Query by Singing/Humming

Problem definition

Dynamic programming (DP) is a very important and useful method for sequence comparision. It has been used extensively in speech recognition, natural language processing, music analysis, and a variety of other areas. In this homework, we shall explore the use of DP for query by singing/humming (QBSH), which is an interesting paradigm of music information retrieval. (For simplicity, we shall assume our QBSH system always starts the comparison from the beginning of a song.)

In particular, you need to find the alignment between the pitch vector of human's singing and the note vector of a given melody, both in the unit of semitone or MIDI number. The correspondence between MIDI numbers and piano keyboard can be found here. Moreover,

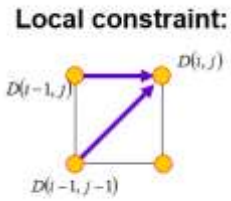
- Here is a typical singing clip and its corresponding pitch vector which can be played here.
- The note vector is the music notes corresponging to the intended song of the singing. For simplicity, there no duration information in the vector. Therefore for the song of "Twinkle twinkle little star", here is the music note vector for the first two phrases: [60 60 67 67 69 69 67 65 65 64 64 62 62 60].

Given a pitch vector and a note vector, your mission is to find an optimum mapping to assign each pitch point to a music note, such that the total distance is minimized. To be more specific, let's use the following notation:

- $p(i)p(i)$ is element ii of the pitch vector, where $0 \le i \le m-10 \le i \le m-1$.
- $q(j)q(j)$ is element jj of the note vector, where $0 \le j \le n-10 \le j \le n-1$.

The alignment path can be denoted by its coordinates, that is, $\{(0,u_0),(1,u_1),(2,u_2),...,(m-1,u_{m-1})\}\{(0,u0),(1,u1),(2,u2),...,(m-1,um-1)\}$, where $u_k,k=0,1,\cdots,m-1uk,k=0,1,\cdots,m-1$ are indices into the note vector with the following constraints:

- $u_0=0u0=0$.
- $u_0 \le u_1 \le u_2 \cdots \le u_{m-1}u0 \le u1 \le u2 \cdots \le um-1$.



For instance, the alignment path of the above figure is (0,0),(1,0),(2,1),(3,1),(4,1),(5,1),(6,2),(7,2),(8,3),(9,3),(10,3),(11,3)(0,0),(1,0),(2,1),(3,1),(4,1),(5,1),(6,2),(7,2),(8,3),(9,3),(10,3),(11,3), with $m=12m=12$ (the length of pp) and $u_{m-1}=3um−1=3$ (the last index of music note that have at least one pitch point being assigned to). In other words:

- p(0) and p(1) are assigned to q(0).
- p(2), p(3), p(4) and p(5) are assigned to q(1).
- p(6) and p(7) are assigned to q(2).
- p(8), p(9), p(10) and p(11) are assigned to q(3).

The distance between a pitch vector pp and note vector qq can then be defined as follows:

$$D(p,q)=\min_{u0,u1,...um-1}\sum i=0m-1|p(i)-q(u_i)|,D(p,q)=minu0,u1,...um-1\sum i=0m-1|p(i)-q(ui)|,$$
subject to$u_0=0u0=0$and$u_0\le u_1\le u_2\cdots\le u_{m-1}u0\le u1\le u2\cdots\le um-1$.

Suggested approach

Based on the above description, we can solve the task using DP. The 3-step formula for DP can be described as follows:

1. Optimum-value function $D(i,j)D(i,j)$ is defined as the minimum distance between $p(0:i)p(0:i)$ and $q(0:j)q(0:j)$.
2. Recurrent equation for $D(i,j)D(i,j)$ is shown next, with $i>0i>0$ and $j>0j>0$:

$$D(i,j)=|p(i)-q(j)|+\min\{D(i-1,j)D(i-1,j-1)D(i,j)=|p(i)-q(j)|+\min\{D(i-1,j)D(i-1,j-1)$$

(See the local constraint in the previous figure.) Please derive the recurrent equation when i=0i=0 or j=0j=0 by yourself. The boundary condition is D(0,0)=|p(0)−q(0)|D(0,0)=|p(0)−q(0)|.

3. Answer to the original task:

$$dist(p,q)=\min_{0\le j\le n-1} D(m-1,j).dist(p,q)=min0≤j≤n−1D(m−1,j).$$

Input/output formats

- Input format:
- m ← The number of elements of the pitch vector
- p0p0 p1p1 p2p2 . . . . . pm−1pm−1 ← the mm elements of the pitch vecotr
- n ← The number of elements of the note vector
- q0q0 q1q1 q2q2 . . . . . qn−1qn−1 ← the nn elements of the note vecotr
- Output format:
- distance ← The overall minimized distance
- s0s0 s1s1 s2s2 s3s3 s4s4 ... sum−1sum−1 ← Starting index of pitch vector being assigned to each music notes

  where

  - s0s0 to s1−1s1−1 are the indices of the pitch vector being assigned to note 0. (s0s0 is always 0.)
  - s1s1 to s2−1s2−1 are the indices of the pitch vector being assigned to note 1.
  - s2s2 to s3−1s3−1 are the indices of the pitch vector being assigned to note 2.
  - ...
  - sum−1sum−1 to m−1m−1 are the indices of the pitch vector being assigned to note um−1um−1, where um−1+1um−1+1 is the total number of music notes that have at least one pitch point being assigned to.

  Take the alignment path shown in the previous figure as an example, the second line of the output file should be "0 2 6 8",
  with um−1=3um−1=3.

Requirements & suggestions

- You can safely assume $m<1000$ and $n<100$. (This is irrelevant if you use dynamic memory allocation for creating the matrix.)
- If there are repeated notes, the optimal path might not be unique. As a result, TA's judge system will check your output based on the returned minimum distance. If the deviation between your distance and TA's distance is less than 1, it'll be considered as correct.
- Your best bet is to use DP for the homework. Of course, you can still use other approaches if they are as efficient as DP.
- The alignment path shown in the previous plot is based on X-Y mode, which is commonly used in the Cartesian coordinate system. If you want to adopt I-J mode (commonly used in matrix indexing), you can simply rotate the plot by 90-degree clockwise.
- My test using a pitch vector of 207 elements and a note vector of 14 elements takes only 0.000154 second on my i7-5700HQ notebook PC. As a result, a time limit of 0.01 second will be imposed on your program for each task of a similar size. Your score is zero if the time limit is exceeded.
- The process of converting a singing clip into its pitch vector is call pitch tracking, which is out of the scope of this class.
- If you are interested in trying out QBSH systems, try our lab's system at http://mirlab.org/demo/miracle.
- As usual...
    a. Your program should take the input from the standard input and send the output to the standard output.
    b. You need to write the program from scratch. You cannot use any open-source or readily available solvers.