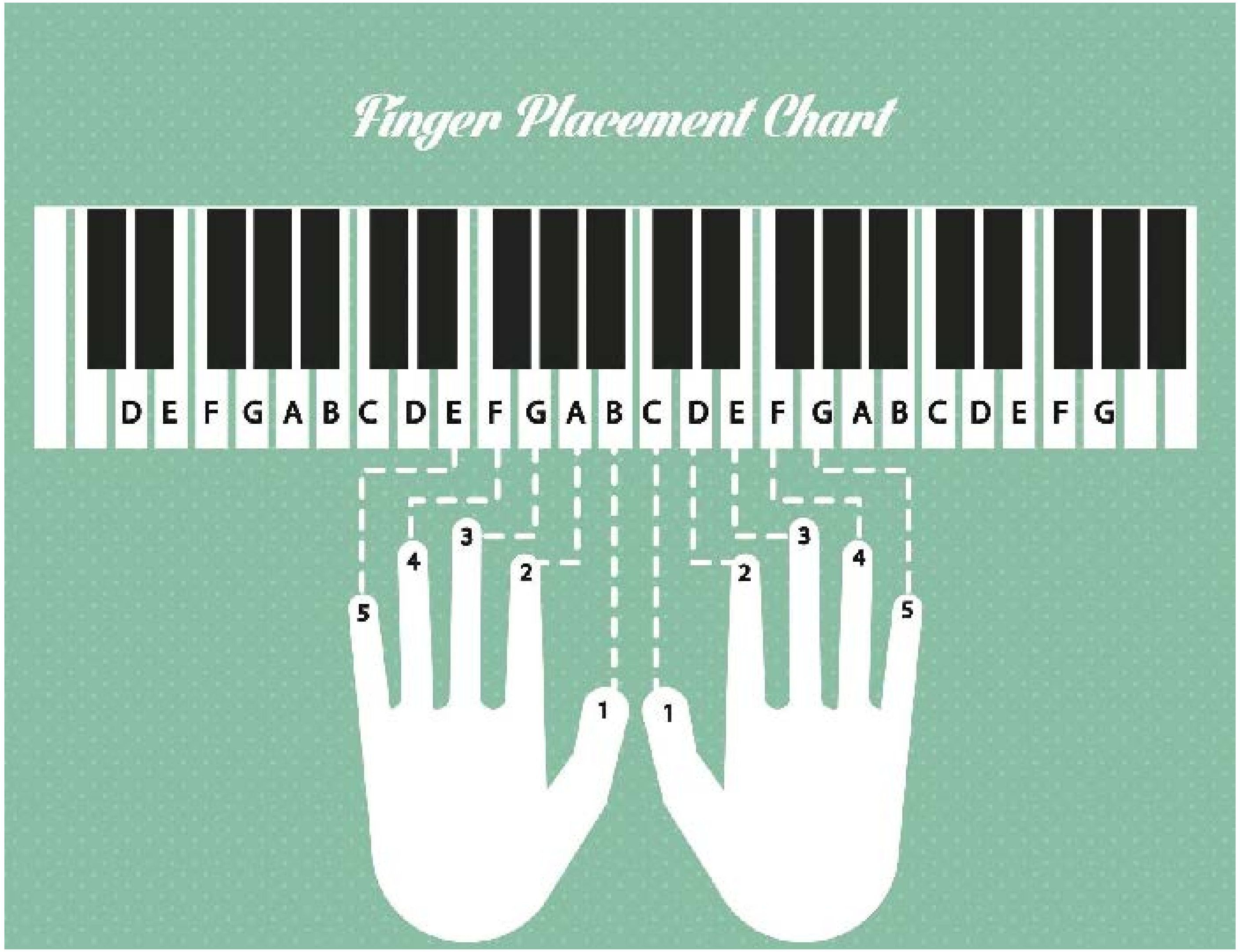
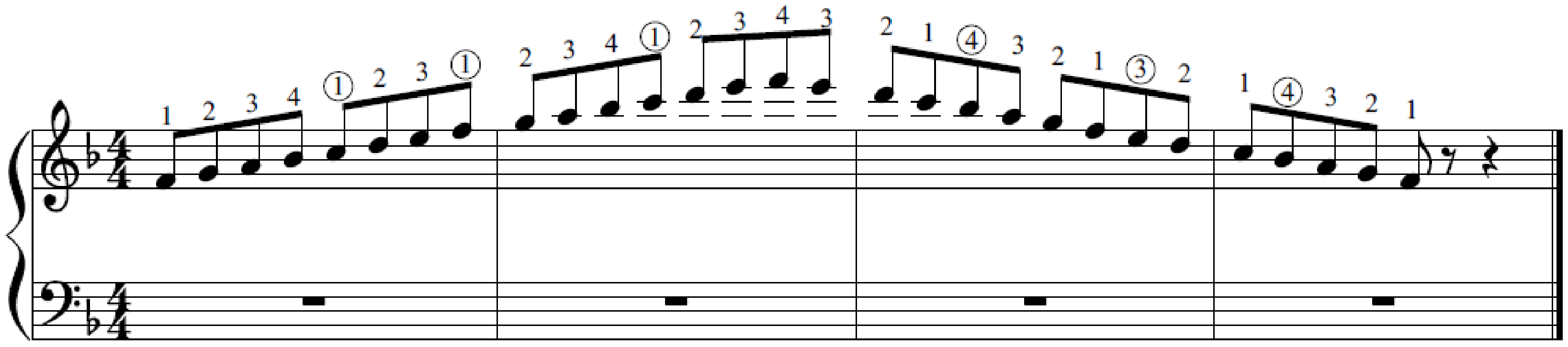
