Math 372 lecture for Friday, Week 11

## A tree bijection

Let $G = (V, E)$ be an undirected, loopless, connected multigraph, and let $s \in V$ be designated as the *sink* vertex. Let $\widetilde{V} := V \setminus \{s\}$, the set of non-sink vertices. A *sandpile configuration* on $G$ is an element $c \in \mathbb{Z}\widetilde{V}$ and write $c = \sum_{v \in \widetilde{V}} c(v)v$ with $c(v) \in \mathbb{Z}$ Given two configurations $c, c'$, we write $c \geq c'$ if $c(v) \geq c'(v)$ for all $v \in \widetilde{V}$. In particular, $c$ is *nonnegative* if $c \geq 0$, where 0 denotes the 0 configuration. Fixing an ordering $v_0, \ldots, v_{n-1}$ of $V$ and taking $s = v_0$, we have that $\mathbb{Z}\widetilde{V} \simeq \mathbb{Z}^{n-1}$ by identifying each configuration $c$ with the vector $c = (c_1, \ldots, c_{n-1})$ where $c_i := c(v_i)$. The *degree* of a configuration is

$$\deg(c) = \sum_{v \in \widetilde{V}} c(v) = \sum_{i=1}^{n-1} c_i.$$

Let $c$ be a configuration. If $v \in \widetilde{V}$, we can *fire* or *topple* $v$ in $c$ to get a new configuration $c'$ by

$$c' = c - \widetilde{L}e_v$$

where $\widetilde{L}$ is the reduced Laplacian of $G$ with respect to $v$ and $e_v$ is the $v$-th standard basis vector. This means that: (i) if $w$ is a neighbor of $v$, then $c'(w) = c(w) + a_{vw}$ where $a_{vw}$ is the number of edges from $v$ to $w$, (ii) $c'(v) = c(v) - \deg_G(v)$, and (iii) $c'(w) = c(w)$, otherwise. If $S \subseteq \widetilde{V}$, then we can perform a *set-firing* on $c$ by toppling each vertex in $S$:

$$c' = c - \widetilde{L}\chi_S,$$

where $\chi_S$ is the indicator vector for the set $S$ (so $\chi_S(s) = 1$ of all $s \in S$, and is 0, otherwise). A *sandpile* is a nonnegative sandpile configuration, i.e., an element $c \in \mathbb{N}\widetilde{V}$. If $c$ is a sandpile, and $v \in \widetilde{V}$, then firing $v$ in $c$ is *legal* if the resulting configuration is a sandpile, i.e., no vertex becomes negative after firing. A set-firing by $S \subseteq \widetilde{V}$ is *legal* if the resulting configuration is a sandpile. A sandpile is *superstable* if it has no legal non-empty set-firings.

One might think that to determine whether $c$ is superstable, one would need to check all $2^{n-1} - 1$ nonempty subsets of $\widetilde{V}$ to set whether any was legal. However, the check can actually be done in at most $n$ steps:

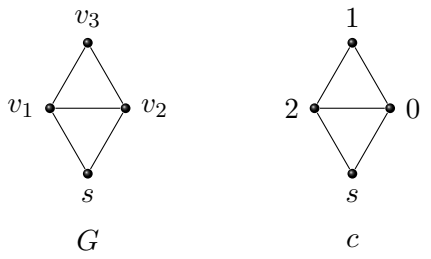
**Dhar's algorithm.** Let $c$ be a sandpile. To determine if $c$ is superstable, start with $S := \widetilde{V}$ and fire $S$ to obtain the configuration $c'$. If $c' \geq$, then $S$ is a legal set-firing, and hence $c$ is not superstable. Otherwise, there exists $v \in S$ such that $c'(v) < 0$. Replace $S$ by $S \setminus \{s\}$, and repeat: set-fire $S$ on $c$ to obtain a new $c'$. If $c'$ is a sandpile, then $c$ is not superstable. Otherwise, there exists $u \in S$ such that $c'(u) < 0$. Remove $u$ from $S$, etc. In the end, we either find a legal non-empty set firing, or $S = \emptyset$, in which case $c$ is superstable.

