1 What is Treewidth

It is known that many $\mathcal{NP}-Hard$ problems are easy on trees. So, if a graph is "close" to being a tree (in some sense which will become clear later) then solving those problems might be easier on these classes of graphs. In order to define treewidth, we make a few preparations. We begin by defining what we call tree decomposition. A tree decomposition of a graph G is a pair (T, W) where T is a tree and W is a family of subsets indexed by $V(T)$ such that

1. $\bigcup_{\mathbf{t} \in \mathbf{V(t)}} \mathbf{W_t} = \mathbf{V(G)}$
2. Every edge of G has both ends in some W_t, and
3. if $\mathbf{t' \in tTt''}$, then $\mathbf{W_t \cap W_{t''} \subseteq W_{t'}}$

To give a more suggestive picture, we say that all vertices $v \in V(T)$ have bags which contain a subset W_t of vertices of the original graph G. Thus, the above definition says that in a tree decomposition the union of the contents of all these bags (kept at the vertices of the tree T) generates $V(G)$. And corresponding to every edge, there is some bag which contains both of its ends. Finally, if there are vertices $t,\ t',\ t'' \in T$ and t' lies along the unique shortest path between t and t'', then the intersection of bag contents at t and t'' is contained in t'. Alternatively, this can also be read as - for all vertices $v \in G$, the set of bags that contain the vertex v form a subtree of T (why?). Perhaps the most important property of tree-decomposition is that it transfers the separation properties of its tree to the graph decomposed. The following exercise demonstrates this

Exercise 1 Let t_1t_2 be any edge of T and let T_1, T_2 be the components of $T - t_1t_2$, with $t_1 \in T_1$ and $t_2 \in T_2$. Then $W_{t_1} \cap W_{t_2}$ separates $\bigcup_{t \in T_1} W_t$ and $\bigcup_{t \in T_2} W_t$ in G.

The *width* of a decomposition (T, W) is defined as $\max_{\mathbf{t} \in \mathbf{V(T)}} (|\mathbf{W_t - 1}|)$. Having worked our way through the definition of width of a tree decomposition, we are now ready to define treewidth. It is defined to be the minimum value of width over all tree decompositions of a graph G. Treewidth of a graph G will be denoted as $tw(G)$ in the rest of this lecture. To help develop more intuition about this concept, here are a few examples and exercises.

Example 1 $V(T) = t_0$ and $W_{t_0} = V(G)$. (Trivial tree decomposition with everything in the same bag).
Example 2 For a tree $\mathbf{T_0}$, we have $\mathbf{tw(T_0) = 1}$.
Example 3 For a clique $\mathbf{K_n}$, we have $\mathbf{tw(K_n) = n - 1}$.

Exercise 2 If H is a minor of G, then $tw(H) \leq tw(G)$.
Solution Sketch Obvious if H is a strict subgraph of G. If not, a graph isomorphic to H can be obtained by a sequence of edge contractions in G. We just need to show that $tw(G \backslash e) \leq tw(G)\ \forall e = (u, v) \in E(G)$. This follows by just replacing the u and v with w in respective bags (where w is the vertex formed by contracting uv.)

Exercise 3 If (T, W) is a tree decomposition of G and K is a complete subgraph of G, then $V(K) \subseteq W_t$ for some $t \in V(T)$.

Solution Sketch We just need to show (using exercise 1) that given a set $W \subseteq V(G)$, there is either a bag $W_t, t \in T$ that contains W or else there are $t_1, t_2 \in T$ such that the intersection of bags W_{t_1} and W_{t_2} separates some vertices in W.

Exercise 4 If G is simple, then $tw(G) \leq 1 \Leftrightarrow G$ is a forest.
Solution
($\Leftarrow$) See example 2.
($\Rightarrow$) Suppose that G is not a forest. Then there is a cycle in G, and furthermore by contracting edges in the cycle, a cycle of length 3 (or K_3) can be obtained. Thus, the graph has a K_3 minor and therefore any tree decomposition of this graph has this clique in some bag which means $tw(G) \geq 2$ which is a contradiction.

Exercise 5 If G is simple and $tw(G) \leq 2 \Leftrightarrow G$ has no K_4 minor.

Exercise 6 For a $n \times n$ grid graph H, $tw(H) = n$.
Solution We can easily see that it $tw(H) \leq n$. It follows from the following tree decomposition. Keep a vertex t_i with $W_{t_i} = \{v : v \in \ row\ i\}$. Connect these t_i's via a path. That, it is at most n will be proved using the notion of cop robber games which provide an alternative way to study treewidth. Below, we will only prove that $tw(H) \geq n - 1$ as the proof for n is much more technical and so it will be left as an exercise.

2 Cop Robber Games

These games give an alternative characterization of treewidth. We have the following players in this game which is played on an undirected graph $G = (V, E)$
A Robber He is *infinitely* "fast" and moves along the edges of the graph. And he lives in $V(G)$.
k **cops** They move in "helicopters" and are *infinetly* "stupid". (Maybe corrupt, but for the sake of these games we call them stupid).

A short note on the funny terms used above - Robber stands on some vertex of $u \in G$ and at any time, he can run along any path to any other vertex v. However, he is not permitted to run through the cop-occupied vertices. There are k cops and at any given point in time, they are either at some vertex or are in the ***helicopters***; the point of helicopter being to allow the cop to land at any arbitrary vertex - they are not constrained to move along the paths of the graph. The cops are ***stupid*** in the sense that they announce the location to which they would move before doing it. While they leave a vertex v_0 and are in the air, v_0 is considered cop-free and the robber can run through this vertex if he can find some cop free path passing through v_0. Multiple cops on the same vertex is allowed.

