Optimal bowing

Steven

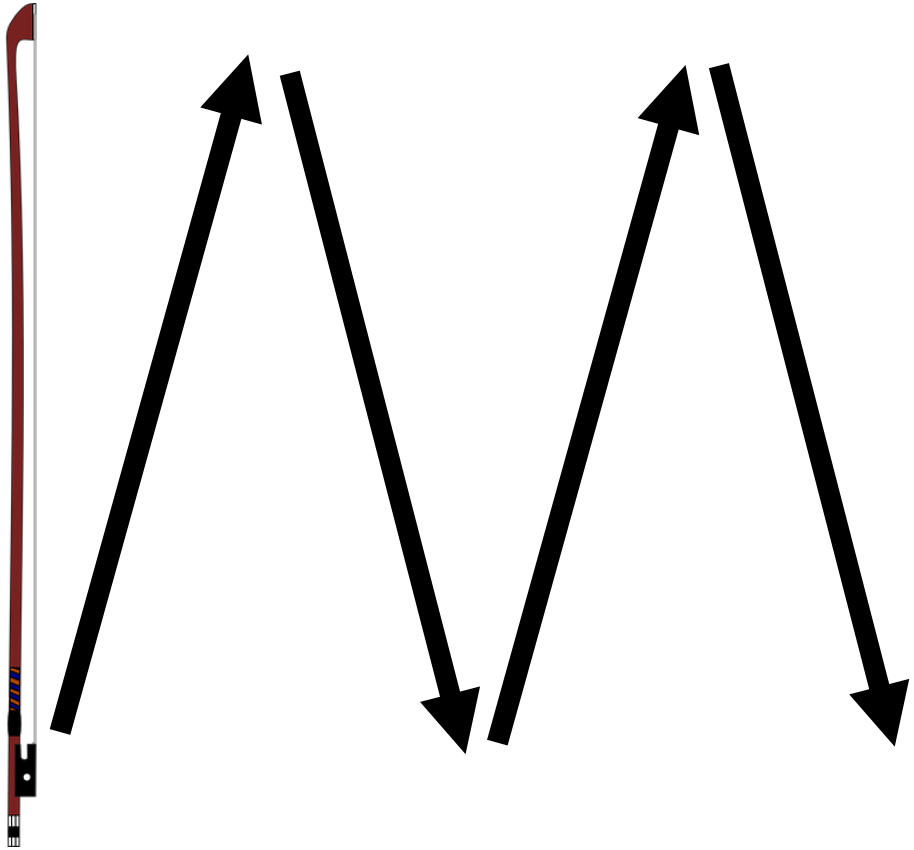
6 May 2017

Problem statement

- Given N music notes with duration $1/a_1$, $1/a_2$, …, $1/a_N$
- Play the music on a violin as loud as possible
- Output the optimal bowing (whether each note should be played up or down)