Recall the isomorphism

$$\begin{aligned} \mathrm{Jac}(G) &\xrightarrow{\sim} \mathbb{Z}^{n-1}/\operatorname{im}(\widetilde{L}) \\ (c_0, \ldots, c_{n-1}) &\mapsto (c_1, \ldots, c_{n-1}) \end{aligned}$$

**Theorem.** There is exactly one superstable sandpile in each equivalence class of $\mathbb{Z}^{n-1}$ modulo the image of $\widetilde{L}$. Thus, the superstables form a complete set of distinct representatives for the elements of $\mathrm{Jac}(G)$. Let $\mathrm{Sp}(G)$ denote the set of superstable elements of $G$. Define the sum of superstables $c$ and $c'$ to be the superstablization of $c + c'$. This turns $\mathrm{Sp}(G)$ into a group isomorphic to $\mathrm{Jac}(G)$.

**Example.** Consider the configuration $c = (2, 0, 1)$ on *diamond graph* $G$:




Apply Dhar's algorithm to see if $c$ is superstable. Firing the set $S = \widetilde{V}$ yields the configuration $c' = (1, -1, 1)$ for which $c'(v_2) = -1 < 0$. So let $S = \widetilde{V} \setminus \{v_2\} = \{v_1, v_3\}$. Firing $S$ on $c$ now yields $c' = (0, 2, 0) \geq 0$. So we have found a non-empty legal set-firing for $c$. Therefore, $c$ is not superstable. However, one may now check that $(0, 2, 0)$ *is* superstable. Since set-firing changes a configuration by subtracting an element of $\operatorname{im}(\widetilde{L})$, it does not change the configuration's equivalence class modulu $\operatorname{im}(\widetilde{L})$. Thus, for example $(2, 0, 1) = (0, 2, 0)$ in $\mathrm{Jac}(G)$, i.e., $(0, 2, 0)$ is the superstable representative of $(2, 0, 1))$ in $\mathrm{Jac}(G)$.

**Example.** For the diamond graph, above, we have $|\mathrm{Jac}(G)| = \det(\widetilde{L}) = 8$. Find the eight superstable elements on $G$. In fact, $\mathrm{Jac}(G) \simeq \mathbb{Z}/8\mathbb{Z}$. Find a superstable generator for $\mathrm{Jac}(G)$.

**Depth-first search tree bijection and tree inversions.** By the matrix-tree theorem, the number of elements of $\mathrm{Jac}(G)$ is the number of spanning trees of $G$, and by the theorem, above, the superstables serve as a complete set of representatives for $\mathrm{Jac}(G)$. We will now describe a bijection between superstables and spanning trees. This bijection will have the further property that the degree of the superstable will be related to a statistic for the corresponding tree called the *inversion number*.

In the following we will refer to our vertices by their indices. So vertex $j$ means $v_j$. To describe our tree-bijection, take a superstable configuration $c$ and think of it as an assignment of firefighters to vertices. We think of the edges and vertices as being made of a flammable

material, perhaps wood. Light the sink vertex on fire and let is spread along edges a controlled manner, which we will now describe, burning one edge at a time. The spreading is determined by a *depth-first search* of the vertices of $G$. In detail: at the beginning of each step of the algorithm, there will be a currently *active* burnt vertex $i$ (initially $i = s$). To find the next edge to burn, find the maximal (in numerical order) vertex $j$ such that: (i) $j$ is unburnt, and (ii) $\{i, j\}$ is an unburnt edge. (We consider the case there is no such $j$ below.) Burn the edge $e := \{i, j\}$. If the number of burnt edges incident on $j$ is now greater than $c(j)$, then the firefighters at $j$ are overwhelmed and abandon the vertex. In that case, $e$ is added to the tree we are constructing, and $j$ is burnt, becoming the active vertex for the next step in the algorithm. If not, the next step of the algorithm proceeds with $i$ again the active vertex.

