Exercise 5, Discrete Mathematics for Bioinformatics

Sascha Meiers, Martin Seeger

Winter term 2011/2012

3.1 Tree decomposition

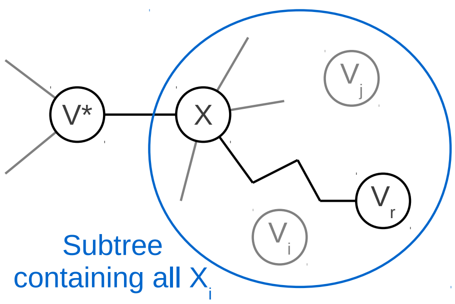
Let $G = (V, E)$ be a graph with $V = v_1, \ldots, v_n$ and $E = \binom{V}{2}$. We'll prove that the graph's tree width is $n-1$, meaning that any tree decomposition of G contains at least one piece with n elements.

Proof: Given a tree decomposition T, let V^* be the largest piece and assume that $v_n \notin V^*$ without loss of generality. We know by edge coverage property that there must be pieces containing v_n and v_i at the same time, for $1 \leq i \leq n-1$. Let these edges be covered by the k pieces $V_1, \ldots, V_k$ with $2 \leq k \leq n-1$ (but there could also be other pieces). k cannot be one since then V_k would be larger than V^*.

Now we analyze the structure of the tree T and regard two cases:

1. The piece V^* is somewhere "between" the pieces V_i. This means, there is at least one pair (i, j) such that V^* lies on a path from V_i to V_j. V_i and V_j both contain v_n, but V^* does not. This hurts the coherence property $\Rightarrow$ contradiction

2. The piece V^* is not between the pieces V_i. It could be a leaf of the tree, but there could also be further pieces connected to it. However, V^* is connected to the subtree that contains all V_i by a single edge. Let X be the next piece on the path from V^* to any V_i. Usually[1] the piece X is missing at least one node $v_l \neq v_n$. We know that there is at least one piece V_r in the subtree that contains v_l, and we also know $v_l \in V^*$. By coherence property, X would also have to contain $v_l \Rightarrow$ contradiction.



[1] X can contain at most $n-1$ nodes, so at least one node is missing, as stated. Theoretically, the missing node could be v_n. But in this case, we have $X = V^*$ and (if this is allowed at all) the argumentation (case 1 or 2) can be applied on X itself as the largest set.

3.2 Tree decomposition

3.3 Bellman-Ford

a)

	z	u	v	x	y
$k=0$	$(0)_z$	∞	∞	∞	∞
$k=1$	0_z	6_z	∞	(7_z)	∞
$k=2$	0_z	6_u	(4_x)	7_x	2_u
$k=3$	0_z	(2_v)	4_v	7_x	2_y
$k=4$	0_z	2_u	4_v	7_x	(-2_u)
$k=5$	(0_y)	2_u	4_v	7_x	-2_y

Example: the shortest path from z to z is (z, x, v, u, y, z) with weight zero (see traceback).

b) Let f be the result of the Bellman-Ford-algorithm started in node s. We'll prove equivalency between the two statements (I) and (II):

(I) The graph contains a circle of negative weight reachable from s.

(II) There is a node v with $f_n(v) < f_{n-1}(v)$.

(II) $\Rightarrow$ (I) Let $v \neq s$ be the node for which we have $f_n(v) < f_{n-1}(v)$. By definition, $f_{n-1}(v)$ computed already the shortest walk π^* from s to v using at most $n-1$ arcs. We say walk, because ensuring π^* to be a path would require the graph not to contain negative circles. But we consider π^* to be a proper path, since if it was not we would already have shown the implication.

So we assume π^* is the shortest of all paths from s to v, considering every possible path (since up to $n-1$ arcs suffice to cover every path). If $f_n(v) < l(\pi^*)$ then there is a new walk π° that uses exactly n arcs (if it used at most $n-1$ arcs, it would have been found in f_{n-1}). A walk of n arcs must contain a cycle C. And it contains a path from s to v. Now the weight $l(\pi^\circ)$ is made up of the weight of this cylce C plus the weight of the path π from s to v that is contained in π° (Note that this decomposition need not be edge disjoint, but the edges that belong to the circle and the path at the same time are walked twice during π°, so the the addition of the lengths holds). Now we have

$$l(\pi^\circ) < l(\pi*) \Rightarrow l(\pi^\circ) < l(\pi) \Rightarrow l(\pi) + l(C) < l(\pi) \Rightarrow l(C) < 0$$