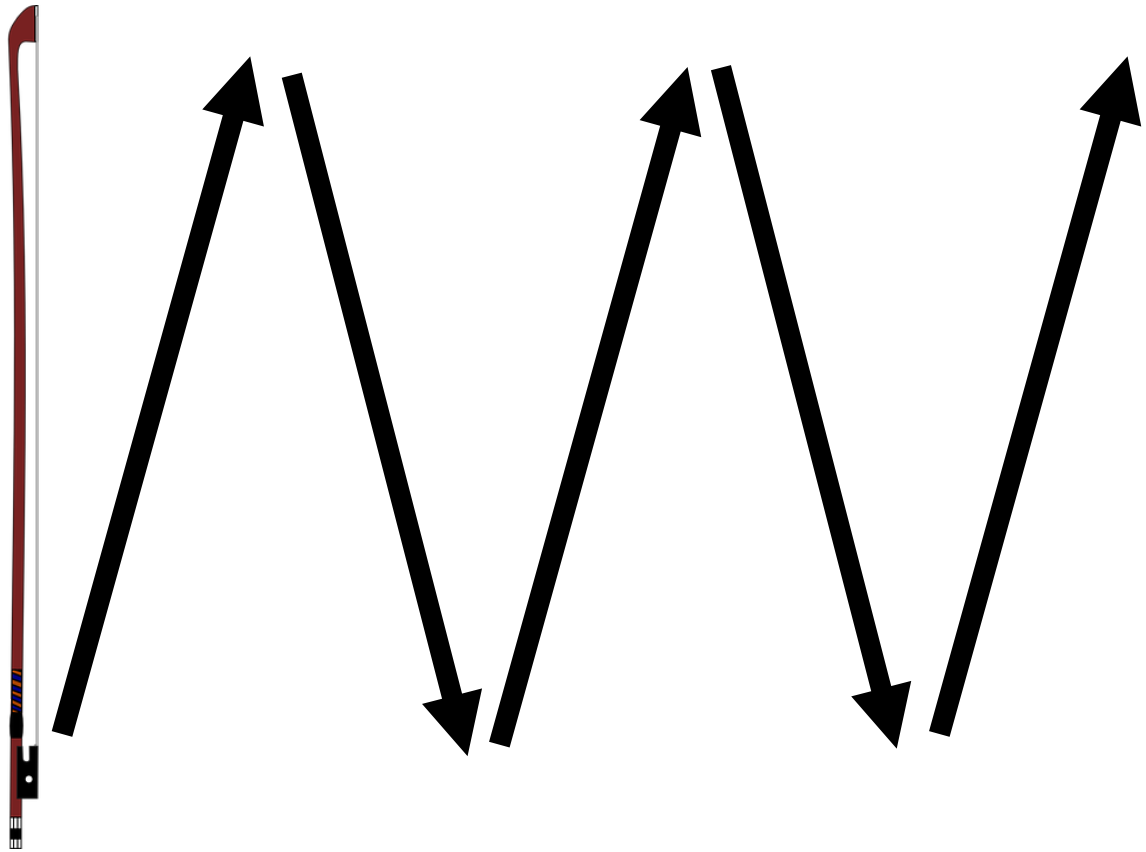
Sample

Input	Output
1 1 1 1	dudu



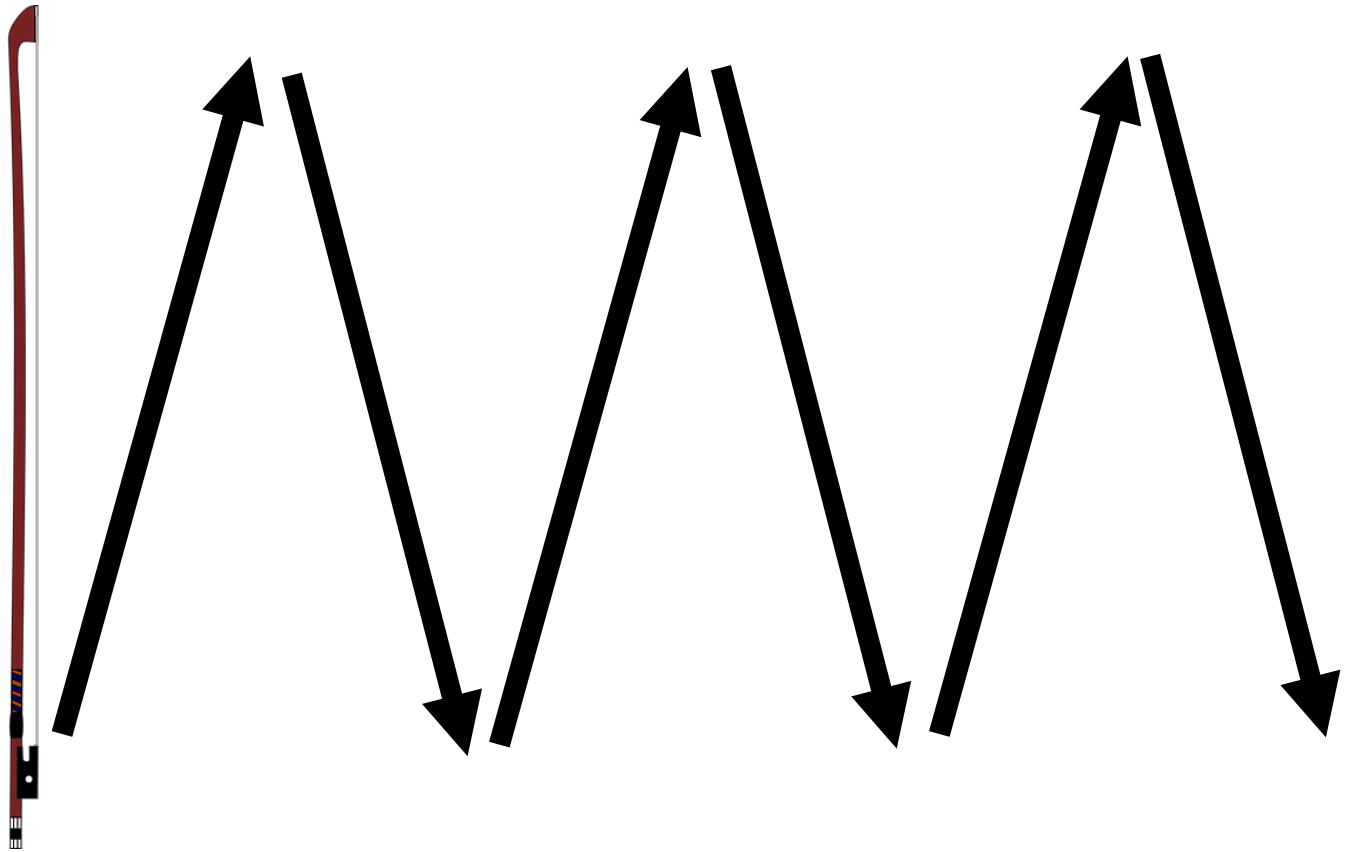
Sample

