COMP52315 Coursework
Advanced Algorithms

CIS Username: bgtt65

Problem 1. (a)

The clique number, chromatic number, acyclic chromatic number, and star colouring number for the graphs in Figures 1, 2 and 3 are shown in Table 1.

Name of Number	**Calculation result**				
No. of figure:		Figure 1		Figure 2	Figure 3
No. of graph:	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5
Clique number	2	3	3	3	3
Chromatic number	2	4	3	3	3
Acyclic chromatic number	2	4	4	4	5
Star chromatic number	3	4	4	5	5

Table 1: Calculation result of different numbers for the figures

Problem 1. (b)

In the GraphColouring folder, the main.py is a Python file which can create DIMACS SAT-instances files (.cnf) for the chromatic number and clique number problem, and those .cnf files are the proper results for GraphR1.txt to GraphR5.txt. There is a simple graph consisting of 4 vertices and 4 edge Fig. 1. The following two problems will take this graph for example to explain how the algorithm work.

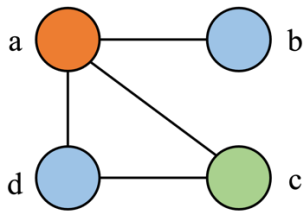

Figure 1: Example of a graph with colour

Chromatic number problem

In this problem, three rules are defined for searching the chromatic number. When we consider whether the example graph can be coloured with 3 colours (called 1, 2 and 3), it can be defined that if a is coloured with colour 1, then $a1$ is true.

- Rule 1: Each vertex must be coloured

Take vertex a for example, the clause is $a1 \vee a2 \vee a3$.

- Rule 2: Each vertex only be coloured by a single colour

Take vertex a for example, the clauses are $\neg a1 \vee \neg a2$, $\neg a1 \vee \neg a3$ and $\neg a2 \vee \neg a3$.

- Rule 3: Each edge cannot be coloured with the same colour

Take edge $\{a,b\}$ for example, the clauses are $\neg a1 \vee \neg b1$, $\neg a2 \vee \neg b2$ and $\neg a3 \vee \neg b3$.

Following these three rules, we can create a DIMACS SAT-instances file for our chosen number. The minimum number that its file can be solved will be the chromatic number.

Clique number problem

In this problem, three rules are defined for searching the clique number. Suppose we determine whether the example graph can have a clique with 3 vertices. We can define that a is true when vertex a is in a specific clique and is false otherwise.

- Rule 1: Two distinct vertices that are member of a clique must be connected by an edge

In rule 1, a constraint is enforced that if two vertices are true then an edge of them exists in the graph. In other words, if the edge does not exist in the graph, then two vertices cannot both be true.
Take vertices a and b for example, there is no clause for a, whereas the clauses for b are $\neg b \vee \neg c$ and $\neg b \vee \neg d$.

- Rule 2: Each vertex of the graph that is connected to all the vertices of a maximal clique must be also a member of the clique

We need to enforce a constraint that at least one of at least a vertex itself or one of those vertices that are not adjacent to this vertex should be true.
Take vertices a and b for example, a should be true the clause for b is $b \vee c \vee d$.

- Rule 3: Constraint the size of a maximal clique

A constraint is set in rule 3 that at least one of vertices in any component (with the size equal to the total vertices number – clique number we chosen + 1) should be true.
Take vertex a for example, the component size is 2, then the clauses are $a \vee b$, $a \vee c$ and $a \vee d$.

Following these three rules, we can create a DIMACS SAT-instances file for our chosen number. The maximum number that its file can be solved will be the clique number.

Calculation results

The clique number and chromatic number for the Graphs 1 to 5 are shown in Table 2.

Name of Number	**Calculation result**				
No. of graph:	Graph 1	Graph 2	Graph 3	Graph 4	Graph 5
Clique number	3	3	3	3	3
Chromatic number	4	4	3	3	3

Table 2: Calculation result of clique number and chromatic number for the graphs

Problem 1. (c)

If a graph admits an acyclic colouring with two colours, it should be an acyclic graph. An acyclic graph is a graph having no graph cycle, meanwhile, it is bipartite. The algorithm of GRAPH ACYCLIC 2-COLOURING problem is shown in Alg. 1.