If there are no vertices $j$ adjacent to $i$ meeting the two criteria specified above, the algorithm proceeds by recursively backtracking: the vertex $i' \neq i$ that was active just before $i$ became active is set as the active vertex, and the algorithm proceeds as before. (The vertex $i'$ will be the unique vertex adjacent to $i$ in the tree built so far.)

In any event, the algorithm halts as soon as all vertices are burnt, returning a spanning tree of $G$. A precise description is provided in Algorithm 1, displayed below.

**Algorithm 1 Depth-first search burning algorithm**.

INPUT:
  $G = (V, E)$ – simple undirected graph with $V = \{0, \ldots, n\}$
  $s \in V$ – sink vertex
  $c \in \mathbb{N}\widetilde{V}$ – sandpile on $G$ with respect to $s$
OUTPUT: `tree` – tree of $G$, a spanning tree iff $c$ is superstable
**initialization:**
  `burnt_vertices` $= \{s\}$, `burnt_edges` $= \emptyset$, `tree` $= \emptyset$
**execute** DFS_FROM$(s)$
**return** `tree`

DFS_FROM
**function** DFS_FROM$(i)$
  **for all** $j$ adjacent to $i$ in $G$, from largest numerical value to smallest **do**
    **if** $j \notin$ `burnt_vertices` **then**
      **if** $c(j) = 0$ **then**
        append $j$ to `burnt_vertices`
        append $(i, j)$ to `tree`
        DFS_FROM$(j)$
      **else**
        $c(j) = c(j) - 1$
        append $(i, j)$ to `burnt_edges`

**Example.** Figure 1 considers the depth-first search algorithm for the superstable $c = (1, 0, 2, 1)$ on the complete graph $K_5$. The graph and its vertex labels appear in the bottom right of the figure. The sink vertex, 0, is lit and its neighbors are probed in reverse numerical order. Firefighters protect vertices 4 and 3, but there are none at vertex 2. So the edges connecting 0 to these vertices are burnt. The vertex 2 is burnt and the edge $\{0, 2\}$ becomes part of the tree. Vertex 2 becomes the active vertex, and the algorithm continues.

The largest unburnt neighbor of 2 is vertex 4, and the single firefighter at vertex 4 is already occupied with the burnt edge $\{0, 4\}$. So that firefighter is overwhelmed. The edge $\{2, 4\}$ is burnt and added to the tree. The vertex 4 is newly burnt and becomes the active vertex. The unburnt edges joining 4 to unburnt neighbors are $\{3, 4\}$ and $\{1, 4\}$. There are sufficient

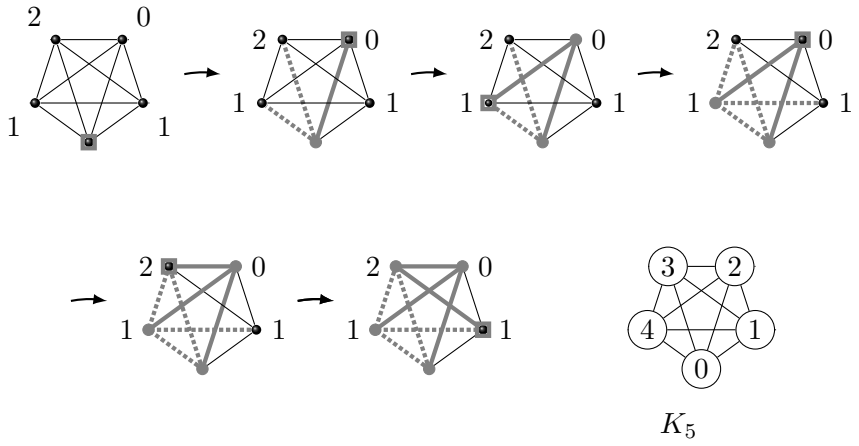



Figure 1: The depth-first search algorithm for the superstable $c = (1, 0, 2, 1)$ on $K_5$. The gray rectangle marks the active vertex, all gray edges are burnt, and solid gray edges are tree-edges.

firefighters to protect vertices 1 and 3. So these edges are burnt but not vertices 1 and 3. We then backtrack: vertex 2 again becomes the active vertex, and the algorithm continues from there to completion. In the figure, the edges of the resulting spanning tree are shown in solid red. Dotted red edges are edges that were burnt but did not become part of the tree.

