Sparsity — tutorial 6

Tree-depth, tree-width and low tree-depth colorings

Problem 1. Prove that a class $\mathcal{C}$ of graphs has bounded tree-depth if and only if there is a number k such that no graph $G \in \mathcal{C}$ contains a path of length k.

Proof. First observe that if $H \subseteq G$, then $\mathsf{td}(H) \leq \mathsf{td}(G)$ (we can use the same decomposition for H as for G).

We now prove that The tree-depth of an n-vertex path is equal to $\lceil \log_2(n+1) \rceil$.

It suffices to prove that whenever $2^k \leq n \leq 2^{k+1} - 1$, the tree-depth of P_n is equal to $k+1$. We proceed by induction on k. For $k = 0$ the claim holds trivially. Let P_n be the path on n vertices, where $2^k \leq n \leq 2^{k+1} - 1$.

We first show that $\mathsf{td}(P_n) \leq k+1$. Let u be a middle vertex on P_n, that is, one whose removal splits P_n into two subpaths on at most $\lfloor n/2 \rfloor < 2^k$ vertices each. By induction assumption, for each of these subpaths we can find a tree-depth decomposition of height at most k, and these can be combined into a tree-depth decomposition of P_n of height $k+1$ by taking their union and adding u as the root.

We now show that $\mathsf{td}(P_n) \geq k+1$. Take any tree-depth decomposition T of P_n; T is a rooted tree since P_n is connected. Let u be the root of T. Then each of the two subpaths of $P_n - u$ is placed entirely in one subtree rooted at a child of u in T. Since $n \geq 2^k$, one of these subpaths has at least 2^{k-1} vertices, so by the induction assumption its tree-depth is at least k. Then the corresponding subtree of T rooted at a child of u has height at least k, implying that T has height at least $k+1$.

Now, if $\mathcal{C}$ contains graphs with arbitrary long paths as subgraphs, the class of all paths $\mathcal{P}$ cannot have larger tree-depth. However, as we showed, the tree-depth of this class is unbounded.

On the other hand, if $\mathcal{C}$ has a bound on the path lengths, then we will find a bound on tree-depth, as the next exercise will show. □

Problem 2. Let G be an n-vertex graph with m edges. Show that we can compute in time $\mathcal{O}(n+m)$ a tree-depth decomposition of depth at most $2^{\mathsf{td}(G)}$.

Proof. We show that every rooted forest F obtained by running a depth-first search in each connected component of G satisfies $G \subseteq \mathrm{clos}(F)$ and $\mathrm{height}(F) < 2^{\mathsf{td}(G)}$.

Let F be such a rooted forest. It is straightforward to see that $G \subseteq \mathrm{clos}(F)$. Indeed, if there was an edge $uv \in E(G)$ such that u and v were not bound by the ancestor-descendant relation in F, then provided u was visited earlier by the DFS than v, the edge uv would be used by the DFS to access v from u, so v should have been a descendant of u. To see that $\mathrm{height}(F) \leq 2^{\mathsf{td}(G)}$, observe that if $d := \mathrm{height}(F)$, then G contains a path on d vertices. By what we showed in exercise 1 we infer that the tree-depth of this path, and consequently also the tree-depth of G, is at least $\lceil \log_2(d+1) \rceil$. Since $\mathsf{td}(G) \geq \lceil \log_2(d+1) \rceil > \log_2 d$, it follows that $d < 2^{\mathsf{td}(G)}$. □

Definition 1. A *tree-decomposition* of a graph G is a pair $\mathcal{T} := (T, (B_t)_{t \in V(T)})$ consisting of a tree T and a family $(B_t)_{t \in V(T)}$ of sets $B_t \subseteq V(G)$, such that

1. for all $v \in V(G)$ the set
$$B^{-1}(v) := \{t \in V(T) \ : \ v \in B_t\}$$
is non-empty and induces a subtree of T and
2. for every edge $e \in E(G)$ there is $t \in V(T)$ with $e \subseteq B_t$.

The *width* of $\mathcal{T}$ is $\max\{|B_t| - 1 \ : \ t \in V(T)\}$. The *tree-width* of G, denoted $\mathsf{tw}(G)$, is defined as the minimal width of any tree-decomposition of G.

Problem 3. Let G be an n-vertex graph. Prove that $\mathrm{col}_n(G) = \mathsf{tw}(G) + 1$.

Proof. First, fix an order σ of $V(G)$, say the vertices are enumerated as $v_1, \ldots, v_n$, such that $\mathrm{col}_n(G, \sigma) = \mathrm{col}_n(G)$. For notational simplicity, let G' be the graph obtained by adding a vertex v_0 to G which is adjacent to all vertices of G. We define a tree-decomposition of G' as follows (the decomposition of G is obtained by removing v_0 from every bag).

We define a tree T with vertex set $x_0, x_1, \ldots, x_n$. For $0 \leq i \leq n$, we denote the bag B_{x_i} by B_i. The root of T is x_0 with $B_0 = \{v_0\}$.