Algorithm 1 GRAPH ACYCLIC 2-COLOURING problem

1: Assign colour 1 to the source vertex (putting into colour 1 set).
2: Colour all the neighbours with colour 2 (putting into colour 2 set).
3: Colour all neighbour's neighbours with colour 1 (putting into colour 1 set).
4: Repeat statement 2 and statement 3, assign a colour to all vertices.
5: Break if a neighbour's neighbour has been coloured (the graph is not acyclic)

If a graph is represented using an adjacency list, first, we traverse each vertex once. Then for each vertex, we visit each neighbour of a vertex once. Therefore, this algorithm takes $\mathcal{O}(V + E)$ time. As the time cost is in linear time and not complex, we can mention this algorithm is efficient.

Problem 1. (d)

If a graph admits an acyclic colouring with three colours, the chromatic number should be no more than three. Meanwhile, all the cycles cannot be induced by two colours. The algorithm of GRAPH ACYCLIC 3-COLOURING problem is shown in Alg. 2.

Algorithm 2 GRAPH ACYCLIC 3-COLOURING problem

Part 1: Colouring
1: Assign colour 1 to the source vertex (putting into colour 1 set).
2: Colour neighbours with appropriate colour 2 or 3 (putting into colour 2 or 3 sets).
3: Colour neighbour's neighbours with appropriate colour 1, 2 or 3 (putting into colour 1, 2 or 3 set).
4: Repeat statement 3, assign a colour to all vertices.
5: Break if a vertex cannot be coloured by colour 1 to 3 (the chromatic number is more than 3)
Part 2: Find Cycles
1: Mark the source vertex as visited.
2: Visit neighbours and mark them as visited (we mark when we visit any vertex).
3: Visit neighbour's neighbour, if their adjacent vertex is visited and not parent, then there is a cycle.
4: Repeat statement 3, search all the possible cycles in the graph.
Part 3: Check the colours in any cycle
1: Check how many colours have been used in a cycle (should be 2 or 3).
2: When the number of colours is two, check whether any vertex can be replaced by the third colour.
3: Break if any cycle cannot satisfy statement 2. (the acyclic chromatic number is more than 3)

We suppose that a graph is represented using an adjacency list. For part1 and part 2, we traverse each vertex once and visit neighbours of each vertex once. So, it takes $O(V + E)$ time for each part. In part 3, we can assume that there is a cycle set named C. For the worst case, all the vertices in cycles and their neighbours should be checked to reach the requirement, then it takes $O(C + E)$ time for each part. Therefore, combining the time of each part together, this algorithm takes $O(2V + C + 3E)$ time. The time cost is in linear time but a little bit complex, so this algorithm is good but not that efficient.

Problem 1. (e)

In question 1. (b), three rules were set to solve the chromatic number problem. In order to find the star chromatic number, we can search all the paths in the graph in the beginning. For example, the Depth-first search (DFS) and Breadth-first search (BFS) are algorithms for traversing or searching a graph that can achieve our purpose. After forming the path information of the graph, we can find all the possible combinations of the three-length path (with four vertices). Then, the rule 4 can be added to satisfy the requirement of star colouring:

- Rule 4: A three-length path cannot be induced by two colours

Following the assumption for the chromatic number problem in question 1. (b) that our chosen number is 3. In Fig. 1, take path $\{b,a,c,d\}$ for example, the clauses are $(\neg b1 \vee \neg a2 \vee \neg c1 \vee \neg d2)$, $(\neg b1 \vee \neg a3 \vee \neg c1 \vee \neg d3)$, $(\neg b2 \vee \neg a1 \vee \neg c2 \vee \neg d1)$, $(\neg b2 \vee \neg a3 \vee \neg c2 \vee \neg d3)$, $(\neg b3 \vee \neg a1 \vee \neg c3 \vee \neg d1)$, and $(\neg b3 \vee \neg a2 \vee \neg c2 \vee \neg d2)$.

Following those three rules in the chromatic number problem and rule 4 above, we can create a DIMACS SAT-instances file for our chosen number. The minimum number that its file can be solved will be the star chromatic number.

