Treewidth

Recall from the previous lesson :

Are there classes of graphs and some values of k where deciding if a graph in the class is k-colorable does not have the same complexity as finding a k-coloring of the graph ?

Answer : yes! For instance if $k = 4$, on planar graphs. You can decide if a planar graph is k-colorable in constant time, but you need at least linear time to provide such a coloring (you at least need to read the imput).

Motivation

A lot of problems are easy to solve on trees by dynamic programming. For instance, the maximum stable set can be computed in linear time this way :

⚠Consider a tree T with n vertices. Root your tree at any vertex r. Now your tree is a rooted tree, each node (= vertex) v has exactly one father (except the root r), that is its neighbour in the path from r to v. Now v is a *descendent* of w if v is on the path from w to the root r. Now for each vertex v, let $T(v)$ be the sub-rooted-tree rooted at v (that is the rooted tree formed by the descentents of v). We will now compute, for each vertex v, the size $s'(v)$ of a maximum stable set containing v in $T(v)$, and the size $s''(v)$ of a maximum stable set not containing v in $T(v)$. We can then compute $s(v) = \max(s'(v), s''(v))$ the size of a maximum stable set in $T(v)$.

If v is a vertex with sons $v_1 ... v_k$, we compute $s'(v) = 1 + \sum_{i\in\{1,...,k\}}(s''(v_k))$ and $s''(v) = \sum_{i\in\{1,...,k\}}(s(v_k))$. Of course, we use dynamic programming, that is we compute each value of $s(v)$, $s'(v)$ and $s''(v)$ only once. Now the value we wanted is exactly $s(r)$.

Another way to make algorithm for trees is to use the notion of separator. A balanced separator is a set of vertices whose removal diconnects the graph into smallish components (of size bonded by $c \times n$, where $c < 1$). Indeed, in a tree there is always a vertex whose removal leaves only small connected components.

Lemma. ⚠*Every tree T with n vertices has a vertex v such that removing v leaves T with connected components of size at most $\frac{n}{2}$.*

Proof. (∗)To prove this, just pick a vertex v, and while one component $\mathscr{C}$ of $T \setminus \{v\}$ has size more than $\frac{n}{2}$, replace v with its neighbour u in $\mathscr{C}$. If this does not terminate, it means that there two vertices u and v in the tree such that the component of v in $T \setminus \{u\}$ and the component of u in $T \setminus \{v\}$ both have more than $\frac{n}{2}$ vertices. They cannot have a vertex in common, since otherwise there would be a path in T between u and v not using uv, and thus a cycle in T. □

This can be done repetedely on the components until each component has one vertex only. For each component $\mathscr{C}$ of the previous decoposition, find a new vertex v that separates it into smaller parts, and so on. Now we have a way to rebuild the tree from the small components to the big one, by each time adding only one vertex and going from one component to a component at least twice as large. We can use this decomposition to do dynamic programming : for each vertex v used by the decomposition, compute the largest independent set of each component you got when removing the vertex, while forcing the vertex v to either be or not be in the independant set.

The same kind of algorithm can be applied on graphs that are "close to being a tree". By that we mean that they have some sort of tree-like structure. This "closeness to being a tree" is measured by the *treewidth* of the graph. Informally, the treewidth will be small if and only if one can find small separators.

Treewidth

We will first define treewidth from the notion of chordal graphs.

Definition. ⚠*A graph is* chordal *if and only if every cycle of length at least* 4 *in the graph has a chord. Equivalently, a graph is chordal if and only if it has no induced cycle of length at least* 4.

See Figure 1 for an illustration of simplicial and non-simplicial graphs.

Definition. ⚠*The* treewidth *of a graph G is the minimum, over all chordal supergraph H of G, of the size of a largest clique in H, minus 1 (that is $\omega(H) - 1$).*

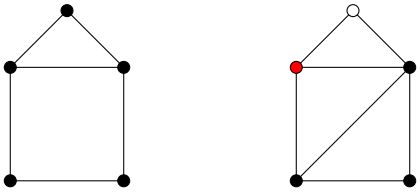

Figure 1: The graph on the left is not chordal, the graph on the right is chordal. The white vertex is simplicial in both, once removed, there is no remaining simplicial vertex on the left, while on the right the red one is still simplicial, and then all vertices are simplicial.

See Figure 2 for an illustration.