The objective of the cops is to ***capture*** the robber. In order to do that, they first need to occupy all the vertices adjacent to the vertex where the capture is to take place, because otherwise the robber would be able to avoid capture by running away to a different vertex. Cops and robbers make their moves alternately. A cop move consists in them landing at announced vertices, announcing where they would go next if they have not captured the robber so far and then getting up in the air again in their helicopters (some cops may choose to stay). Robber's move consists in moving from his old vertex u a cop-free vertex v via a cop-free path (if

possible) to avoid capture. Below, we consider the case where the robber is always ***visible*** to the cops - the difficulty lies only in cornering him somehow. There are other variants where the robber is invisible but we will not discuss it in this lecture.

We are interested in the following question – *How many cops are needed to capture the robber?*. We proceed by observing that 2 cops suffice in a forest. The argument is pretty easy. The cops just identify the tree component on which robber is. Say, he is in component T at vertex v. The first cop lands at some vertex $u \neq v$ in T. This makes all the vertices w where $u \in wTv$ and wTv is a simple path inaccessible to the robber. Further, the second cop lands at a vertex $u' \in uTv$. Thus, the cops zero into the robber's position and it follows that the robber eventually gets caught. A natural generalization of this statement says the following

Exercise Show that $(k+1)-$cops suffice in a graph G with $tw(G) = k$.
Solution We will just outline major ideas in the solution. It just more or less mimics the 2 cops proof for a forest. The idea is to have the cops land on W_t corresponding to some $t \in V(T)$. Next, we look at the components $T_1, T_2 \ldots T_k$ of the tree T obtained on deletion of vertex t and the corresponding graph $G \backslash W_t$. The crucial point is that all the components of $G \backslash W_t$ will have their vertices in the same $V(T_i)$ ($i \in [k]$) where $V(T_i)$ denotes the union of the bags of all the vertices in the subtree rooted at T_i.

That, this is true can be shown by contradiction. Suppose, there is some component C_0 of $G \backslash W_t$ with one vertex $u \in V(T_j)$ and other vertex $v \in T_{j'}$. Since C_0 is connected we can very well assume that $(u, v) \in E(G)$. Thus, there is some bag W_r which contains both u and v. Further, we may assume $r \notin V(T_j)$ – which means W_r, the bag containing u and v, is not connected to the bag W_j which contains u. But then this contradicts condition 3 in tree decomposition. With this thing taken care of, the rest of the proof is just like the previous proof where we keep one cop on the ground and move the other towards the robber. Clearly, the robber is some $V(T_i)$. Let t' be the neighbor of t in the component T_i. In order to escape T_i, robber must run along $W_t \cap W_{t'}$ and we let the cops stay on these vertices. The other cops move onto $W_{t'} - W_t$ and thus zero in on the robber finally capturing him.

Cor $tw(n \times n)$grid $\geq n - 1$.
Proof $\exists$ a strategy for the robber against $n-1$ cops. This shows, by the previous exercise that $tw(G)$ is at least $n-1$. The strategy is simple – at every point in time, there is some cop-free row and some cop-free column. We will show that the robber can always go to this location. Initially, this is clear. Suppose that at some point one or more cops enter the free row and/or free column where the robber is located. Then the robber can move along the previously free row to the to-be free column and within this column to the to-be free row and escape from getting captured and this completes the proof.

Let us define a strategy for the robber to mean a mechanism which gives a list of safe places for the robber given that cops occupied a set $X \subseteq V$ which helps him avoid capture indefinitely no matter what sequence of moves is played by the cops. Next, we look at

2.1 A possible strategy for the robber against $k-1$ cops

This motivates the following definition
A haven of order k in a graph $G = (V, E)$ is a mapping

$$\beta \colon X \to C(G \backslash X)$$

where $|X| < k$ $C(G \backslash X)$ refers to some component of $G \backslash X$. In addition to this, β is also required to satisfy the following condition which we call (*)

$$\textbf{if } \mathbf{X} \subseteq \mathbf{Y} \; with \; |Y| \; < k, \textbf{then } \beta(\mathbf{Y}) \subseteq \beta(\mathbf{X}) \; (*)$$

It is easy to see that a haven of order k provides the robber with an escape strategy against less than k cops. If the cops reveal the set $X \subseteq V$ locations they will go next, the robber would go to $\beta(X)$. That this can always be done when cop-occupied locations grow from X to Y, with $X \subseteq Y$, follows because of the property (*).

We immediately see that if a graph G has treewidth at most $k-1$ then it must have a haven of order k. This is because $tw(G) < k$ means that there is no escape strategy for the robber against k cops which means no haven of order k. The converse of this statement if also true but harder to prove, and we will not do that here.

At this point as a curious aside, we mention a few thing(s) about the cop-robber games and tree decompositions. We begin by discussing the notion of monotonicity in cop-robber games. Cops can search monotonely for the robber in following two senses

1. They can ensure that robber does not ever enter the positions visited by the cops, or
2. They can ensure that once they leave a vertex, they never need to revisit it.

And then we have the following theorem – *If k cops can search at all, they can do it monotonely in both the above respects.* This can be done with the help of tree decompositions and we will not go into its proof either. Finally, we mention the following metatheroem for treewidth which tells us how useful a parameter it really is

Theorem Let ϕ be any graph property that is definable in MSO logic. For any fixed k, the there is a linear time algorithm for testing the property ϕ on any graph of treewidth at most k.