Discrete Optimization

MA2827

Fondements de l'optimisation discrète

Dynamic programming (Part 2)

https://project.inria.fr/2015ma2827/

Material based on the lectures of Erik Demaine at MIT and Pascal Van Hentenryck at Coursera

Outline

- Dynamic programming
  - Guitar fingering
- More dynamic programming
  - Tetris
  - Blackjack
- Quiz: bracket sequences

Dynamic programming

- DP ≈ “careful brute force”
- DP ≈ recursion + memoization + guessing
- Divide the problem into subproblems that are connected to the original problem
- Graph of subproblems has to be acyclic (DAG)
- Time = #subproblems · time/subproblem

5 easy steps of DP

Analysis:

1. Define subproblems — #subproblems
2. Guess part of solution — #choices
3. Relate subproblems (recursion) — time/subproblem
4. Recurse + memoize — time
   OR build DP table bottom-up
   - check subprobs be acyclic / topological order
5. Solve original problem — extra time

Guitar fingering

Task: find the best way to play a melody

Guitar fingering

Task: find the best way to play a melody

- Input: sequence of notes to play with right hand
- One note at a time!
- Which finger to use? 1, 2, …, F = 5 for humans
- Measure d(f, p, g, q) of difficulty to go
  from note p with finger f
  to note q with finger g

Examples of rules:

crossing fingers: $1 < f < g$ and $p > q$ => uncomfortable
stretching: $p \ll q$ => uncomfortable
legato (smooth): ∞ if $f = g$

Guitar fingering

Task: find the best way to play a melody

Goal: minimize overall difficulty

Subproblems:

min. difficulty for suffix note[i :]

#subproblems = O(n) where n = #notes

Guesses:

finger f for the first note[i]

#choices = F

Recurrence:

DP[i] = min{ DP[i + 1] + d(note[i], f, note[i +1], next finger) }

Guitar fingering

Task: find the best way to play a melody

Goal: minimize overall difficulty

Subproblems:

min. difficulty for suffix note[i :]

#subproblems = O(n) where n = #notes

Guesses:

finger f for the first note[i]

#choices = F

Recurrence:

DP[i] = min{ DP[i + 1] + d(note[i], f, note[i +1], next finger) }

Not enough information!

Guitar fingering

Task: find the best way to play a melody

Goal: minimize overall difficulty

Subproblems:

min. difficulty for suffix note[i :] when finger f is on note[i]

#subproblems = O(n F)

Guesses:

finger f for the next note, note[i + 1]

#choices = F

Recurrence:

DP[i, f] = min{ DP[i + 1, g] + d(note[i], f, note[i +1], g) | all g }

Base-case: DP[n, f] = 0

time/subproblem = O(F)

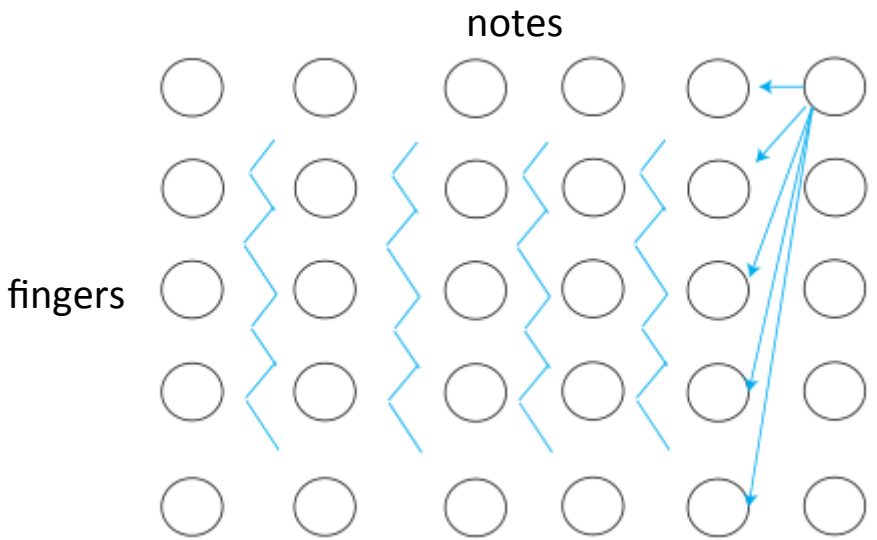
Guitar fingering

Task: find the best way to play a melody

Topological order:

for i = n-1, n-2, …, 0:

for f = 1, …, F:



