6.006 Introduction to Algorithms
Spring 2008

Lecture 22: Dynamic Programming IV: Piano Fingering, Structural DP (Trees), Vertex Cover, Dominating Set, Beyond

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, Raekallio, 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., $1 < f < g$ & $p > q \implies$ uncomfortable
  stretch rule: $p \ll q \implies$ uncomfortable
  legato (smooth) $\implies \infty$ 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($DP[i+1] + d(\text{note}[i], f, \text{note}[i+1], ?)$ 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[inf] = $\min(DP[i+1, g] + d(\text{note}[i], f, \text{note}[i+1], g) \text{ for } g \in \text{range}(F))$
   $\leftarrow \sharp$ fingers = 5 for humans
   DP$[n, f] = \phi$
4. Fn subproblems, F choices per subproblem $\implies O(F^2 n)$ time
5. $\min(\text{DP}[\phi, f] \text{ for} f \text{ in range}(F))$

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

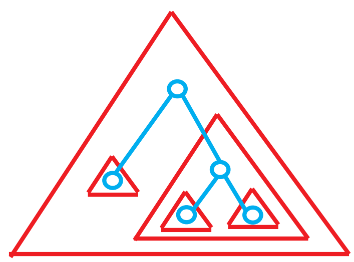

Figure 1: DP on Trees

Vertex Cover:

Find minimum set of vertices (cover) such that every edge is covered on ≥ 1 end

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

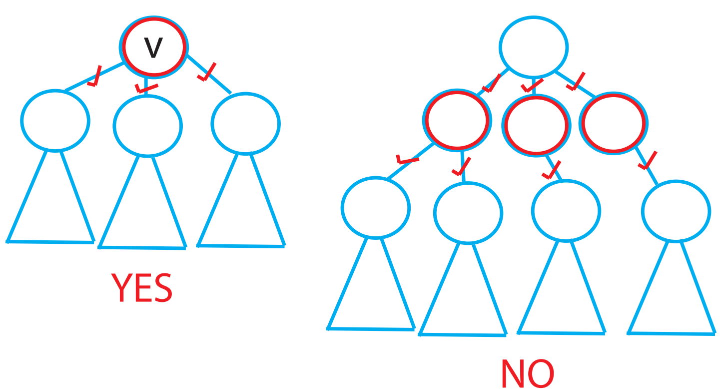

Figure 2: Vertex Cover

- $\implies$ 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. DP[v] = min(1 + sum(DP[c] for c in children[v]) YES
   len(children) + sum(DP[g] for g in grandchildren(v))) NO
4. time = $O(n)$
5. 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 <u>some</u> child in dom. set
     $\implies$ dominate that child's children
3. DP[v] = min(1 + sum($\underbrace{DP'[c]}$ for c in children[v] YES
   but c is already dominated $\cdots$ diff. subprob
   1 + sum($DP(c)$ for $c \neq d$ in children[v])) NO
   +sum($DP'[g]$ for g in children[d])) NO
   again already dominated $\sim$ different subprob
   - guessing of the second type (B)
   for d in children[c]) $\leftarrow$ guess child ϵ set A

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

3'. DP'[v] = min(1 + sum($DP'[c]$ for c in children[v], YES
sum($DP[c]$ for c in children[v])) NO

4. time = $O(\sum deg(v)) = O(E) = O(n)$

5. DP[root]

Beyond:

Treewidth:

Many graphs are "thick trees" with reasonable "thickness" (~ 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

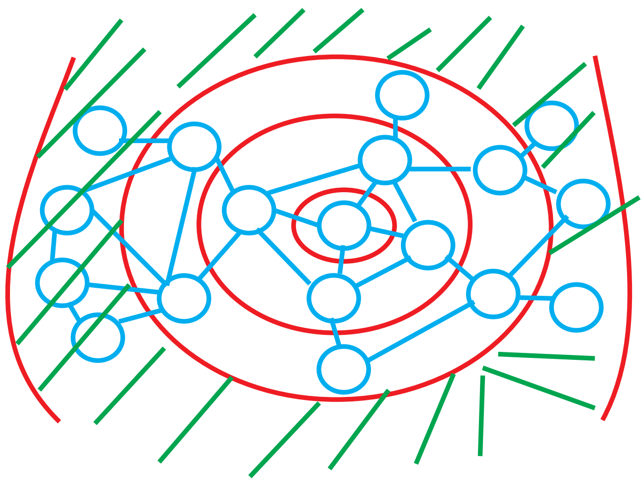

Figure 3: Planar Graphs

- 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 $O(k)$
  $\Longrightarrow$ 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)