Music Search Engine

After building a web search engine, we thought it would be interesting to build a search engine on our own too, but for music instead! We are keeping the function of this search engine simple since music can get really complicated. Including other search variables such as time and key signatures would be ideal, but we currently have limited inputs to music notes only for the sake of simplicity.

Main Idea

The purpose of this search engine is for users to be able to search for musical pieces without knowing the musical title. More often than not, some people have had short melodies in their heads and were unable to identify the music from which it originates. By inputting some notes from that melody into our search engine, the search engine will find song matches based on the input. This of course requires the user to have a little background in music, to be able to know what musical notes to search for. Again, for simplicity, our search engine only takes basic notes. The search engine will return the best possible results, ranked.

For example:

We use a database containing the main melodies of various songs.

Database consists of:

'Twinkle_Twinkle_Little_star', 'CCGGAAGFFEEDDCGGFFEEDGGFFEEDCCGGAAGFFEEDDC'
'Mary_Had_a_Little_Lamb', 'EDCDEEEDDDEGGEDCDEEEEDDEDC'
'Scarborough_Fair', 'DDAAAEFEDACDCABGADDCAAGFECDAGFEDCD'
'Green_Sleeves',
'EGABCBAF#DEF#GEED#EF#D#BEGABCBAF#DEF#GF#ED#C#D#EEDDC#BAF#DEF#GEED#EF#D#BDDC#BAF#DEF#GF#ED#C#D#EE'
'Star_Wars_Imperial_March_Simplified', 'EEECGECGEBBBCGEbCGE'

Now, a user inputs 'EDC'. The search will return an ordered result:

'Twinkle_Twinkle_Little_star', 'Mary_Had_a_Little_Lamb', 'Scarborough_Fair', 'Green_Sleeves', 'Star_Wars_Imperial_March_Simplified'

How does the Music Search Engine work

Ranking System:

Highest rank: Exact Note Match

We match the user input notes against the notes of the music database we have.

This means that if a user inputs '*ABC*', music titles with exact notes of '*ABC*' will be returned as exact note matches.

Second highest rank: Exact Note-Distance Match

We match the distance between the notes, of user input against music database.

By distance between notes, we mean the tone distance. For example, '*A*' and '*B*' have a distance of 1, 'C' and 'G' have a distance of 4.

The purpose of this note-distance function is to allow users, who do not know the key signature of the piece of music, to be able to still find the intended song.

Miscellaneous: Ranking based on Edit Distance, either by Note or Note-Distance Matches

We rank the rest of the database items, by determining the edit distance between the notes and note-distance. The fewer the edits are, the higher the music title is ranked.

The purpose of this edit_distance function is to allow users to have some error margins in getting the notes right.

Rank Value:

We set the highest rank to be -2, 2nd highest rank to be -1 and edit_distance ranks to be positive numbers, so that we can differentiate between the various values.

To view the ranks of each title, go to:

1) ranked_search(index)
2) Comment off "result.append(title[0])"
3) Remove comment on "result.append(title)"

The output of results with rank values has been set to default so as to ease the judges from having to search for the line of code.

Output:

Output looks like this:

With values:

[['Twinkle_Twinkle_Little_star', -2], ['Mary_Had_a_Little_Lamb', -2], ['Scarborough_Fair', -2], ['Green_Sleeves', -1], ['Star_Wars_Imperial_March_Simplified', -1]]

Without values:

['Twinkle_Twinkle_Little_star', 'Mary_Had_a_Little_Lamb', 'Scarborough_Fair', 'Green_Sleeves', 'Star_Wars_Imperial_March_Simplified']

Limitations

1) As the edit_distance code takes exponential amount of time to run based on the length of user input, we are only able to allow 4 or less search input. Longer inputs can be done without being seemingly trapped in infinity loop, by shortening the database notes.
2) To make this more practical, we need a way to retrieve musical notes based on mp3 instead of inputting them manually.