Input	Output
2 2 2 2 2	dudud



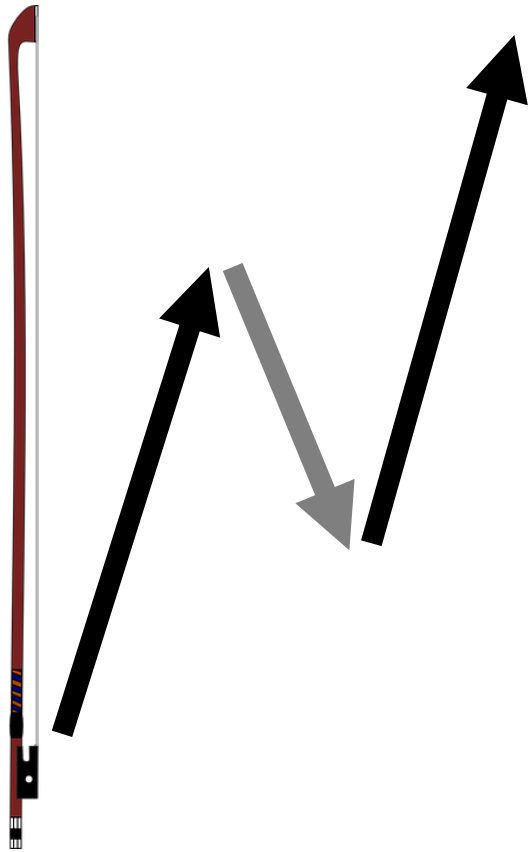
Sample

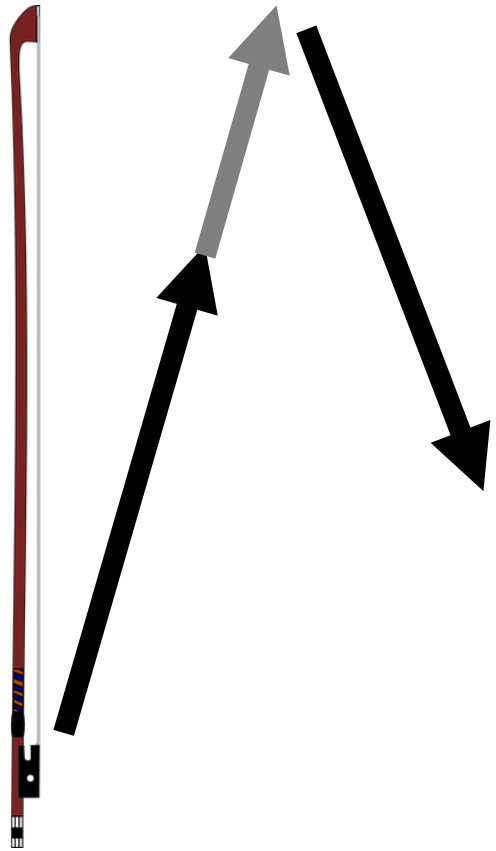
Input	Output
