Graph Algorithms — Home assignment

1 A consequence of Brooks' Theorem for triangle-free graphs

The goal of this exercice is to prove the following theorem.

Theorem 1 *Let G be a graph of clique number $\omega(G) \leq 3$ and of maximum degree Δ. Then*

$$\chi(G) \leq 3 \left\lceil \frac{\Delta+1}{4} \right\rceil .$$

Let G be such a graph. Set $k := \lceil \frac{\Delta+1}{4} \rceil$. Let $(V_1, \ldots, V_k)$ be a partition of $V(G)$ that minimises the number of internal edges (i.e. the number of edges uv such that $u, v \in V_i$ for some $1 \leq i \leq k$).

1. *Show that $\Delta(G[V_i]) \leq 3$, for every $1 \leq i \leq k$.*

   Assume for the sake of contradiction that some vertex $v \in V_i$ has degree at least 4 in $G[V_i]$. Let d_j be the degree of v in $G[V_j]$, for every $j \in [k]$. Then $\sum_{j \in [k]} d_j = \deg_G(v) \leq \Delta$. By the pigeonhole principle, there exists j such that $d_j \leq \Delta/k < 4$. Then, by moving v from V_i to V_j, we obtain a new partition with $d_i - d_j > 0$ fewer internal edges, a contradiction.

2. *Show that $\chi(G) \leq 3k$, with the help of Brooks' theorem.*

   Every graph $G[V_i]$ has maximum degree at most 3 and clique number at most 3, so in particular none of its connected component is a copy of K_4. We conclude by Brooks' Theorem that they are all 3-colourable. We introduce three distinct colours to properly colour each of them. Since the colours between $G[V_i]$ and $G[V_j]$ are distinct when $i \neq j$, there is no conflict on the external edges by taking the union of those colourings; this yields a proper $3k$-colouring of G.

3. *Using the above, write an algorithm that computes a proper $3k$-colouring of G. What is its complexity?*

   We use the algorithm `Brooks` defined in [TD3, Q1.2]. This algorithm takes a connected graph G as an input, and returns a $\Delta(G)$-colouring if G is not a complete graph nor an odd cycle, in time $O(|E(G)|)$. We extend it so that it can take as input a non-connected graph of maximum degree at most 3 by applying `Brooks` on each connected component of maximum degree 3, and the Greedy Colouring Algorithm on each connected component of maximum degree ≤ 2. This returns a proper 3-colouring in time $O(m)$, where m is the number of edges.

Algorithm 1: Colouring

Data: G: graph with m edges
$V_1 \leftarrow V(G)$
$V_2, \ldots, V_k \leftarrow \varnothing$
while $\exists i \in [k], \exists v \in V_i, \deg_{G[V_i]}(v) > 3$ **do**
  | Move v to the part V_j that contains the least number of neighbours of v.
end
for $i \in [k]$ **do**
  | $c_i \leftarrow$ `Brooks`$(G[V_i])$
end
return $\bigcup_{i \in [k]} c_i$

The `While` loop is repeated at most m times, since the number of internal edges decreases by at least 1 at each iteration. Indeed, the same argument as that of Question 1 ensures that the degree of v in V_j is at most 3.

Using a bucket queue similar to the one described in [TD2, Exercise 1.1], we can do the operations in each iteration of the `While` loop in time $O(\Delta)$. So the total complexity of the `While` loop is $O(m\Delta)$. The complexity of each iteration of the `For` loop is $O(|E(G[V_i])|) = O(|V_i|)$, so the total complexity of the `For` loop is $O(|V(G)|)$. We conclude that the total complexity of the algorithm is $O(m\Delta) = O(n\Delta^2)$, where $n = |V(G)|$.

2 List colouring

Given a graph G, a *list assignment of* G is a function $L\colon V(G) \to 2^{\mathbb{N}}$. If $|L(v)| = k$ for all $v \in V(G)$, we say that L is a k*-list assignment* of G. The elements in $\bigcup_{v \in V(G)} L(v)$ are the *colours* of L, and $L(v)$ is the *list of colours* allowed for each vertex $v \in V(G)$. A proper L-colouring of G is a proper colouring c of G such that $c(v) \in L(v)$ for every vertex $v \in V(G)$ (every vertex gets a colour from its list). In particular, a proper k-colouring of G is a proper L-colouring with $L(v) = \{1, \dots, k\}$ for all $v \in V(G)$.

The minimum k such that G is L-colourable for every k-list assignment L of G is the *list-chromatic number* of G, denoted $\chi_\ell(G)$. The goal of this exercise is to study some properties of list colourings.