For $i \geq 1$, denote by G_i the component of $G' - (v_0, \ldots, v_{i-1})$ which contains the vertex v_i. By induction, we continue build the tree T as follows. If i is minimal such that G_i is a component of $G' - (v_0, \ldots, v_{i-1})$ (and not already a component of $G' - (v_0, \ldots, v_{i-2})$ or $G' - v_0$ if $i = 1$), we attach x_i as a child of x_{i-1}. This clearly defines a tree T.

For $i \geq 1$, we define $B_i \coloneqq \mathrm{SReach}_n[G, \sigma, v_i]$.

Let us show that this decomposition satisfies the properties of a tree-decomposition. Its width is as desired be definition of $\mathrm{SReach}_n[G, \sigma, v_i]$.

The second property is easy to verify: every edge $v_i v_j$ can be found in the bag B_j.

It remains to verify that for all i, the set $B^{-1}(v_i)$ induces a subtree of T, that is, it is connected in T (It is non-empty as $v_i \in \mathrm{SReach}_n[G, \sigma, v_i] = B_i$). By definition, v_i is contained in exactly those B_j with $v_i \in \mathrm{SReach}_n[G, \sigma, v_j]$. For $x, y \in V(T)$, denote by $x \wedge y$ the least common ancestor of x and y in T.

Assume towards a contradiction that $B^{-1}(v_i)$ is not connected in T. We distinguish two cases. First case, there are j_1, j_2, j_3 such that $x_{j_1} <_T x_{j_2} <_T x_{j_3}$ in the partial tree order $<_T$ and $v_i \in B_{j_1}, B_{j_3}$ and $v_i \notin B_{j_2}$. We claim that this is not possible. v_{j_3} is in the component of $G' - (v_0, \ldots, v_{j_2})$ and there is a path P between v_{j_3} and v_1 which uses internal vertices larger than v_{j_3}, which hence are also in the same component of $G' - (v_0, \ldots, v_{j_2})$ as v_{j_3}. As this component was created at the time when x_{j_2} was put into the tree, there is a path Q between v_{j_2} and v_{j_3} such that all internal vertices are larger than v_{j_2}. Then the walk which is the concatenation of Q and P contains a path which has all internal vertices larger than v_{j_2} and which ends at v_{j_1}. Hence, also v_1 in B_{j_2}, contradicting our assumption.

Second case, there are j_1, j_2 which are incomparable in the partial tree order with $v_i \in B_{j_1}, B_{j_2}$ and $v_i \notin B_j$ for $j = j_1 \wedge j_2$. This case reduces to the first case by observing we must have $x_i <_T x_j <_T x_{j_1}$.

This contradiction shows that $B^{-1}(v_i)$ is connected in T, which finishes the proof.

Now conversely assume that we are given a tree-decomposition with bags of size at most k. We first prove that every bag of the decomposition (in fact, the intersection of any two neighboring bags) is a separator in G. Formally, let $X, Y \subseteq V(G)$. A set $S \subseteq V(G)$ *separates* X and Y, if every path from a vertex in X to a vertex in Y contains a vertex from S. We prove that if $e = xy \in E(T)$, then $B_x \cap B_y$ is a separator of G, more precisely, if T_1, T_2 are the components of $T - e$, then $B_s \cap B_t$ separates the sets $B(T_1) = \bigcup_{x' \in V(T_1)} B(x')$ and $B(T_2) = \bigcup_{y' \in V(T_2)} B(y')$.

As $B^{-1}(w)$ is connected in T for each $w \in V(G)$, if $w \in B(T_1)$ and $w \in B(T_2)$, then $w \in B_x \cap B_y$. Now, let $P = v_1, \ldots, v_\ell$ be a path in $G - (B_x \cap B_y)$, without loss of generality $v_1 \in B(T_1)$. Assume towards a contradiction that there is i, $1 \leq i < \ell$, such that $v_i \in B(T_1)$ and $v_{i+1} \in B(T_2)$. As $\{v_i, v_{i+1}\} \in E(G)$, there is a bag B_r with $v_i, v_{i+1} \in B_r$. Without loss of generality we have $r \in T_1$. But then we have $v_{i+1} \in B(T_1)$ and $v_{i+1} \in B(T_2)$ and hence, as observed above, $v_{i+1} \in B_x \cap B_y$, contradicting our assumption that P is a path in $G - (B_x \cap B_y)$.

We now construct an order from the decomposition with the desired strong reachability properties. We assign any vertex of T to be the root of T. With any vertex $v \in V(G)$ associate the vertex $T(v) = x \in V(T)$ such that such that x is minimal in the tree order with $v \in B_x$. We now define a partial order on $V(G)$ such that $v < w$ if $T(v) < T(w)$, or if $T(v) = T(w)$, we break ties arbitrarily. Now let σ be any linearization of this partial order. We claim that $|\mathrm{SReach}_n[G, \sigma, v]| \leq k$ for all $v \in V(G)$.

