LEC17 Dyn.Prog 3

Preview

Subproblem definition

- Describe the meaning of a subproblem in words, in terms of parameters
- Often subsets of input: prefixes, suffixes, contiguous substrings of a sequence
- Often multiply possible subsets across multiple inputs
- Often record partial state: add subproblems by incrementing some auxiliary variables
  add subproblems & constraints to "remember state"

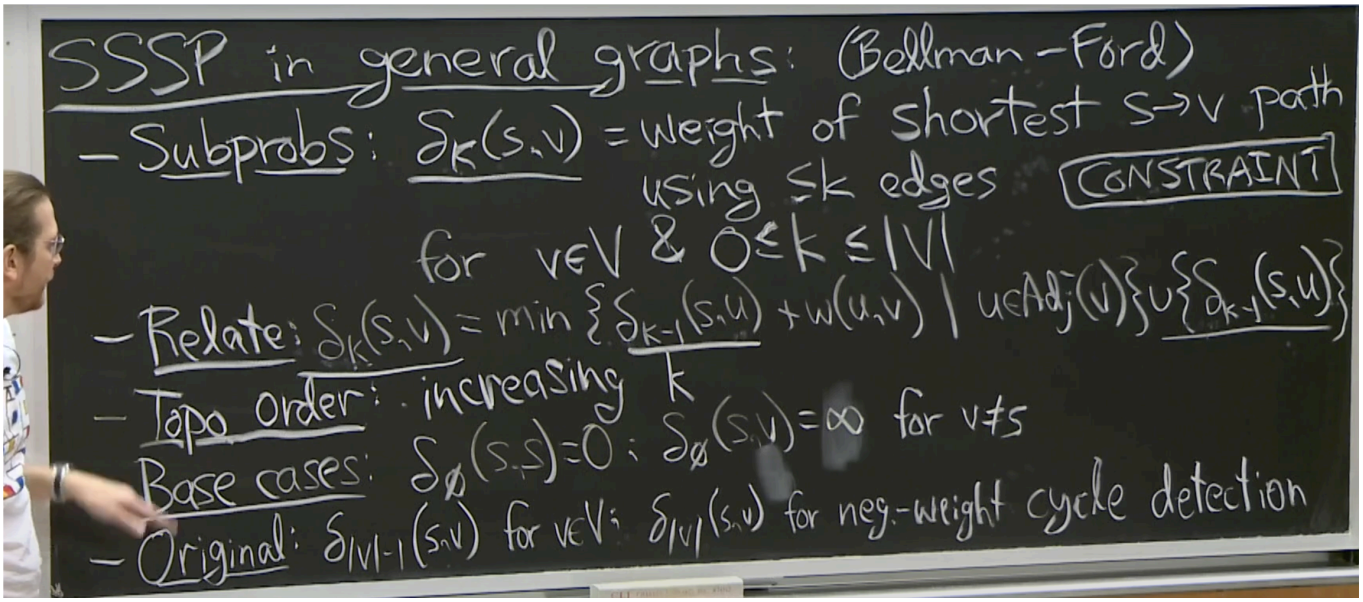
Relate Subproblem solutions recursively

- Identify a question about a subproblem solution that, if you knew the answer to, reduces the subproblem to smaller subproblem(s)
- Locally brute-force all possible answers to the question
  identify question about subproblem solutiont that if you knew answer, reduces to "smaller" subproblems

Today: Subproblem Constraints and Expansion

Examples

SSSP in general graphs: (Bellman-Ford)



