Graph Algorithms

TD3 : Complexity

Throughout this TD, given a graph G, n is its number of vertices, and m its number of edges.

1 Algorithmic complexity

1. *Give an explicit implementation of a bucket queue so that one can compute a degeneracy ordering of a given graph G in time $O(m)$.*

   We need to represent the lists `D[i]` ith a structure that lets us insert and remove a given element in constant time. To do so, we are going to emulate doubly linked lists for `D[i]`. For each element in `D[i]`, we need to now its successor and its predecessor in `D[i]`. We register these values in two vectors `succ` and `pred`, of size $n = |V(G)|$, initialised to `Void` values, and in `D[i]` we register the value of the first element.

   Algorithm 1: `Insert(`x`,`d_x`)`

   ```
   if D[d_x] = Void then
   |  D[d_x] ← x
   else
   |  y ← D[d_x]
   |  pred[y] ← x
   |  succ[x] ← y
   |  D[d_x] ← x
   end
   ```

   Algorithm 2: `Remove(`x`,`d_x`)`

   ```
   z ← succ[x]
   if pred[x] ≠ Void then
   |  y ← pred[x]
   |  succ[y] ← z
   |  pred[z] ← y
   else
   |  D[d_x] ← z
   end
   pred[x] ← Void
   succ[x] ← Void
   ```

2. *Write an algorithm in pseudo-code that computes a $\Delta(G)$-colouring of a connected graph G that is neither complete nor an odd cycle. It should have complexity $O(m)$.*

   Let G be a connected graph of maximum degree Δ, that is neither complete nor an odd cycle. Let us compute a rooted block-decomposition T of G in time $O(|E(G)|)$.

   - First assume that T contains at least 2 blocks. For every block B of G, the maximum degree of $G[B]$ is at most $\Delta - 1$, so one can colour $G[B]$ greedily in any order with Δ colours. We simply colour the blocks of G in a DFS ordering with respect to T, where the first coloured vertex in each block is the articulation point that is shared with the parent block.

- Assume now otherwise that G is 2-connected. In order to find an induced P_3 with extremities a, b such that $G \setminus \{a, b\}$ is connected, we need to find a vertex x that is not universal, and compute a block decomposition of $G \setminus x$. This can be done in time $O(|E(G)|)$. The rest of the operations can be done in time $O(|E(G)|)$.

2 NP-completeness

In this exercise, we study the NP-completeness of the INDEPENDENTSET problem.

1. **Reduction from SAT**. *Let $C_1, \ldots, C_r$ be the clauses in an instance X of* SAT. *Construct a clique of size $|C_i|$ for every clause C_i, and label its vertices with the literals that appear in C_i. Add an edge between every pair of vertices labelled with opposite literals x and $\overline{x}$. This returns a graph G_X. Show that the instance X is satisfiable if and only if $\alpha(G_X) \geq r$.*

   Let us assume that X is satisfiable, and let ϕ be a truth assignment of its boolean variables that certifies the satisfiability of X. For every clause C_i, let x_i be its first literal such that $\phi(x) = \texttt{True}$. We construct a set I which consists of the vertex labelled x_i from the clique associated to the clause C_i. By construction, if there is an edge between two vertices in I, it lies between two cliques associated to different clauses, and so it links vertices labelled with opposite literals $x, \overline{x}$. But if a vertex $v \in I$ is labelled x, then $\phi(x) = \texttt{True}$, so that contradicts the presence of an edge in I. We conclude that I is an independent set, of size r, and so that $\alpha(G_X) \geq r$.

   Conversely, let us assume that $\alpha(G_X) \geq r$, and let I be an independent set of size r in G_X. For every $v \in I$ labelled with the literal x, we set $\phi(x) \coloneqq \texttt{True}$. Since I is an independent set, we never set a $\texttt{True}$ value to two opposite literals, so this yields a partial truth assignment of the boolean variables of X. Each clause is satisfied with this partial truth assignment, hence X is satisfiable. This concludes the proof.

2. **Reduction from COLOR**. *Let G be a graph. Let $k \cdot G$ be the graph obtained by replacing each vertex $v \in V(G)$ in G with a clique $W(v)$ of size k (denote its vertices $v_1, \ldots, v_k$), and each edge $uv \in E(G)$ with the complete matching $u_1v_1, \ldots, u_kv_k$. Show that $\chi(G) \leq k$ if and only if $\alpha(k \cdot G) \geq |V(G)|$.*

   Assume that $\chi(G) \leq k$, and let c be a proper k-colouring of G. Let $I \leftarrow \varnothing$, and for every vertex $v \in V(G)$ with $c(v) = i$, we add the vertex $v_i \in V(k \cdot G)$ to I. I is a subset of $V(k \cdot G)$ of size $|V(G)|$, and we argue that I is independent. Indeed, if there exists and edge $u_iv_j \in G[I]$, then $uv \in E(G)$, and moreover $c(u) = i = j = c(v)$; this contradicts the fact that c is proper. So $\alpha(k \cdot G) \geq |V(G)|$.

   Conversely, assume that $\alpha(k \cdot G) \geq |V(G)|$, and let I be an independent set of $k \cdot G$ of size $|V(G)|$. For each $v \in V(G)$, since $W(v)$ is a clique, it contains at most one vertex of I. Since moreover $V(k \cdot G) = \bigcup_{v \in V(G)} V(W(v))$, then each clique $W(v)$ contains exactly one vertex v_i in I, and we define $c(v) \coloneqq i$. We argue that c is a proper k-colouring of G. Indeed, assume for the sake of contradiction that $c(u) = c(v) = i$ for some edge $uv \in E(G)$. Then $G[I]$ contains the edge u_iv_i, a contradiction. So $\chi(G) \leq k$.

3 VERTEXCOVER is FPT

The problem VERTEXCOVER consists in deciding if a graph G contains a vertex cover, that is a set of vertices X such that each edge $e \in E(G)$ has an extremity in X, of size at most k.

1. Prove that the algorithm is correct.

Algorithm 3: VertexCover

Data: G: graph, k: integer
Result: decides whether G has a vertex cover of size $\leq k$

```
if E(G) = ∅ then
  | return True
end
if k = 0 then
  | return False
end
uv ← an edge from E(G)
G1 ← G \ u
G2 ← G \ v
return VertexCover(G1, k-1) ∨ VertexCover(G2, k-1)
```

2. Compute its complexity. In what parameter is VERTEXCOVER FPT?