Let $v \in V(G)$ and let $w \in \mathrm{SReach}_n[G, \sigma, v]$. As w is not larger than v with respect to σ, we have $T(w) \leq T(v) = x$. Now by definition of σ and as B_x (in particular) is a separator it follows that $w \in B_x$. Hence $\mathrm{col}(G, \sigma) \leq k$. □

Problem 4. Let G be an n-vertex graph. Prove that $\mathsf{tw}(G) \leq \mathsf{td}(G) \leq (\mathsf{tw}(G)+1) \cdot \log n$.

Proof. Without loss of generality we may assume that G is connected. For the first inequality, let F be a tree-depth decomposition of G (which is a tree, as we assume that G is connected). Let L be the set of leaves of F. Let ρ be an embedding of F in $\mathbb{R}^2$ so that we have an order on L. We define a tree-decomposition of G as follows. The tree T is a path containing the leaves L from left to right in the embedding. Now let $B_x = \{y : y \leq_F x\}$. Then $|B_x| \leq \mathsf{td}(G)$ as desired. Clearly, every edge is contained in some bag and every vertex is contained in some bag. Furthermore, $B^{-1}(v)$ is connected for every $v \in V(G)$, as v is contained in exactly those B_x such that x is a leaf of the subtree rooted at v. By our ordering of the leaves on a line in $\mathbb{R}^2$ this is a connected subpath.

We have proved in the previous exercise that each bag of a tree-decomposition is a separator of G. This implies that there exists a bag B_t such that each component of $G - B_t$ contains at most $\lfloor n/2 \rfloor$ vertices. Put the vertices of B_t as the smallest vertices in the tree-depth decomposition and continue recursively on the components of $G - B_t$. □

Problem 5. Let $\mathcal{C}$ be a class of graphs of tree-width at most k. Prove that it can be tested in linear time whether $G \in \mathcal{C}$ is 3-colorable.

Proof. Let $G \in \mathcal{C}$ be given. According to a theorem of Bodlaender, we can compute a tree-decomposition of width k of G in time $f(k) \cdot n$. We choose an arbitrary node $r \in V(T)$ as root and consider the decomposition to be a rooted tree-decomposition. If $t \in V(T)$ is not the root and e is the edge $\{t, t'\}$ on the path from the root r to t, then we denote by T_t the component of $T - e$ that does not contain r.

The algorithm computes, starting from the leaves, for all $t \in V(T)$ the following information.

Col(t)**:** The set of all valid 3-colourings of $G[B_t]$.

ExCol(t)**:** The set of all colourings from $Col(t)$, which can be extended to a valid 3-colouring of $G\big[B(T_t)\big]$.

Obviously, G is 3-colourable if, and only if, $ExCol(r) \neq \emptyset$.

The data structures $Col(t)$ and $ExCol(t)$ can be computed as follows. Let $t \in V(T)$. $Col(t)$ contains all valid 3-colourings of $G[B_t]$, i.e. all functions $B_t \to \{1,2,3\}$, which are proper colourings of $G[B_t]$. There are at most 3^{k+1} such functions and for each of these we can decide in time $\mathcal{O}((k+1) \cdot k)$ whether they form a proper colouring. The running time to compute $Col(t)$ for a fixed bag $B(t)$ therefore is $\mathcal{O}(3^{k+1} \cdot (k+1)k)$.

If t is a leaf of T, then $ExCol(t) = Col(t)$. Otherwise, let $t_1, \ldots, t_m$ be the successors of t in T_t. Suppose $ExCol(t_i)$ has already been computed. We compute $ExCol(t)$ as follows. For each colouring $c \in Col(t)$ we test whether for every $1 \leq i \leq m$ there is a colouring $c_i \in ExCol(t_i)$ such that $c(v) = c_i(v)$ for all $v \in B_t \cap B_{t_i}$. In this case we add c to $ExCol(t)$.

Clearly, after the above computation, every colouring in $ExCol(t)$ can be extended to a colouring of $G\big[B(T_t)\big]$. Furthermore, the restriction c_{B_t} of every proper 3-colouring c of $B(T_t)$ to B_t is a proper colouring appearing in $ExCol(t)$.

The time needed to compute $ExCol(t)$ is bounded by $3^{k+1} \cdot 3^{k+1} \cdot m \cdot k$.

In total, the algorithm requires for each edge in $E(T)$ time $\mathcal{O}(3^{2(k+1)} \cdot k)$. We may assume that the decomposition has only $|V(G)|$ nodes and therefore a total running time of $2^{p(k)} \cdot n$ for a polynomial p and $n := |V(G)|$. □

Problem 6. Let $\mathcal{C}$ be a class of graphs of tree-width at most k and let H be graph. Prove that for every n-vertex graph $G \in \mathcal{C}$ it can be tested in time $f(|H|, k) \cdot n$ whether G contains a subgraph isomorphic to H.

Proof. We can analogously interprete the subgraph relation as a coloring and in the above dynamic programming save with each coloring which vertices still have to be found. □