Journal Report 7
10/14/19-10/28/10
Irina Lee
Computer Systems Research Lab
Period 1, White

Daily Log

Tuesday October 15

I continued to test my minor alterations method. Another error made on "Hedwig's Theme" resulted in a tweak on how the algorithm handles calculating hand position when processing static cases. The change caused another minor mistake to happen, which was apparent in "Can You Feel The Love Tonight" and "Exercise No. 19."

Thursday October 17

Based on our conversation today, I had the idea to place fingering based on motif. In music, a motif is a small fragment of music that acts as a recurring theme throughout the piece.... Finding the motif in a piece with loops proved more difficult than expected. I needed to find a section of music that repeated, but without knowing what was in the motif or how long it might be, my code became inefficient. I then realized that regular expressions were a good solution. I found this regex ((.+?)\1+$) on stack overflow that could find repeats, but it didn't match exactly what I needed so I tweaked it to (̂.+).*\1+", which successfully matches the motif in any given piece.

Monday October 21

I continued to test my minor alterations method. Another error made on "Hedwig's Theme" resulted in a tweak on how the algorithm handles calculating hand position when processing static cases. The change caused another minor mistake to happen, which was apparent in "Can You Feel The Love Tonight" and "Exercise No. 19." I split the arrange hand method into two separate methods, one to arrange the hand given a static position and the other to calculate music sections. The former took in an extra argument of prev and made decisions for hand position based on that.

Tuesday October 22

I modified the algorithm to account for "jump cases," which means two notes that are consecutively fingered with a 1. Jump cuts are defined by a 7th or greater (a difference of 10 between local min and max in midi note values), and I could use a similar algorithm for the jump cases that I could use for fingering scales. The bottom note is labelled with a 1 and the top note is labelled with a 5. Fingering is then done from left to right, with each next fingering depending on the previous note.

Thursday October 24

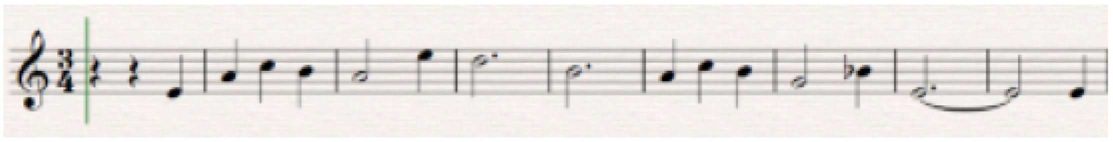
I modified the algorithm to account for "jump cases," which means two notes that are consecutively fingered with a 1. Jump cuts are defined by a 7th or greater (a difference of 10 between local min and max in midi note values), and I could use a similar algorithm for the jump cases that I could use for fingering scales. The bottom note is labelled with a 1 and the top note is labelled with a 5. Fingering is then done from left to right, with each next fingering depending on the previous note.

Timeline

Date	Goal	Met
September 27th	Write script that can deal with consecutive minor alterations and finger a C Major Scale	The algorithm can finger most consecutive minor alterations although some debugging is still necessary
October 4th	Write script that can deal with one major alteration (a crossover)	The algorithm can finger the major scales
October 11th	Be able to finger a piece that contains at least one crossover, one minor alteration, and one jump case	Algorithm can finger a piece containing multiple minor alterations and one jump case
October 18th	Be able to finger 6 major arpeggios	–
October 25th	Be able to finger all 12 major arpeggios	–

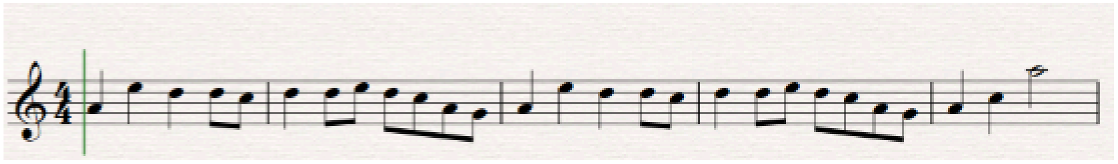
Reflection

This week, I fixed several problems with my minor alterations algorithm and wrote in code to deal with 1-1 jumps. Below is an excerpt from Hedwig's Theme, a greater challenge for the minor alterations algorithm than Believer because it has more true changes in hand position



Array: [[256, 64, 1], [256, 69, 3], [256, 72, 5]] [[256, 71, 2], [512, 69, 1], [256, 76, 5], [768, 74, 4], [768, 71, 2], [256, 69, 1]] [[256, 72, 4], [256, 71, 3], [512, 67, 1]] [[256, 70, 4], [1280, 64, 1], [256, 64, 1]]

Below is an excerpt from Believer, showing the successful 1-1 jump.



Array: [[256, 69, 1], [256, 76, 5], [256, 74, 4], [128, 74, 4], [128, 72, 3], [256, 74, 4], [128, 74, 4]] [[128, 76, 5], [128, 74, 4], [128, 72, 3], [128, 69, 2], [128, 67, 1]] [[256, 69, 2], [256, 76, 5], [256, 74, 4], [128, 74, 4], [128, 72, 3], [256, 74, 4], [128, 74, 4]] [[128, 76, 5], [128, 74, 4], [128, 72, 3], [128, 69, 2], **[128, 67, 1]] [[256, 69, 1]**, [256, 72, 2], [512, 81, 5]]