1. *Given a cycle C, and a 2-list-assignment L of C, show that C is not L-colourable if and only if C is odd and all lists are the same.*

   We first note that for every 2-list-assignment L of a path $P = x_1, \dots, x_n$, P is greedily L-colourable by colouring the vertices in the order of the path, since at each step i there is at most one colour from the list $L(v_i)$ which is forbidden for v_i.

   Let C be a cycle, and L a 2-list assignment where not all the lists are the same. In particular, there exist two consecutive vertices u, v on C such that $L(u) \neq L(v)$. Let $x \in L(u) \setminus L(v)$, and set $c(u) := x$. Then, colour greedily the path $C \setminus uv$ in the order of the path, by ending in v. The colour given for v is necessarily different from x, hence this returns a proper L-colouring of C.

   If now L is a 2-list assignment where all lists are the same, then C is L-colourable if and only if C is 2-colourable, so if and only if C is an even cycle.

2. *Let $n = \binom{2k-1}{k}$ for some $k \geq 1$, and let $G = (U, V, E)$ be the complete bipartite graph $K_{n,n}$ (i.e. $E = \{uv : u \in U, v \in V\}$). Let L be a list assignment of G such that, for every subset X of $\{1, \dots, 2k-1\}$ of cardinality k, there exists $u \in U$ and $v \in V$ such that $L(u) = L(v) = X$. Show that G is not L-colourable.*

   Assume for the sake of contradiction that there exists a proper L-colouring c of G. We first show that at least k different colours appear in U. Indeed, assume otherwise that the set of colours $c(U)$ has size at most $k-1$, then there exist k different colours that do not appear in U, and by construction there is a vertex $u \in U$ whose list consist exactly of these k colours, a contradiction. By symmetry, the same holds for V. Since there are only $2k-1$ different colours, it means that some colour must appear both in U and in V. This yields a conflict since all edges are present between U and V, a contradiction.

3. *Prove that* $(2,3)$-LIST-COLOUR *is NP-complete on the class of bipartite graphs. The reduction is from* 3-SAT.

   The problem $(2,3)$-LIST-COLOUR is a decision problem, and if the answer is positive then a proper L-colouring of G is a certificate that can be checked in time $O(m)$ by enumerating all the edges and checking that they induce no conflict for c. So the problem is in NP.

   Let $X = C_1 \wedge \cdots \wedge C_n$ be an instance of 3-SAT, and let $x_1, \dots, x_m$ the boolean variables of that instance. We construct a bipartite graph $G_X = (U, V, E)$, where each vertex in U represents one of the clauses, each vertex in V represents one of the boolean variables, and there is an edge $uv \in U \times V$ whenever the variable represented by v appears in the clause represented by u. For every vertex $u \in U$ representing a clause C, we let $L(u)$ be the set of literals of the clause C, and for every vertex $v \in V$ representing the boolean variable x_i, we let $L(v) := [x_i, \overline{x_i}]$ (using an implicit bijection between the set of literals and $[2m]$).

If the graph G_X has a proper L-colouring c, we define the truth assignment ϕ by $\phi(x_i) \coloneqq \texttt{True}$ if $c(v) = \overline{x_i}$ for the vertex v representing the variable x_i, and $\phi(x_i) \coloneqq \texttt{False}$ otherwise. For every vertex u representing a clause C, the colour $c(u)$ represents a literal t that is true by ϕ, since the vertex v representing the corresponding boolean variable must be coloured with $c(v) = \bar{t}$. We conclude that every clause C is satisfied by ϕ, hence X is satisfiable.

Conversely, assume that there exists a truth assignment ϕ that satisfies X. For every vertex $v \in V$ representing the variable x_i, let $c(v) \coloneqq \overline{x_i}$ if $\phi(x_i) = \texttt{True}$, and $c(v) \coloneqq x_i$ otherwise. For every vertex u representing a clause C, there exists a literal t of C that is true, and we let $c(u) \coloneqq t$. Then it is straightforward that c is a proper L-colouring of G_X.

We conclude that indeed 3-SAT reduces polynomially to (2,3)-LIST-COLOUR, hence (2,3)-LIST-COLOUR is NP-complete.

4. *Prove that* 2-LIST-COLOUR *is in* P.

   Let G be a graph, and $L\colon V(G) \to \binom{\mathbb{N}}{2}$ a 2-list assignment of G. Let us construct an instance X of 2-SAT that is satisfiable iff G is L-colourable.

   To that end, we construct for every vertex $v \in V(G)$ and every colour $c \in L(v)$ the boolean variable $x_{v,c}$, that will be true if v is coloured with c and false otherwise. To force each vertex v to be coloured with a vertex from its list $L(v) = \{c_1, c_2\}$, we add to X the clause $x_{v,c_1} \vee x_{v,c_2}$. To forbid a conflict on each edge $uv \in E(G)$, we add to X the clauses $\overline{x_{u,c}} \vee \overline{x_{v,c}}$ for every $c \in L(u) \cap L(v)$.