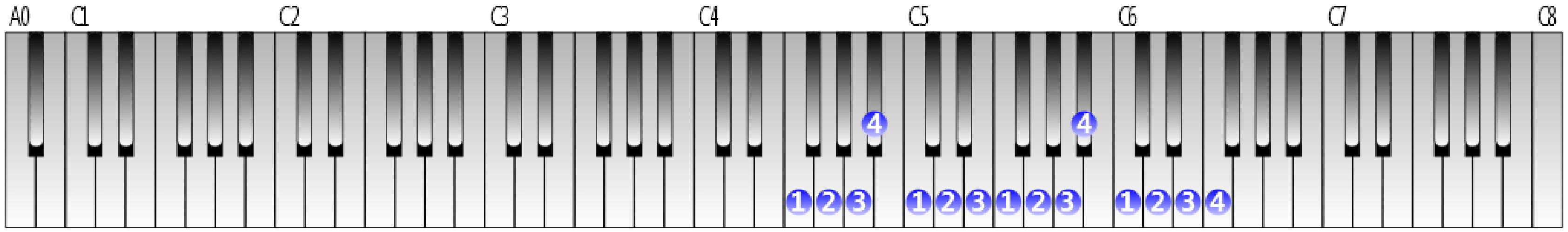
Practical Dynamic Programming: Game