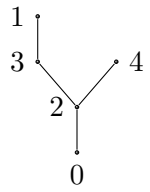
**Tree inversions.** Let $T$ be a tree with vertices $V = \{0, \ldots, n\}$, and pick a root/sink vertex $s \in V$. If $i, j \in V$ and $i$ is on the unique path from $j$ to $s$, then $i$ is an *ancestor* of $j$, and $j$ is a *descendant* if $i$. If $i$ is an ancestor of $j$ and $\{i, j\}$ is an edge of $T$, we say $i$ is the *parent* of $j$ and $j$ is a child of $i$. Each non-root vertex of $T$ has a unique parent, but vertices may have many children.

**Definition** An *inversion* of the rooted tree $T$ is an ordered pair $(i, j)$ of vertices such that: (i) $i > j$, (ii) $i$ is not the root vertex[1], and (iii) $i$ is an ancestor of $j$. The number of

[1]We will usually choose 0 as the root vertex, in which case condition (i) implies condition (ii).

inversions of $T$ is the *inversion number* for $T$.

**Example.** In Figure 1, the depth-first burning algorithm produces the following tree rooted at vertex 0:




This tree has two inversions, $(2,1)$ and $(3,1)$, so its inversion number is 2.

**Exercise.**

1. Find all sixteen trees with vertices $\{0,1,2,3\}$, or equivalently, the sixteen spanning trees of the complete graph $K_4$ on this vertex set.
2. Describe the sixteen superstable sandpiles on $K_4$.
3. Let $\tau_k$ denote the number of these trees with inversion number $k$, and let $h_k$ denote the number of superstables on $K_4$ with degree $k$. Verify the following table:

| $k$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\tau_k$ | 6 | 6 | 3 | 1 |
| $h_k$ | 1 | 3 | 6 | 6 |

.

Note that the inversion and degree counts in the table in Exercise are the same but in reverse order. This is indicative of a general phenomenon first proved by Kreweras: Let $g = n(n-1)/2$, the genus (cycle rank) of the complete graph $K_{n+1}$. Let $\tau_k$ denote the number of spanning trees on $V = \{0,\ldots,n\}$ with inversion number $k$, and let $h_k$ be the number of superstables on $K_{n+1}$ of degree $k$. Then,

$$\tau_k = h_{g-k} \tag{1}$$

for $0 \le k \le g$.

To try to generalize Kreweras' formula, fix an arbitrary graph $G$ of the type considered in this section. Let $\tau_k$ be its number of spanning trees with inversion number $k$, let $h_k$ be the number of superstables of $G$ with degree $k$, and let $g := |E| - |V| + 1$ be its *genus* (or cycle rank). It turns out that it is *sometimes* the case that equation (1) continues to hold, but not always. For a case where it does not hold consider the house graph as pictured in Figure 2 with its 11 spanning trees. The genus of the house graph is 2. The inversion numbers and degree counts for this graph are:

| $k$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\tau_k$ | 4 | 3 | 3 | 1 |
| $h_k$ | 1 | 4 | 6 | 0 |

.

The problem is that our notion of an inversion of a spanning tree does not take into account the structure of $G$. The appropriate generalization is:

**Definition** Let $G$ be a graph (simple, connected, undirected, with the vertex set $\{0, \ldots, n\}$ and fixed root/sink vertex $s$), and let $T$ be a spanning tree rooted at $s$. An inversion $(i, j)$ of $T$ is a *$\kappa$-inversion* if the parent of $i$ is adjacent to $j$ in $G$. The *$\kappa$-inversion number*, $\kappa(T) = \kappa(G, T)$, is the number of $\kappa$-inversions of $T$.