total time = $O(n F^2)$

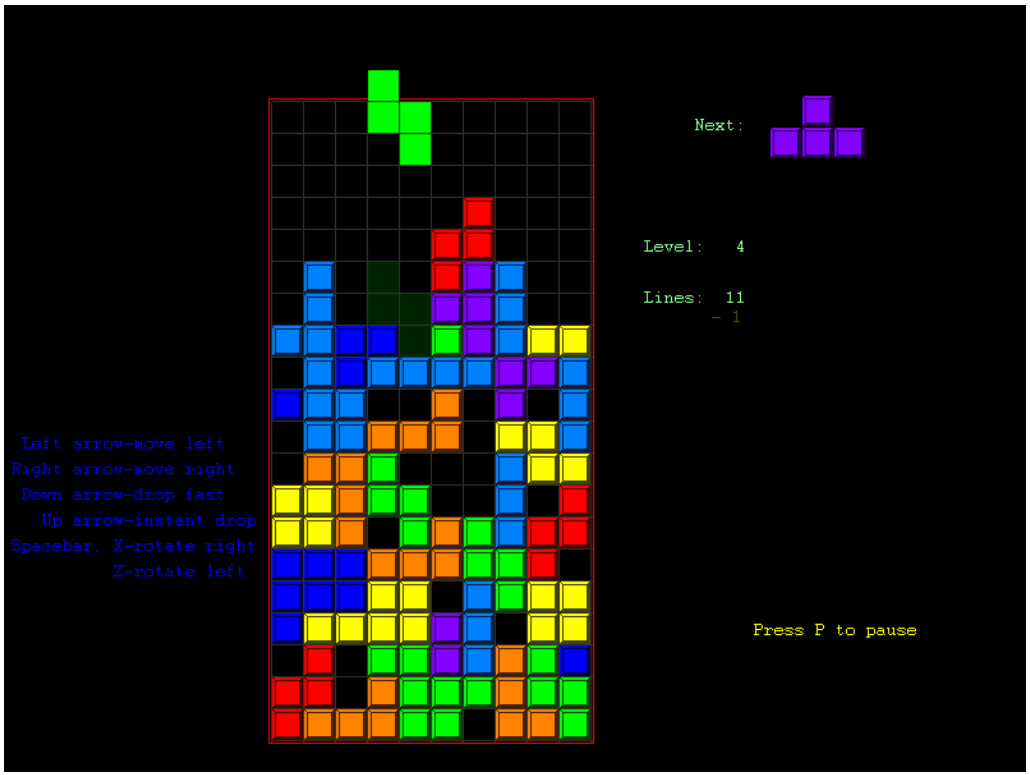
Final problem:

find minimal DP[0, f] for f = 1, …, F

guessing the first finger

Tetris

Task: win in the game of Tetris!



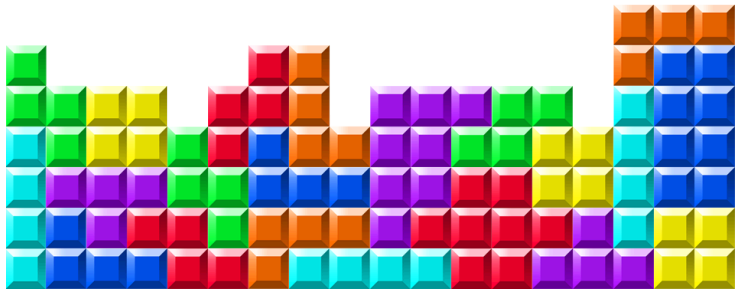
Tetris

Task: win in the game of Tetris!

- Input: a sequence of n Tetris pieces and
  an empty board of small width w
- Choose orientation and position for each piece
- Must drop piece till it hits something
- Full rows do not clear
- Goal: survive i.e., stay within height h

Tetris

Task: stay within height h



Subproblem:
survival? in suffix [i :]
given a particular column profile
#subproblems = $O(n (h+1)^w)$

Guesses:
where to drop piece i?
#choices = O(w)

Recurrence:
DP[i, p] = max { DP[i + 1, q] | q is a valid move from p }
Base-case: DP[n+1, p] = true for all profiles p
time/subproblem = O(w)

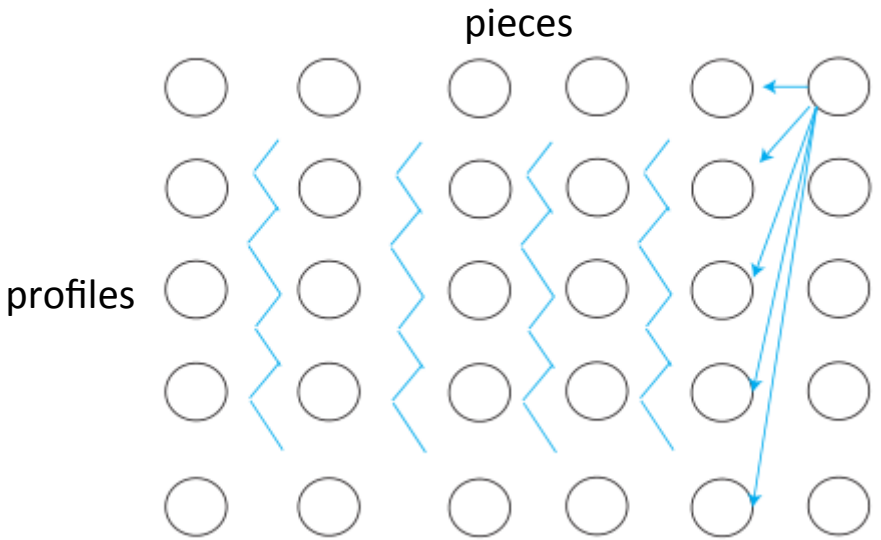
Tetris

Task: stay within height h

Topological order:

for i = n – 1, n – 2, …, 0:

for p = 0, …, $(h+1)^w - 1$:



total time $O(n w (h+1)^w)$

Final problem:

DP[0, empty]

Blackjack

Task: beat the blackjack (twenty-one)!

Blackjack

Task: beat the blackjack!

Rules of Blackjack (simplified):

- The player and the dealer are initially given 2 cards each
- Each card gives points:
  - Cards 2-10 are valued at the face value of the card
  - Face cards (King, Queen, Jack) are valued at 10
  - The Ace card can be valued either at 11 or 1
- The goal of the player is to get more points than the dealer, but less than 21, if more than 21 than he looses (busts)
- Player can take any number of cards (hits)
- After that the dealer hits deterministically: until ≥ 17 points

Perfect-information Blackjack

Task: beat the blackjack with a marked deck!

- Input: a deck of cards $c_0, \ldots, c_{n-1}$
- Player vs. dealer one-on-one
- Goal: maximize winning for a fixed bet $1
- Might benefit from loosing to get a better deck

Quiz (as homework)

- Write the DP for perfect-information blackjack
- Derive the number of subproblems for the tetris problem