance ratio,” in *Graph Theory and combinatorics 1988* (B. Bollobás, ed.), vol. 43 of *Annals of Discrete Mathematics*, pp. 319–325, Elsevier, 1989.