6.006	Lecture 22	May 1, 2008

Outline: Dynamic Programming IV (of 4)
- piano fingering
- structural DP (trees)
- vertex cover & dominating set
- beyond: treewidth, planar graphs, folding

Reading: 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 subprob.
- lets parent subproblem know features of sol.

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. $1 < f < g$ & $p > q \Rightarrow$ uncomfortable
  stretch rule: $p \ll q \Rightarrow$ uncomfortable
  legato (smooth) $\Rightarrow \infty$ if $f = g$
  weak-finger rule: prefer to avoid $g \in \{4, 5\}$
  $3 \to 4$ & $4 \to 3$ annoying ~ 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(note[i], f, note[i+1], ?) for f...)~~

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[i, f] = min(DP[i+1, g] + d(note[i], f, note[i+1], g) for g in range(F))

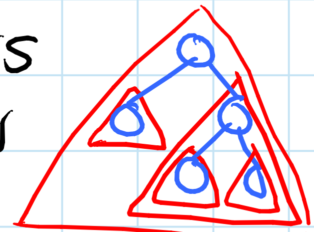
DP[n, f] = $\emptyset$

#fingers = 5 for humans

④ Fn subproblems, F choices per subproblem $\Rightarrow O(F^2 n)$ time
⑤ min(DP[$\emptyset$, f] for f in range(F))

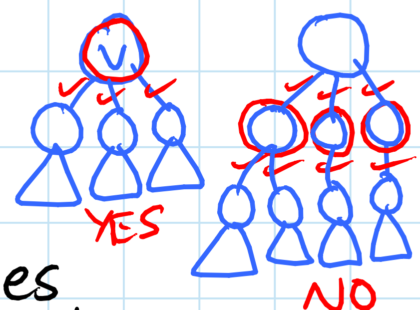
Structural DP: follow combinatorial structure other than a (few) sequence(s)
(by analogy to structural vs. regular induction)

* for DP on trees, useful subproblem is subtree rooted at vertex v, for all v



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
⇒ n subproblems



② guessing = is v in cover?
⇒ 2 choices
- YES ⇒ cover children edges
⇒ left with children subtrees
- NO ⇒ all children must be in cover
⇒ 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]

<u>Dominating set</u>: 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 ⇒ dominate children
  - NO ⇒ must put <u>some</u> child in dom. set ⇒ dominate that child's children

③ DP[v] = min(
  1+sum($\underbrace{\text{DP}'[c]}$ for c in children[v]), } YES
  but c is already dominated --- diff subp. (B)
  1+sum(DP[c] for c≠d in children[v])
  +sum($\underbrace{\text{DP}'[g]}$ for g in children[d]) } NO
  again already dominated ~ different subprob.
  — guessing of type (B)
  for d in children[c]) ← guess child ∈ set (A)

①′ subproblem′ = min. dom. for subtree rooted at v given that v dominated already (by parent subproblem)

⇒ 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

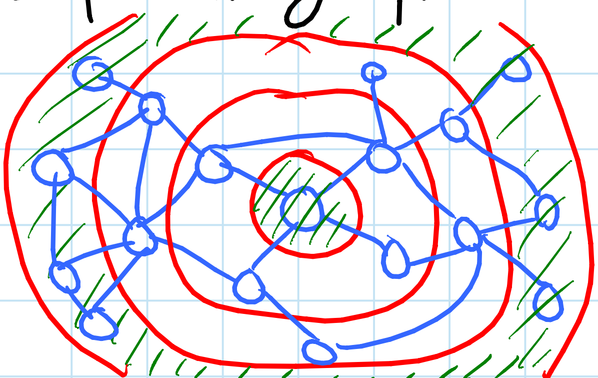
④ time = $O(\sum \deg(v)) = O(E) = O(n)$

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
- divide planar graph into BFS levels:



- throw away every kth level (e.g. k=3) starting from levels Ø, 1, ..., k-1 (guess)
- in all cases, remaining graph is a "thick tree" of thickness O(k)

⇒ 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... ∀ const. k

Folding polygons into polyhedra:
[Metamorphosis of the Cube video]
- DP on substrings of cyclic sequence (polygon)