   Then G is L-colourable iff X is satisfiable. Indeed, if there is a proper L-colouring ϕ of G, then setting $x_{v,c}$ to true whenever $\phi(v) = c$ and to false otherwise yields a valuation that satisfies X. Conversely, if X is satisfiable, then each clause $x_{v,c_1} \vee x_{v,c_2}$ is satisfied, hence x_{v,c_i} must be true for some $i \in \{1, 2\}$. We then set $\phi(v) \coloneqq c_i$, and we claim that this is a proper L-colouring of G. For each vertex $v \in V(G)$, the fact that $\phi(v) \in L(v)$ follows from the fact that $L(v) = \{c_1, c_2\}$, so ϕ is indeed an L-colouring. The fact that ϕ is proper is ensured by the clauses of the form $\overline{x_{u,c}} \vee \overline{x_{v,c}}$ that prevent two adjacent vertex to share the same colour.

   It is clear that $|X| = O(|G|)$, so we have just described a polynomial reduction from 2-LIST-COLOUR to 2-SAT. Since 2-SAT is in P, we infer that 2-LIST-COLOUR is also in P, as desired.

5. *Prove that* $\chi_\ell(G) \leq \delta^*(G) + 1$ *for every graph* G.

   We extend the Greedy Colouring Algorithm so that it can be applied to L-colourings.

Algorithm 2: Greedy Colouring Algorithm

Data: G: d-degenerate graph with reverse degeneracy ordering $V(G) = \{v_1, \dots, v_n\}$
  L: $(d+1)$-list assignment of G
Result: c: a proper L-colouring of G
for i *from* 1 *to* n **do**
  | $c(v_i) \leftarrow \min L(v_i) \setminus c(N(v_i))$
end
return c

Let us prove that Algorithm 2 is correct. Let $N^-(v_i) \coloneqq \{v_j \in N(v_i) : j < i\}$ be the set of neighbours of v_i with a lower index, for all $i \in [n]$. By the property of a reverse degeneracy ordering of G, we have $|N^-(v_i)| \leq d$ for every $i \in [n]$. We note that at step i of the for loop in Algorithm 2, only the vertices with index lower than i are coloured. Hence, at step i, $|c(N(v_i))| \leq |N^-(v_i)| \leq d$. Since $|L(v_i)| = d + 1$, we conclude that $L(v_i) \setminus c(N(v_i))$ is non-empty, and so that $c(v_i)$ is well-defined.

We have proved that Algorithm 2 correctly constructs a proper L-colouring of G under the assumption that G is d-degenerate and L is any $(d+1)$-list assignment of G. This proves that $\chi_\ell(G) \leq \delta^*(G) + 1$. We note moreover that one can implement Algorithm 2 with a time complexity of the form $O(\sum_i \deg(v_i)) = O(|E(G)|)$.

3 A polytime algorithm for solving 3-COLOUR in dense graphs

In this exercise, G is a graph on n vertices. We say that G is *dense* if $\delta(G) \geq n/2$.

1. A *dominating set* of G is a set of vertices $D \subseteq V(G)$ such that $N_G[D] = V(G)$ (every vertex $v \in V(G) \setminus D$ has a neighbour in D). We denote $\gamma(G)$ the minimum size of a dominating set of G. We will show that the following greedy algorithm returns a dominating set of G of size at most $\log_2 n + 1$ when G is dense.

Algorithm 3: Greedy Dominating Set

```
Data: G: graph
Result: D: dominating set of G
V_0 ← V(G), i ← 0
while V_i ≠ ∅ do
    v_i ← vertex of V(G) with maximum degree in V_i
    V_{i+1} ← V_i \ N[v_i]
    i ← i + 1
end
return {v_0, ..., v_i}
```

(a) *Let $H = (X, Y, E)$ be a bipartite graph. Show that $|X| \operatorname{ad}(X) = |Y| \operatorname{ad}(Y)$.*
We have

$$|X| \operatorname{ad}(X) = \sum_{x \in X} \deg(x) = \sum_{x \in X} \sum_{e \in E(H)} \mathbb{1}_{x \in e} = \sum_{e \in E(H)} \sum_{x \in X} \mathbb{1}_{x \in e} = \sum_{e \in E(H)} 1 = |E(H)|.$$

Likewise,

$$|Y| \operatorname{ad}(Y) = |E(H)|.$$

The conclusion follows.