10 10 10 10 10 10	dududu



Sample

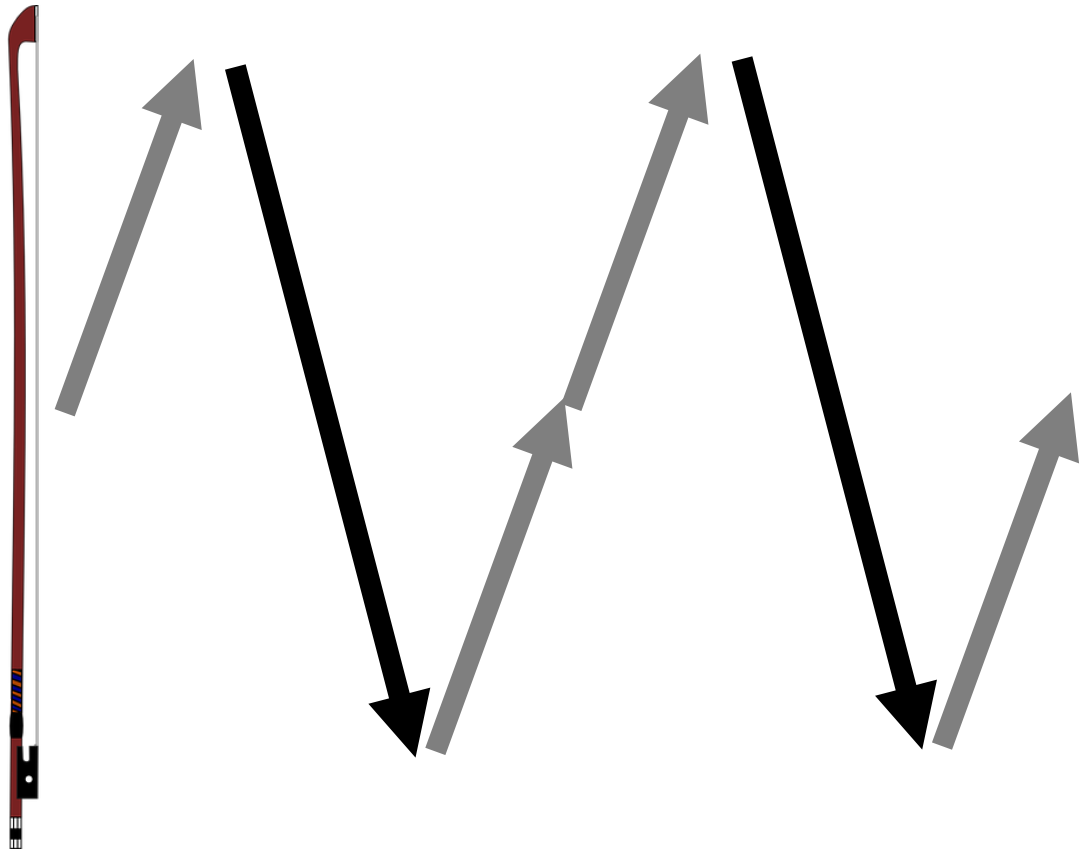
Input	Output
1 2 1	dud / ddu (lexicographical smallest)





