Journal Report 5
9/30/19-10/07/10
Irina Lee
Computer Systems Research Lab
Period 1, White

Daily Log

Monday September 30

After drawing out all 12 major scales, I discovered several helpful patterns for placing crossovers: a switch from black to white key means placing a 1 on the white key, 3-1 is the most common crossover because it offers better stability than 4-1 but also minimizes the number of crossovers compared to 2-1. Instead of fingering the crossover sections in order, I will implement a scan that places the 1s first, then fingers the rest of the section.

Tuesday October 01

I implemented the black-to-white-key scan and a fingering possibilities method. The black-to-white-key scan places 1s when a shift from a black to a white key occurs, and the fingering possibilities method takes the fingering from the previous note to generate all possible fingerings for the current note. I noticed that when going up the keyboard, lower fingers maximize mobility while upper fingers minimize mobility. Thus, by default, I have the algorithm choose the minimum from fingering out of all possibilities. I tested the algorithm on the D Flat, A Flat, E Flat, and B Flat scales and the fingering was successful. Interestingly, I anticipate that the C Major Scale will be the most difficult scale because it contains no black keys.

Thursday October 03

For the white key scales, a second helpful pattern I noticed is that the crossover in major scales usually happens on a half step. All major scales follow the pattern W-W-H-W-W-W-w (with W = Whole step and H = Half step), and the crossover almost always happens on the half step in the case of white key scales, with the exception being F Major (because its half step ends on a black key). With this encoded into my algorithm, I can now finger all 12 major scales. As of now, I'm not sure how applicable the half step crossover rule is to real pieces. The scale fingering algorithm also assumes that the notes are consecutive, meaning it might place incorrect fingering if faced with notes that are farther apart. I'm planning on testing and modifying the code on arpeggios to mitigate this problem.

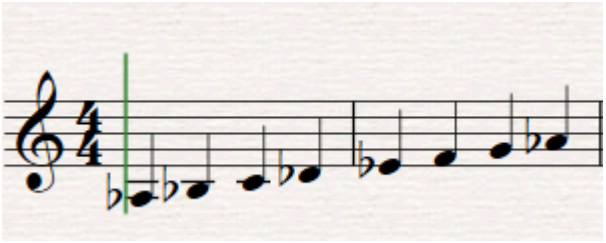
Timeline

Date	Goal	Met
September 19th	Write script that can adjust to minor alterations and finger a C Major Scale	The script can put in fingering for minor alterations
September 26th	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	–
October 18th	Be able to finger 6 major arpeggios	–

Reflection

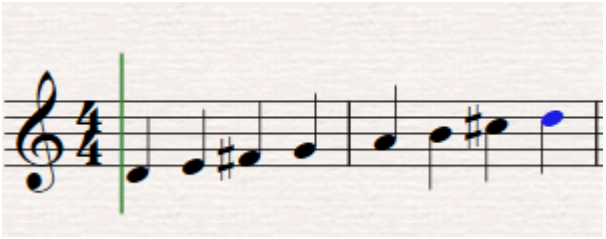
This week, I successfully fingered 12 major scales. Attached are two examples: the first from a white key scale and the second from a black key scale.

A Flat Major



Array: [[256, 56, 2], [256, 58, 3], [256, 60, 1], [256, 61, 2], [256, 63, 3], [256, 65, 1], [256, 67, 2], [256, 68, 3]]

D Major



Array: [[256, 62, 1], [256, 64, 2], [256, 66, 3], [256, 67, 1], [256, 69, 2], [256, 71, 3], [256, 73, 4], [256, 74, 5]]

Next week, I will expand the capabilities of my crossover algorithm into real pieces and look into implementing "jump" cases. Hopefully, I will be able to integrate my crossover, minor alterations, and jump case code into one algorithm that can work in tandem either this or next week.