(b) *Assume that G is dense. Show that, at each iteration of the loop, the degree of v_i in V_i is at least $|V_i|/2$.*

If there is a vertex $u \in V_i$ of degree at least $|V_i|/2$ in V_i, we are done, so we assume that this is not the case.
Let H be the bipartite subgraph of G induced by the cut $(V_i, \overline{V_i})$. For every $u \in V_i$ we have

$$\deg_H(u) = \deg_G(u) - \deg_{V_i}(u) \geq \delta(G) - \frac{|V_i|}{2} \geq \frac{n}{2} - \frac{|V_i|}{2}.$$

By (1a), we have

$$\operatorname{ad}_H(\overline{V_i}) = \frac{|V_i|}{|\overline{V_i}|} \operatorname{ad}_H(V_i) \geq \frac{|V_i|}{n - |V_i|} \left(\frac{n}{2} - \frac{|V_i|}{2} \right) = \frac{|V_i|}{2}.$$

In particular, there exists a vertex $u \in \overline{V_i}$ of degree at least $|V_i|/2$ in H, as desired.

(c) *Show that the algorithm returns a dominating set of G of size at most $\log_2 n + 1$ when G is dense.*
Let $D := \{v_0, \dots, v_i\}$ be the returned set. By construction, at each step of the loop, $V_i = V(G) \setminus \bigcup_{j=0}^{i} N[v_i]$. So when the loop terminates, since we have $V_i = \emptyset$, this means that $N[D] = V(G)$, so D is a dominating set of G.
By (1b), we have $|V_{i+1}| \leq |V_i|/2$ for each i. So after $\lfloor \log_2 n \rfloor$ iterations of the loop, we have $|V_i| \leq 1$. Hence after $\lfloor \log_2 n \rfloor + 1$ iterations, the loop terminates. This proves that $|D| \leq \log_2 n + 1$.

2. *Let D be a dominating set of G, and $\phi\colon D \to [3]$ a proper 3-coloring of $G[D]$. Show that it is possible to test in polynomial time whether ϕ extends to a proper 3-colouring c of G (we must have $c(x) = \phi(u)$ for every $u \in D$).*

   Hint: Show that this reduces to solving an instance of 2-LIST-COLOUR.

   Let L be a list assignment of $V(G) \setminus D$ defined by

   $$L(v) \coloneqq \{1, 2, 3\} \setminus \phi(N_D(v)),$$

   for every $v \in V(G) \setminus D$. Since D is a dominating set, every v has a non-empty neighbourhood $N_D(v)$ in D, hence $|L(v)| \leq 2$ for every v.

   - If $L(v) = \emptyset$ for some v, then we can conclude that ϕ does not extend to a proper 3-colouring of G.
   - If $|L(v)| = 2$ for every v, then $(G \setminus D, L)$ is an instance of 2-LIST-COLOUR which can be solved in polynomial time, and which is equivalent to the problem of extending ϕ to G (easy proof omitted).
   - Otherwise, we add to D each vertex v with $L(v) = \{x\}$ for some colour x, and set $\phi(v) \coloneqq x$. We update the list-assignment L as before, and repeat the above arguments. This process must terminate, since each of its iterations increases the size of D, while we always have $D \subseteq V(G)$.

3. *Using the above, describe a polytime algorithm to solve* 3-COLOUR *on G when G is dense.*

   The algorithm can be described as follows.

Algorithm 4: 3-COLOUR

```
Data: G: dense graph on n vertices
Result: χ(G) ≤ 3?
D ← GreedyDominatingSet(G)
foreach φ: (possibly improper) 3-colouring of G[D] do
    // We first check whether φ is proper
    foreach uv ∈ E(G[D]) do
        if φ(u) = φ(v) then
            Continue to next value of φ
        end
    end
    // Here we now that φ is proper, and we use result from previous question.
    if φ extends to a proper 3-colouring of G then
        return True
    end
end
return False
```

The above algorithm runs in polynomial time. It begins by constructing greedily a dominating set D of G of size at most $1+\log_2 n$, as seen in Question 1. This can be done in time $O(|G|)$ with a suitable implementation.

It then iterates over every 3-colouring of D; there are $3^{|D|} \leq 3^{1+\log_2 n} = O(n^{\log_2 3})$ of them. Each iteration has polynomial cost, so the overall complexity is polynomial.

The algorithm is correct, since if G has a proper 3-colouring c, then $\phi \coloneqq c_{|D}$ is a proper 3-colouring of $G[D]$ that extends to c. The algorithm will therefore return `True` when it iterates over that specific value of ϕ. Otherwise, the algorithm returns `False`, as desired.

4. *What can you say when $\delta(G) \geq c \cdot n$ for some absolute constant $c > 0$?*

   The same strategy can be used to solve 3-COLOUR in polytime. The only effect of changing c from $1/2$ to any other positive value lies in the computations of Question 1: the degree of v_i in V_i is now at least $c\,|V_i|$, and so the dominating set produced has size at most $1 + \log_{1/(1-c)} n$. This size is still logarithmic, so the complexity of Algorithm 4 remains polynomial.