Problem 2. (a)

The definition of Markov's inequality:

$$P(\mathrm{X} \geq \alpha) \leq \frac{\mathbb{E}[\mathrm{X}]}{\alpha} \quad where\ X \geq 0\ and\ \alpha > 0$$

As the integer number $C = 65$ based on my CIS username,

$$P(\mathrm{X}_i = 1) = p = \frac{1}{C} = \frac{1}{65}$$

Consider independent random variables $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_{20}$. Let $\mathrm{X} = \sum_{i=1}^{20} \mathrm{X}_i$,

$$\mathbb{E}[\mathrm{X}] = 20 \cdot \frac{1}{65} = \frac{4}{13}$$

Use Markov's inequality to find a bound on the probability that X is at least 11,

$$P(\mathrm{X} \geq 11) \leq \frac{\mathbb{E}[\mathrm{X}]}{11} = \frac{4}{13} \cdot \frac{1}{11} = \frac{4}{143} \approx 0.03$$

Problem 2. (b)

The definition of Chebyshev's inequality:

$$P(|\mathrm{X} - \mathbb{E}[\mathrm{X}]| \geq \alpha) \leq \frac{1}{\alpha^2} \cdot \mathrm{Var}[\mathrm{X}]$$

In problem 2.(a), p and $\mathbb{E}[\mathrm{X}]$ are solved. As X is binomially distributed and X_i are independent,

$$\mathrm{Var}[\mathrm{X}] = \sum_{i=1}^{20} \mathrm{Var}[\mathrm{X}_i] = 20 \cdot p \cdot (1-p) = 20 \cdot \frac{1}{65} \cdot \frac{64}{65} = \frac{256}{845}$$

Use Chebyshev's inequality to find a bound on the probability that X is at least 11,

$$P(\mathrm{X} \geq 11) = P(|\mathrm{X} - \frac{4}{13}| \geq 11 - \frac{4}{13}) = P(|\mathrm{X} - \frac{4}{13}| \geq \frac{139}{13})$$

$$\leq \left(\frac{13}{139}\right)^2 \cdot \mathrm{Var}[\mathrm{X}] = \frac{169}{19321} \times \frac{256}{845} = \frac{256}{96605} \approx 0.003$$

Problem 2. (c)

The definition of Generic Chernoff bound:

$$P(\mathrm{X} \geq (1+\delta)\mathbb{E}[\mathrm{X}]) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mathbb{E}[\mathrm{X}]}$$

In problem 2.(a), $\mathbb{E}[\mathrm{X}]$ is solved. Calculate δ for this problem,

$$(1+\delta)\mathbb{E}[\mathrm{X}] = (1+\delta) \cdot \frac{4}{13} = 11$$

$$\delta = 11 \cdot \frac{13}{4} - 1 = \frac{139}{4}$$

Use Generic Chernoff bound to find a bound on the probability that X is at least 11,

$$P(\mathrm{X} \geq 11) \leq \left(\frac{e^{139/4}}{(143/4)^{143/4}}\right)^{4/13} \approx 4 \cdot 10^{-13}$$

Problem 3.

Define the period of state j to be the greatest common divisor:

$$d_j = \gcd\left(\{k > 0; p_{jj}^{(k)} > 0\}\right)$$

As two states i and j are reachable from each other and there exists m and n such that $p_{ij}^{(m)} > 0$ and $p_{ji}^{(n)} > 0$. Then, we note that $p_{ii}^{(m+n)} > 0$, the period of state i can be shown as:

$$d_i \mid m + n$$

The result means that $m + n$ is an integer multiple of the period d_i. Since we suppose that $p_{jj}^{(k)} > 0$, then we know that $p_{ii}^{(m+k+n)} > 0$. So that:

$$d_i \mid m + k + n$$

Subtracting the last two displays gives:

$$d_i \mid k$$

Since we have defined that k is an arbitrary integer which satisfy $p_{jj}^{(k)} > 0$, we can find that d_i is a common divisor of the set $\{k > 0; p_{jj}^{(k)} > 0\}$. Remember that d_j is the greatest common divisor of this set, then we can find that:

$$d_j \geq d_i$$

Interchanging the role of two states i and j in the previous argument gives:

$$d_i \geq d_j$$

Combine these two opposite inequalities, it can be proved that two states that are reachable from each other must have the same period:

$$d_i = d_j$$