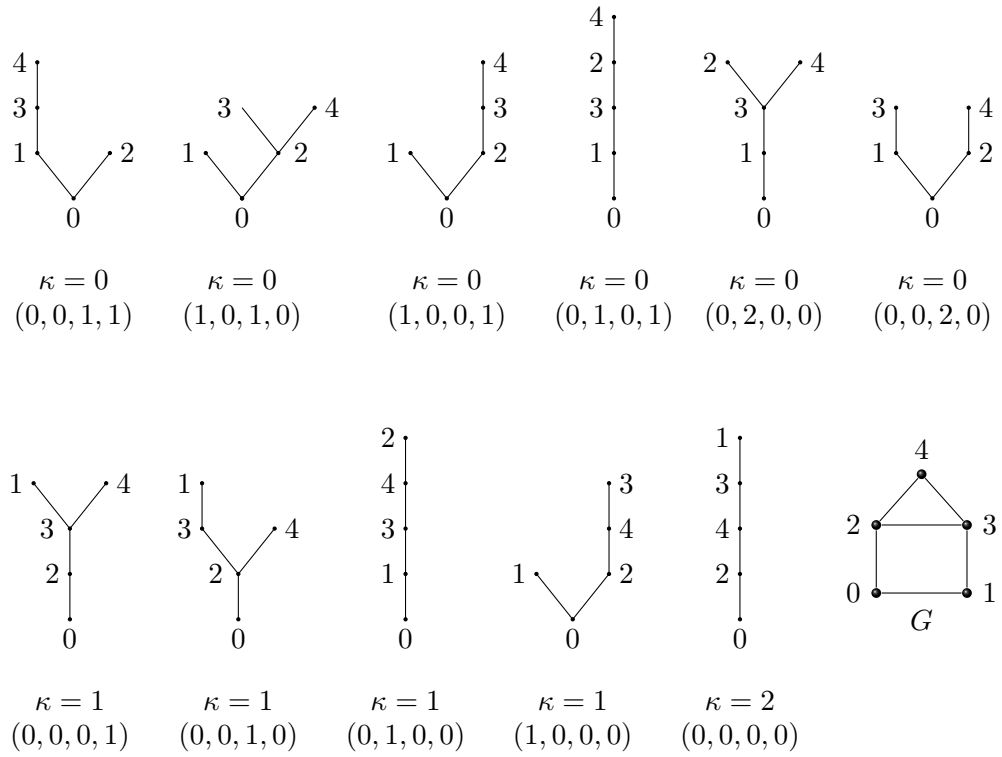



Figure 2: The 11 superstable sandpiles on the house graph, $G$, with their $\kappa$-inversion numbers and corresponding spanning trees (via the depth-first search burning algorithm).

**Exercise.** Verify the $\kappa$ inversion numbers in Figure 2. For instance, the inversion $(3, 2)$ of the spanning tree  is not a $\kappa$-inversion since the parent of 3 in the tree is 1, and $\{1, 2\}$ is not an edge of $G$. Hence, the inversion number of $T$ is 1, but $\kappa(T) = 0$.

We can now state our main theorem relating the depth-first search bijection to $\kappa$-inversions:

**Theorem** (Chiao-Yu Yang, Kuai Yu, P., 2017) The depth-first search algorithm, Algorithm 1, is a bijection between superstables and spanning trees of $G$. If $T$ is the spanning

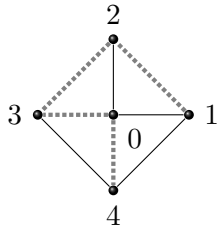
tree corresponding to a superstable $c$, then

$$\kappa(T) = g - \deg(c).$$

**Exercise.**

1. Pick any superstable on the house graph $G$ of Figure 2. Use the depth-first search algorithm to find its corresponding tree, and verify that the formula in Theorem holds.

2. Let $W$ be the following graph with spanning tree $T$ in dashes:




   Let vertex 0 be the root/sink. Find the superstable $c$ that is in bijection with $T$ via the depth-first search algorithm, and verify the formula in Theorem holds in this case.