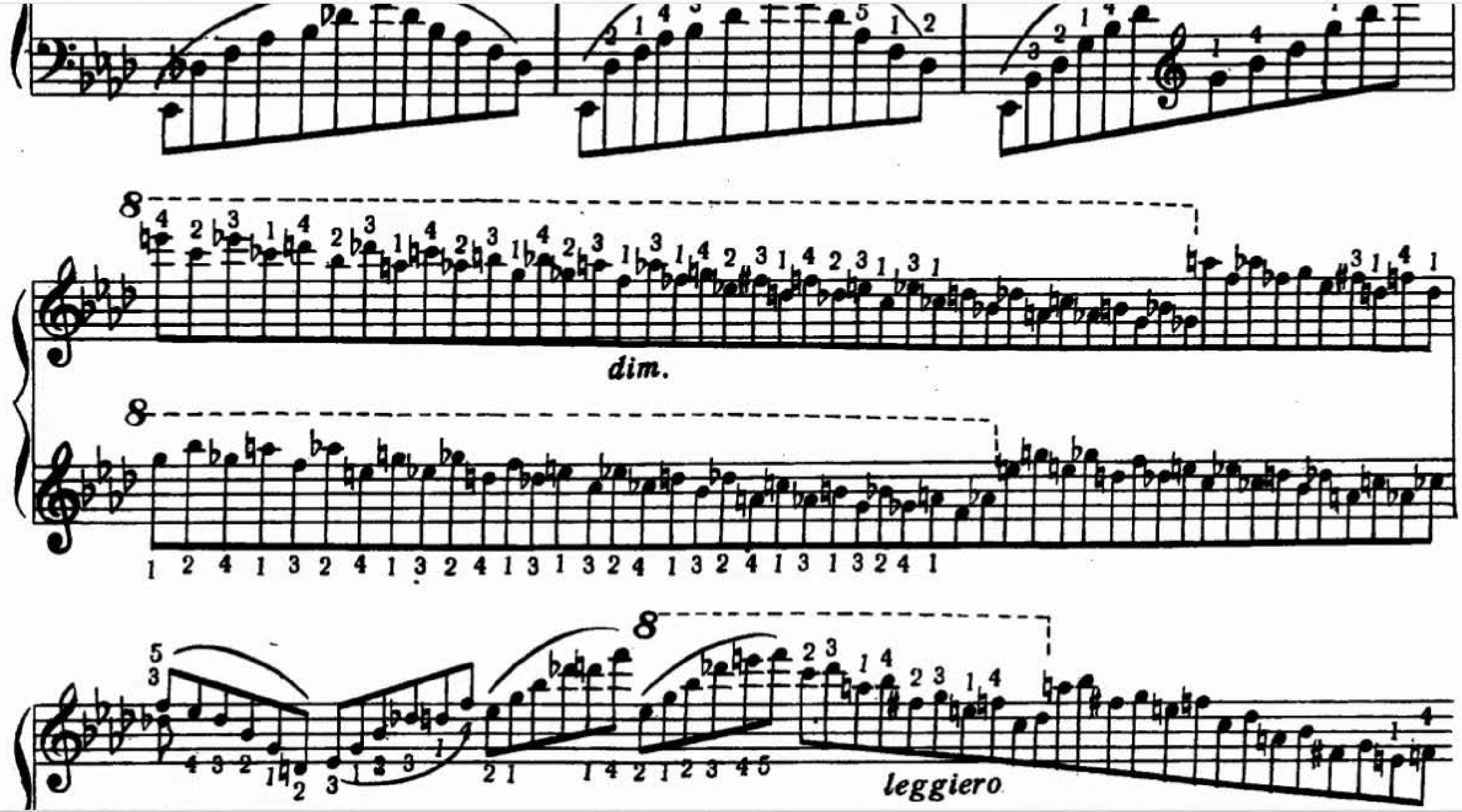
Playing Piano



F Major scale 2 octaves (right hand)






dim.
leggiero

Implements a dynamic programming algorithm

1. Clarify your goal (problem)
2. Divide the final goal to subproblems

2-1. Find a recursive method to derive the final result from previous results and additional search

2-2. Check your method guarantees the results of searching all possible solutions

2-3. Do not need to search all possible solutions. We may prune unnecessary search.

3. Find initial states

* Store the results of subproblems

Problem?

- Given input
  - P: a guitar score (many notes for each time step)
  - H: a set of finger ID
  - C: a cost function of p1, p2, f1, f2 where p1, p2 in P

    F1 and F2 are a subset of F
- Output
  - F: a sequence of f in H (size = |P|)
- Goal?
  - $argmax_F \sum_{i=0}^{n-2} C(f_i, f_{i+1}, p_i, p_{i+1})$

Subproblem?

- Goal
  - $argmax_F \sum_{i=0}^{n-2} C(f_i, f_{i+1}, p_i, p_{i+1})$
  - $=$
    $argmax_F \sum_{i=0}^{n-3} C(f_i, f_{i+1}, p_i, p_{i+1}) \ + cost\ of\ all\ possible\ cases\ for\ p_{n-2}\ and\ p_{n-1}$
  - $=argmax_F\ Cost_{1,n-2} + Cost_{n-2,n-1}$
- Subproblem?
  - Finding a subsequence of F minimizing the cost
  - Repeat the split

Subproblem?

- Goal
  - $argmax_F \, Cost_{1,n-2} + Cost_{n-2,n-1}$
- Can we split the goal as
  - $=argmax_{F_{1,n-2}} \, Cost_{1,n-2} + argmax_{F_{n-2,n-1}} Cost_{n-2,n-1}$
- $F_{n-2,n-1}$: a finger selection
- $F_{1,n-2}$:a sequence for p_1 to p_{n-2}
- After we minimize the cost for $F_{n-2,n-1}$, finding the minimal cost of $F_{n-2,n-1}$ guarantees the optimal?

Subproblem?

- NO!
  - $Cost_{n-2,n-1}$ = C($f_{n-2}, f_{n-1}, p_{n-2}, p_{n-1}$)
  - f_{n-2} is in $F_{1,n-2}$
  - $Cost_{n-2,n-1}$ is dependent to the result of $Cost_{1,n-2}$
  - Selecting $F_{1,n-2}$ without the consideration of $Cost_{n-2,n-1}$ may not find the optimal

Subproblem?