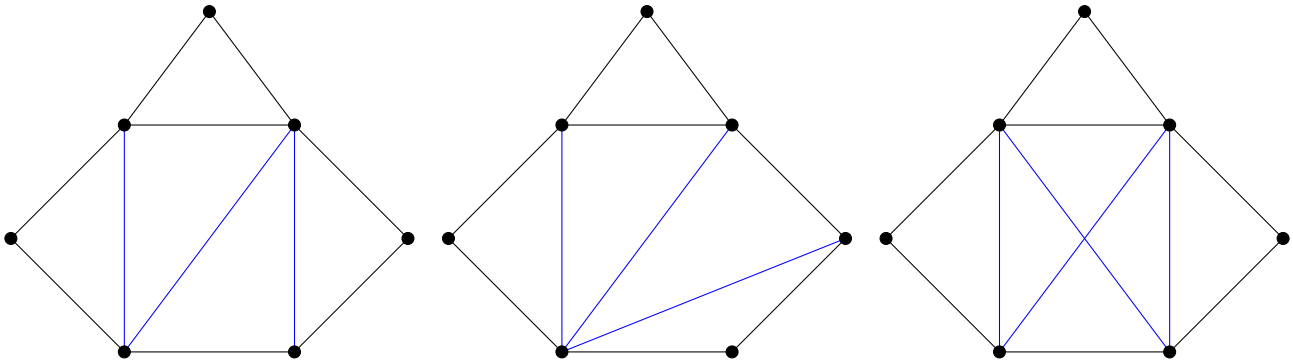

Figure 2: The graph G in black is not chordal. The blue edges are three ways to complete it into a chordal graph (but there are more). The first two have 3 as their clique number (their maximum clique is a triangle). The third one has an induced K_4, and thus its clique number is 4. However, the treewidth is taken as a minimum over all supergraphs, so the treewidth of G is at most 2. Since it's not a tree, there is at least one cycle, and in the completion at least one triangle, so the treewidth of G is also at least 2. So it is 2.

Note that this "minus 1" is such that:

⚠Trees are exactly the connected graphs with treewidth 1 (there is nothing to do since they are chordal, and their largest clique is an edge).

Other example: ⚠Cycles have treewidth 2. You can add non-crossing edges in the cycle to make it chordal.

Definition. ⚠*A* simplicial vertex *is a vertex whose neighbourhood is a clique.*

Characterization. ⚠*A graph is chordal if and only if every induced subgraph has a simplicial vertex.*

See Figure 1 for illustrations of simplicial vertices and of this caracterisation.

(∗)Chordal graphs are also exactly intersection graphs of subtrees in a tree. This means that a graph is chordal if and only if one can build a tree T with some subtrees $T_1, \dots, T_n$ such that $v_i v_j \in E(G)$ if and only if T_i and T_j intersect.

(∗)**Proof that chordal graphs are intersection graphs of subtrees in a tree:** by induction on the number of vertices in G. If G has at most one vertex, the property is trivial. Let v be a simplicial vertex in G. By induction hypothesis we know that $G - \{v\}$ is the intersection graph of some subtrees in the tree. We know that all of the neighbours of v are adjacent, therefore their corresponding subtrees intersect in the tree T. Since T is a tree, this means that there is a point in T that belongs to all of the subtrees corresponding to neighbours of v. Now extend a new branch from this point, and make it belong exactly to the neighbours of v. The subtrees still have the exact same intersection graph $G \setminus \{v\}$, and we can add a piont on this branch as the new vertex v to get the whole graph G.

From this, one can deduce the notion of clique trees in chordal graphs: parts where a specific set of subtrees intersect correspond to a clique, and these cliques are linked in a tree-like fashion.

Definition. (∗)*A* clique tree *of a chordal graph is a tree where each vertex corresponds to a clique of the chordal graph, and such that for any vertex, the cliques containing this vertex form a subtree of the clique tree (i.e. form a connected graph).*

From this we can define a tree decomposition of a graph. Informally a tree decomposition of a graph G is a clique tree of a chordal supergraph of G.

Definition. ⚠ *A* tree decomposition *of a graph G is a tree T whose nodes are labeled by sets of vertices of G, known as* bags*, such that:*

- *For any vertex v of G, the bags containing v induce a connected subtree of T.*
- *For every edge vw of G, there is at least one bag contianing v and w.*

⚠ The width of a tree decomposition is the size of the largest bag minus one.

From this notion, we can derive an alternative definition of treewidth:

Characterization. ⚠ *The treewidth of a graph G is the minimum over all tree decompositions of the width of the decomposition.*

See Figure 3 for an illustration of tree decomposition and their link to chordal supergraphs.