APSP

-Subproblems: $\delta(u,v)\ for\ u,v \in V, 0 < k <= |V|$

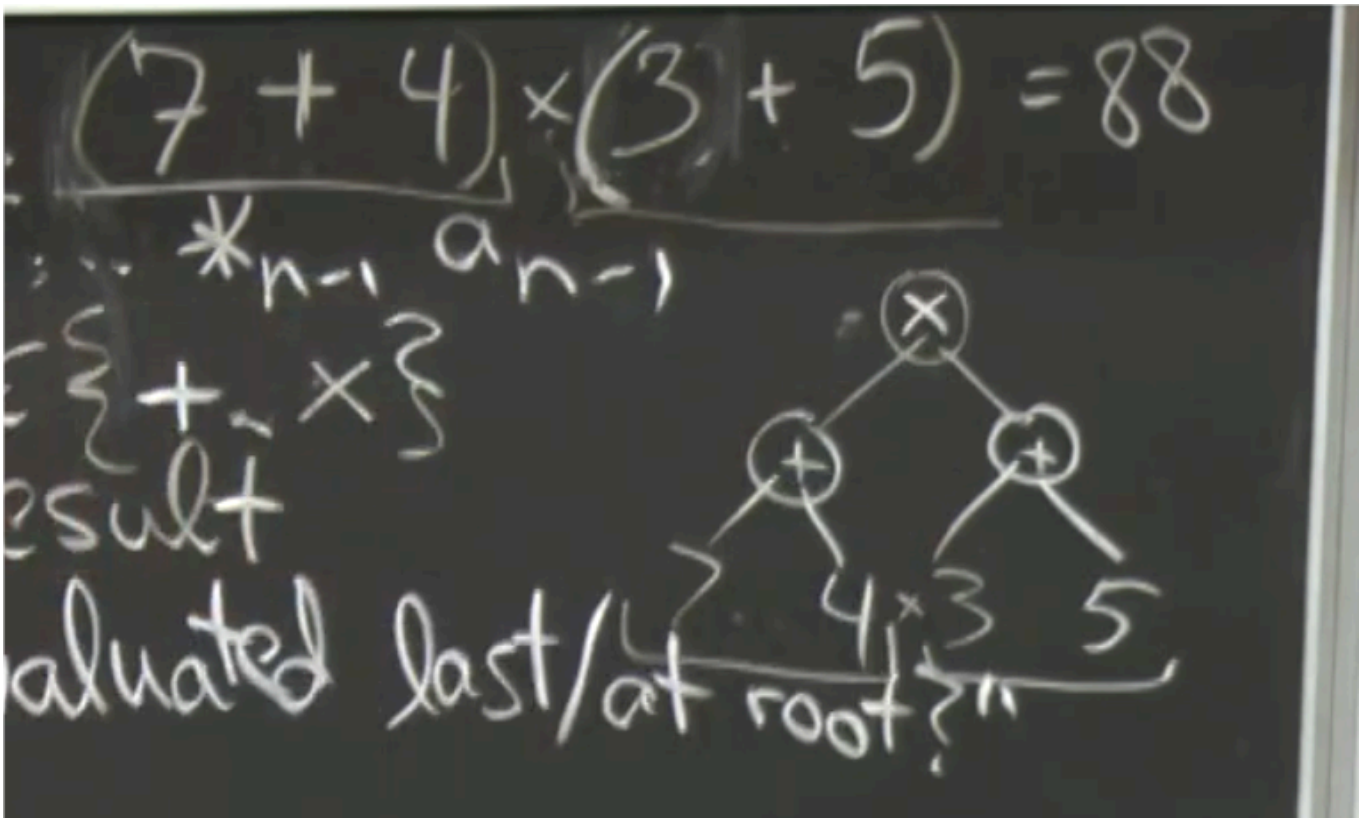
Floyd-Warshall

-number vertices 1,2,...,|V|
-Subproblems: d(u,v,k) = weight of shortest u->v path
add a constraint: using only vertices in {u,v} or {1,2,....,k}.
for u,v in V & 0<= k <= |V| $\theta(|V|^3)$
-Relate:
$d(u,v,k) = min\{d(u,v,k-1), d(u,k,k-1) + d(k,v,k-1)\}$
this k is inceasing by recursation
assume d(u,v,k) is correct
first case: doesn't use vertex k
second case: use vertex k
-Topo order: increasing k
-Base case:

1. $\delta(u,v,0) = 0$ if u =v
2. w(u,v) if (u,v) $\in$ E
3. ∞ otherwise
   -Original problem $\delta(u,v,|V|)$, assuming No negative cycle
   Think every pair of (u,v) is a single DP problem, then the Floyd is V^2 DP problems, and every DP Problems take V times subproblem => O(V^3)
   If u & v is top down iterated, then the algorithm will lose effect. Because base case will no be expended!

Arithmetic Parenthesization

(7+ 4) *(3 + 5)*
-given a formula $a_0*a_1*a_2...a*{n-1}$ where $a_i \in Z, * \in \{+, *\}$
-goal: place parentheses to max result.
-idea: guess "which" operation evaluated last\at root?"



never use a mixture of prefixes and suffixes.

so use substrings

the interger may be nagative

-subproblems: substrings.

x(i,i,opt) = opt value for $a_i *_{i+1} \dots *_{j-1} a_{j-1}$ for 0<=i <j<=n

opt: {min, max}

-Relate: x(i,j,opt) = opt{X(i,k,opt_L) $*_k$ x(k,j, opt_R)} , {{i<k<j}, opt_L, opt_R $\in$ {min, max}}

-Topo: inceasing j - i

-Base: X(i, i+1, opt) = a_i

-Origin: x(0, n, max)

-Time: $\theta(n^2)$ subproblems, * O(n) nonrecursive work = $\theta(n^3)$

Just initlize two arraies to store max_val[i][j], and mini_val[i][j]

just brute force in subproblems.

Piano fingering

given seq single notes t_0,t_1,.....,t_n-1

-fingers 1,2,....,F (=5 for humans)

-metric d(t,f,t',f') of difficulty playing note t with finger f -> palying t' with finger f'

-goal: minimum(sum d(ti,fi,ti+1,f+1))

fi/fi+1 is we need to copute

-Subprob: X(i,f) = min total diff to play t_i,...,tn_1 starting with finger f on note t_i