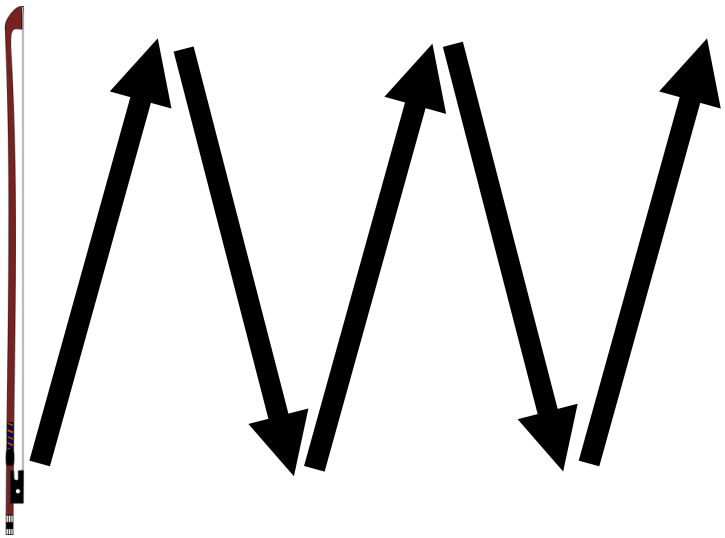
Sample

Input	Output
2 1 2 2 1 2	duddud

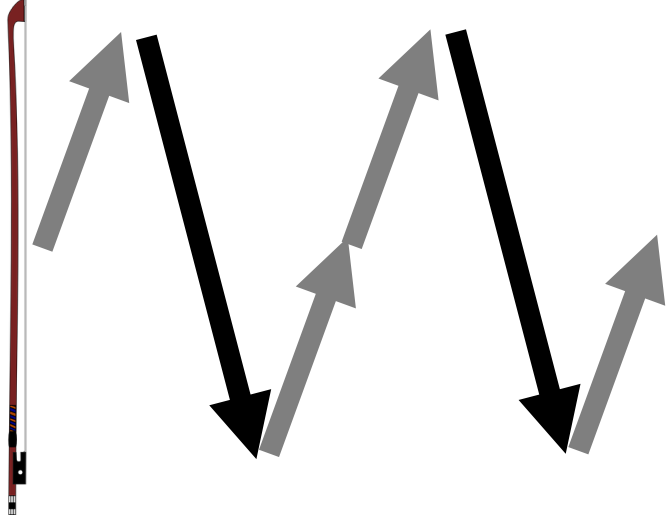


Subtask 1

- Each note is either whole note or half note
- Case 1: all whole notes
  - dududu...
- Case 2: all half notes
  - dududu...