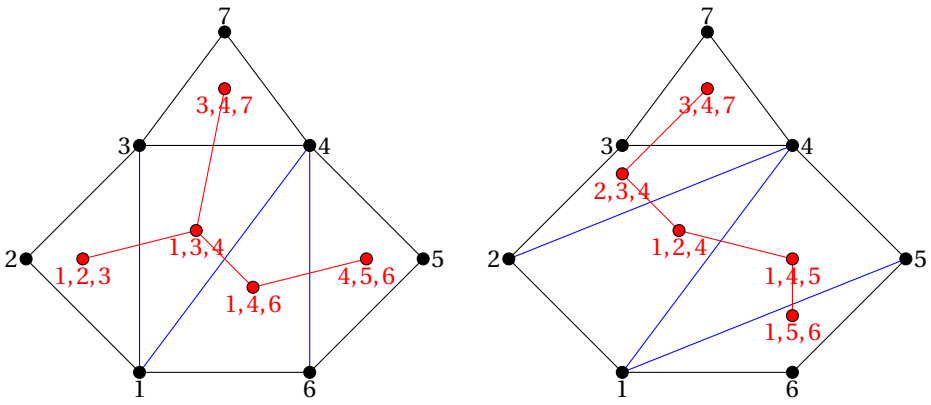

Figure 3: The graph G in black. Two completions into chordal graphs (in blue) and their corresponding tree decomposition (in red). The numbers in red correspond to the vertices in the bags.

⚠ The treewidth of planar graphs is not bounded. In particular, the $n \times n$ square grid has treewidth n.

⚠ For every integer k, the class of graphs with treewidth at most k is minor closed (that is, a minor of a graph with treewidth k has treewidth at most k). Therefore for every k, there is a finite set S_k of graphs such that G has treewidth at most k if and only if G does not have a minor in S_k.

(∗)Actually, a minor-closed family of graphs has bounded treewidth if and only if one of the forbidden minors is planar. The "only if" direction derives directly from the fact that planar graphs do not have bounded treewidth. The "if" direction is less trivial, and not proven in this course.

⚠ One can easily find dynamic algorithms from tree decomposition, just like we can do on trees. The reasonning is pretty similar to the one on trees in the motivation section. Let us present it for maximum independent set.

We root the tree decomposition T, and as above, at a node r. For any node a of the decomposition, this node corresponds to a bag $b(a)$ of vertices, and let us define the set $T(a)$ to be the union of all of the bags of the nodes that are descendents of a. We want to compute, for every node a, and every independent set $S(a)$ of $b(a)$, the maximum size of an independent set S that coincides with $S(a)$ on $b(a)$. To do this, for a given a and $S(a)$, for all son a' of a, we look at all of the ways possible to extend $S(a)$ to an independent set $S'(a')$ of $b(a')$. For each of those, we already computed the corresponding size of a maximum independent set in $T(a')$ respecting $S'(a')$. Now we just need to maximise over all $S'(a')$ and sum over all a' to obtain the maximum independent set S that coincides with $S(a)$ on $b(a)$.

Note that we need to consider all of the independent sets of $b(a)$, and there can be $2^{|b(a)|}$ of them. However, since $|b(a)|$ can be at most the treewidth of G plus one, if the treewidth of G is at most k, we thus only have complexity $O(2^k)$ for each node a, and thus a complexity of $O(n2^k)$ in total.

⚠ More generally, bounded treewidth will mean efficient dynamic programming algorithm, which will mean fixed parameter tractability bounds.

Definition. ⚠ *Let $0 < c < 1$. A* c-balanced separator *in a graph G is a set S of vertices of G such that each component of $G \setminus S$ has at most $c \times |G \setminus S|$ vertices.*

As seen previously, trees have a $\frac{1}{2}$-balanced separator of size 1. From a tree decomposition, one can similarly find a $\frac{1}{2}$-separator of size at most the width of the decomposition. The reasonning is the same as the one on trees, except that we remove a whole bag every time. Therefore, the presence of small balanced separators also implies fast algoritms.

(∗)Actually, one can show that a class essentially has bounded treewidth if and only if it has constant size balanced separators. But there are classes of graphs where the treewidth is unbounded, but we can still find small balanced separators. For instance, on planar graphs, there are balanced separators of size at most a constant times $\sqrt{(n)}$.

On the complexity side, ⚠determining if the treewidth of a graph is at most k is NP-hard. (∗)However, one can get approximations in polynomial time, and determining if the treewidth of a graph is at most k is fixed parameter tractable (parametrized by k).

Some additional interesting notion, for those that want to go further:

- (∗)There is an alternative definition of the treewidth that comes from the notion of *graph searching*. In this variant of the cops and robbers game, there is one robber on the graph that can travel arbitrarily far every round, but only trough edges and trough vertices where there is no cop. The cops must corner the robber. The number of cops necessary is essentially equal to the treewidth of the graph (maybe +1 or -1).
- (∗)There are similar notions of path decompositions and pathwidth. A path decomposition is a tree decomposition where the tree is a path (figures...), and the pathwidth is the minimum width of a path decomposition.