Lecture Overview

- Piano Fingering
- Structural DP (trees)
- Vertex Cover & Dominating Set
- Beyond: treewidth, planar graphs, folding

Readings

CLRS 15

Review:

5 easy steps for DP

1. subproblems (define & count)
2. guessing (what & count)
3. relation (the true test)
4. DP (put pieces together)
5. original problem

*2 kinds of guessing:

A. in 3, guess which other subproblems to use (used by every DP except Fibonacci)

B. in 1, create more subproblems to guess more structure of solution (used by knapsack DP)

- effectively report many solutions to subproblems.
- lets parent subproblem know features of solution.
Piano fingering:

[Parncutt, Sloboda, Clarke, Rackallio, Desain, 1997]
[Hart, Bosch, Tsai 2000]
[Al Kasimi, Nichols, Raphael 2007] etc.

- given musical piece to play, say sequence of (single) notes with right hand
- metric \( d(f, p, g, q) \) of difficulty going from note \( p \) with finger \( f \) to note \( q \) with finger \( g \)

  e.g., \( 1 < f < g \) \& \( p > q \) \implies \text{uncomfortable}
  stretch rule: \( p \ll q \) \implies \text{uncomfortable}
  legato (smooth) \implies \infty \text{ if } f = g
  weak-finger rule: prefer to avoid \( g \in \{4, 5\} \)
  \( 3 \rightarrow 4 \) \& \( 4 \rightarrow 3 \) annoying \( \sim \) etc.

First Attempt:

1. subproblem = min difficulty for suffix notes[\( i : \)]
2. guessing = finger \( f \) for first note[\( i \)]
3. \( DP[i] = \min(DP[i + 1] + d(\text{note}[i], f, \text{note}[i + 1], ?) \text{ for } f \cdots) \)
   \( \rightarrow \) not enough information

1. subproblem = min difficulty for suffix notes[\( i : \)] given finger \( f \) on first note[\( i \)]
2. guessing = finger \( g \) for next note[\( i + 1 \)]
3. \( DP[\text{inf}] = \min(DP[i + 1, g] + d(\text{note}[i], f, \text{note}[i + 1], g) \text{ for } g \in \text{range}(F)) \)
   \( \leftarrow \# \text{ fingers} = 5 \text{ for humans} \)
   \( DP[n, f] = \phi \)
4. \( F n \) subproblems, \( F \) choices per subproblem \( \implies O(F^2n) \) time
5. \( \min(DP[\phi, f] \text{ for } f \in \text{range}(F)) \)
Dynamic Programming IV

Structural DP:
Follow combinatorial structure other than a (few)sequence(s) (by analogy to structural vs. regular induction)
* for DP on trees, useful subproblem is subtree rooted at vertex v, for all v

![Figure 1: DP on Trees](image)

Vertex Cover:
Find minimum set of vertices (cover) such that every edge is covered on \( \geq 1 \) end

- NP-complete in general graphs
- polynomial for trees:
  1. subproblem = min. cover for subtree rooted at v
     \( \Rightarrow \) n subproblems
  2. guessing = is v in cover?

![Figure 2: Vertex Cover](image)
Dynamic Programming IV

- \( \implies \) 2 choices
- YES \( \implies \) cover children edges
  \( \implies \) left with children subtrees
- NO \( \implies \) all children must be in cover
  \( \implies \) left with grandchildren subtrees

3. \( \text{DP}[v] = \min(1 + \sum \text{DP}[c] \text{ for } c \text{ in children}[v]) \quad \text{YES} \)
   \( \quad \text{len(children)} + \sum \text{DP}[g] \text{ for } g \text{ in grandchildren}(v)) \quad \text{NO} \)

4. time = \( O(n) \)

5. \( \text{DP[root]} \)

**Dominating set:**

Find minimum set of vertices such that every vertex is in or adjacent to set
- again NP-complete in graphs, polynomial on trees.

[material below covered in recitation]

1. subproblem = min. dom. for subtree rooted at \( v \)

2. guessing = is \( v \) in dom. set?
   - YES \( \implies \) dominate children
   - NO \( \implies \) must put some child in dom. set
     \( \implies \) dominate that child’s children

3. \( \text{DP}[v] = \min(1 + \sum (\text{DP}'[c] \text{ for } c \text{ in children}[v]) \quad \text{YES} \)
   \( \quad \text{but } c \text{ is already dominated } \cdots \text{ diff. subprob} \)
   \( 1 + \sum \text{DP}(c) \text{ for } c \neq d \text{ in children}[v]) \quad \text{NO} \)
   \( + \sum (\text{DP}'[g] \text{ for } g \text{ in children}[d])) \quad \text{NO} \)
   \( \quad \text{again already dominated } \sim \text{ different subprob} \)
   \( \quad - \text{guessing of the second type (B)} \)
   \( \quad \text{for } d \text{ in children}[c] \quad \leftarrow \text{guess child } e \text{ set A} \)

1’. subproblem ’ = min. dom. for subtree rooted at \( v \) given that \( v \) dominated already
(by parent subproblem)
\( \implies 2n \) subproblems total

3’. \( \text{DP}'[v] = \min(1 + \sum \text{DP}'[c] \text{ for } c \text{ in children}[v]), \quad \text{YES} \)
(\sum \text{DP}[c] \text{ for } c \text{ in children}[v]) \quad \text{NO} \)

4. time = \( O(\sum \text{deg}(v)) = O(E) = O(n) \)

5. \( \text{DP[root]} \)
Beyond:

Treewidth:

Many graphs are “thick trees” with reasonable “thickness” (\( \sim 7 \) e.g.).

- Most problems that are NP-complete in general can be solved in such graphs via DP

Planar Graphs:

Graphs often noncrossing in plane

![Planar Graphs](image)

**Figure 3: Planar Graphs**

- divide planar graph into BFS levels: see Figure 3
- throw away every kth level (e.g., \( k = 3 \)) starting from levels \( \phi, 1, \cdots, k - 1 \) (guess)
- in all cases, remaining graph is a “thick tree” of thickness \( O(k) \)
  \( \implies \) can solve this subproblem in poly-time
- can combine these solutions to solve original problem not optimally, but within \( 1 + 1/k \) factor of optimal \( \forall \) constants \( k \)

**Folding polygons into polyhedra:**

[Metamorphosis of the Cube video]

- DP on substrings of cyclic sequence (polygon)