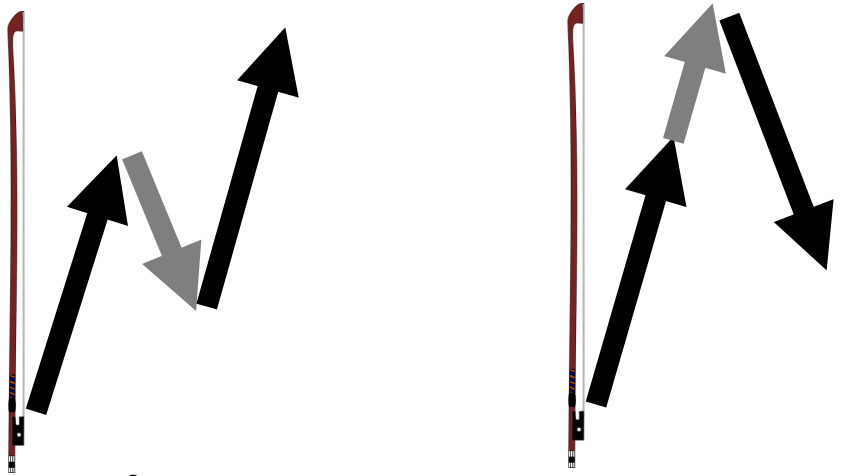


Subtask 1



- Case 3:
  play whole notes with whole bow
  - The number of consecutive half notes between any two whole notes must be even
  - Always make sure the bow is either at the tip or at the bottom when the next note to play is a whole note

Subtask 1



- Case 4:
  play whole notes with 2/3 bow
  - The number of consecutive half notes between two whole notes may be odd
  - Always make sure that bow is at the bottom, 1/3 position, 2/3 position or tip
  - Then, the next note can always be played
- Done

Subtask 2

- N <= 18
- Exhaustion
- Main idea:
  - For all the 2^N possible bowings, find the optimal one

Subtask 2
Transformation of problem

- 1/1, 1/2, 1/3, 1/10???
- Play loudest???
- Make it more computational friendly!

Subtask 2
Transformation of problem

- **Observe:**
  LCM(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) = 2520

$1 / a_i$	b_i
$1 / a_i$	$2520 / a_i$
1 / 1	2520
1 / 2	1260
1 / 3	840
…	…
1 / 10	252

Subtask 2
Transformation of problem

- Let c_i can be -1 or 1
  $d_0 = 0$
  $d_1 = c_1 b_1$
  $d_2 = c_1 b_1 + c_2 b_2$

  ...
  $d_N = c_1 b_1 + c_2 b_2 + \ldots + c_N b_N$
- Loudest bowing yields
  $\min(\max(d_i) - \min(d_i))$

Subtask 2

- $N <= 18$
- Exhaust all possible c_i and find $\min(\max(d_i) - \min(d_i))$
- $O(N * 2^N)$

Subtask 3

- Observation:
  - We can always play the whole note with at least 1/2 bow
- $\min(\max(d_i) - \min(d_i)) < 5040$

Subtask 3

- Let's fix $\max(d_i) - \min(d_i) = L$, see if we can still play all notes.
- The length of the bow is L units; playing a note takes b_i unit of bow

Subtask 3

- dp[i][j] =
  - 1 if the bow can be at position j after playing the i-th note
  - 0 otherwise
- dp[0][j] = 1 for 0 <= j <= L
  - The bow can be at any position initially
- dp[i][j] = dp[i – 1][j – b_i] or dp[i – 1][j + b_i] for 0 <= j, j – b_i, j + b_i <= L

Subtask 3

- Try L = 1, L = 2, L = 3, ... until L can play all notes
- That yields $\min(\max(d_i) - \min(d_i))$
- For each L, run the whole DP; check if dp[N][j] = 1 for some 0 <= j <= L
- $O(N * LCM(a_i)^2)$

Subtask 4

- $0 < L < 2 * LCM(a_i)$
- Binary search on L
- $O(N * LCM(a_i) * \log(LCM(a_i))$

Subtask 4 (Optional)

- $0 < L <= 1.5 * LCM(a_i)$
  - The proof is left as an exercise
- Implement with C++ bitset
  - much faster

Lexicographically smallest

- DP on the reversed music instead, i.e. play b_N, b_{N-1}, …, b_1
- For each dp[N][j] trace to dp[N – 1][j – b_N] or dp[N – 1][j + b_N], use down bow whenever possible
- Output the smallest one
- O(N^2)

Lexicographically smallest (Alternative)

- DP on the reversed music instead, i.e. play b_N, b_{N-1}, …, b_1
- Put j into queue where dp[N][j] = 1
- BFS backward, each path yields a bowing
- Discard non-smallest bowings along the way
- O(N * LCM(a_i))