SURA 2023: Michael Pilson

Michael Pilson

July 2023

Theorem 0.1. *Put theorem statements inside theorem environments.*

JK: This is progress! See if you can formulate a proof of something like: "If there are $q+1$ blue vertices (and all other vertices are white) in $q+1$ different main branches, then Blue can use Rule 3 until the root is blue."

Proof. Consider a situation where the game is being played on a k-ary tree with $q = k-1$ or $q+1 = k$. With general assumptions of $k > 2$ and $h > 2$.

*** = Look at

To begin let's spend $q+1$ or k tokens on the leaflets of the $h = 1$ branches (Main branches). Because these are leaflets (they have no children) and due to the nature of a tree they have one neighbor their parent. Let's enact rule 2 to make the corresponding $h-1$ vertices blue. Since these $h-1$ vertices have $k-1$ white children and a white parent, they are active. By lemma 2.4, we know we can enact rule 3 successfully to gain more blue vertices as we have $q+1$ active vertices. Passing one leaflet for each $h-1$ vertex, white will pass at least one back (One being the most optimal for white and least optimal for blue as it limits blues choices***). We can then enact rule 2 to force this leaflet. Assuming white passes the leaflets for one $h-1$ blue vertex (As they will want to contain blue***), this process will repeat until all the leaflets are filled and the $h-1$ blue vertex becomes inactive as its only neighbor is its white parent. Thankfully, we can now use rule 2 as it only has one white neighbor to make its parent blue. This new parent,$h-2$ like the $h-1$ vertex we just dealt with has $k-1$ white children and a white parent; most importantly, it is active. At this point we still have $q+1$ active vertices to play with, so we can still use rule 3. The process repeats going upward retaining the same amount of active vertices and moves downward as the child of the $h-2$ blue vertex, now a $h-1$ blue vertex is active as it has the k h leaflets. In essence, with each new iteration upward, the number of active vertices remain the same.

This process of filling the main branch blue continues until you reach top of the main branch, or the $h = 1$ and rule 2 into the center of the tree. At this point, white will shift focus to another branch.(If white were to continue with these chain of events, white would be passing components that would lead into other main branches which start expanding blue's number of active vertices at a higher rate***). Assuming white now switches to another branch and the pattern continues, blue still has $q+1$ active vertices, as the center is now active, causing the process to repeat again until that branch has now filled and another neighbor of the center is filled.(k dependent as if $k = 2$ could use rule 2.***) However, after one more iteration of this pattern, with a main branch turning blue, blue will lose an active vertex. As the branch fills out and blue reaches the center, it does not gain an active vertex as the center is already blue and it can no longer force.

JK: Try to focus in on what has to happen for the number of active vertices to go down.

Two branches are now filled along with the center of the tree. However, now blue has only q active vertices (k dependent as if $k < 4$ could possibly use rule 2***), as the center is still active and there are $k-2$ or $q-1$ branches left white and since each contain an active vertex, q active in total. Since only blue vertices left have white neighbors rule 2 is not an option so must spend more tokens. Blue should spend this on one of the leaflets of the remaining white branches such that the existing active vertex and the new

one, after using rule 2 on the $h-1$ vertex, are on different $h=2$ branches within the main branch. This is due to the fact that in replicating the same strategy as previously seen, the two being on the same branch would eventually intersect somewhere up the main branch. This intersection acts as a bad trade since you are losing two active vertices to only gain one more in return which forces additional token spending to keep enacting rule 3. To avoid this for as long as possible, they should be spent on separate $h=2$ branches where they will fill out everything below them and leave minimal white vertices (*** Proof? Seems intuitive). Now they will intersect at $h=1$ returning to q active vertices with $k-2$ $h=2$ subbranches left along with the remaining main branches. Blue is now back down to q active vertices and must spend an additional token. Blue should spend this on a different main branch as to not reduce the active vertex count on the main branch which had two tokens spent on it. The process repeats on this new main branch leaving $k-2$ $h=2$ subbranches unfilled and with q active vertices.

Eventually, this process will terminate when the center becomes inactive after all but one main branch have had two tokens spent on the leaflets. Blue at this point has spent $(k)+(k-3)$. Leaving $((k-2)*(k-3)+k-1)$ subbranches left completely white off $k-2$ main branches along with a final subbranch with just the initial token spent on the leaflet and again with q active vertices.

Blue is now in the endgame and just needs to fill the subbranches. Fill branch or spread out saving active vertices? □

JK: This writeup looks true to the things we discussed in our last meeting. Part of the next step will be to describe how the game works at a higher level, avoiding details of heights etc which are hard to follow. Also, good writing is rewriting, so don't be afraid to go back over what you have written and make changes!