- Goal
  - $argmax_F \; Cost_{1,n-2} + Cost_{n-2,n-1}$
    $\neq \; argmax_{F_{1,n-2}} \; Cost_{1,n-2} + argmax_{F_{n-2,n-1}} Cost_{n-2,n-1}$

- If the terms are dependent?
  - Make them independent
    - may be impossible, conditional independence is required

  - Check all conditions

Subproblem?

- Goal
  - If f_{n-2} is given (fixed),
  - $argmax_F\ Cost_{1,n-2} + Cost_{n-2,n-1} =$
    $argmax_{F_{1,n-3}}\ Cost_{1,n-2}\ +\ argmax\ _{f_{n-1}} Cost_{n-2,n-1}$

  - We do not know what is correct f_{n-2}
  - Check for all f_{n-2},

$argmax_F\ Cost_{1,n-2} + Cost_{n-2,n-1} =$
$argmax_{f_{n-2}}\ Cost(argmax_{F_{1,n-3}}\ Cost_{1,n-2}\ +\ argmax\ _{f_{n-1}} Cost_{n-2,n-1})$

Store in memory

Playing Guitar

Problem?

- Given input
  - P: a guitar score (many notes for each time step)
  - I: a set of finger ID (0~5)
  - H: a combination of elements in I
  - C: a cost function of p1, p2, f1, f2 where p1, p2 in P

    F1 and F2 are a subset of F
- Output
  - F: a sequence of f in H (size = |P|)
- Goal?
  - $argmax_F \sum_{i=0}^{n-2} C(f_i, f_{i+1}, p_i, p_{i+1})$

Subproblem?

- Goal
  - $argmax_F \; Cost_{1,n-2} + Cost_{n-2,n-1}$
- Subproblems

$$argmax_F \; Cost_{1,n-2} + Cost_{n-2,n-1} =$$
$$argmax_{f_{n-2}} \; Cost(argmax_{F_{1,n-3}} \; Cost_{1,n-2} \; + \; argmax_{\,f_{n-1}} Cost_{n-2,n-1})$$

- The number of f_{n-2} to check?
  - O(I^2) (if fingers in a combination is 2)
- Time complexity
  - O(I^2|P|)

Tetris

- https://youtu.be/jH2HGTkAhWo

Problem?

- Given input
  - B: a set of blocks
  - P: a sequence of blocks in B
  - M: a set of block maps
  - L: a set of positions to put a block

  - C: a cost function of current block arrangement in M, an input block in B, and a target position in L
- Output
  - F: a sequence of selected positions f in B (size = |P|)
- Goal?
  - $argmax_F \sum_{i=0}^{n-1} C(m_i, b_i, l_i\ )$

Problem?

- Subproblem
  - If a block map is determined, the cost is simply obtained by maximizing $C(m_i, b_i, l_i)$ for all locations.
  - Does not guarantee to see previous final.
  - We need to check again all possible conditions.
  - Conditions : m_i
  - For all m_i ?
    - Impossible to calculate all combinations $L_{0,i-1}$
  - The next cost is determined by the surface of m_i (maybe with the depth as the longest block)
  - conditions : S(m_i)

$$argmax_l \, Cost_{1,n-2} + Cost_{n-2,n-1} =$$
$$argmax_{S(m_{n-1})} \, Cost(argmax_{S(m_{n-2})} \, Cost_{1,n-2} + argmax_{\,l_{n-1}} Cost_{n-2,n-1})$$

Supermario

- https://youtu.be/ZzV8NBHu2TY

Problem?

- Given input
  - δ: transition function of objects
  - M: a set of object maps
  - A: a set of actions

  - C: a cost function of current map in M, an input action in A
- Output
  - F: a sequence of acitions in A (size = ?)
- Goal?
  - $argmax_F \sum_{i=0}^{n-1} C(m_i, a_i)$

Problem?

- Subproblem

  $S(m_i)$: the object arrangements to affect the cost evaluation of a current action

  $$argmax_l\, Cost_{1,n-2} + Cost_{n-2,n-1} =$$
  $$argmax_{S(m_{n-1})}\, Cost(argmax_{S(m_{n-2})}\, Cost_{1,n-2} + argmax_{\,l_{n-1}} Cost_{n-2,n-1})$$

  How to evaluate $S(m_i)$??
  - Complex in this problem, but possible
  - $S(m_i)$ may not be deterministic (by nondeterministic transition functions)
    - In this case, current decision is conditioned by the next transition results.
    - We need to check all possible transition conditions

  - Too many future conditions?
    - Estimate model is required