15-750

1/29/16

Dynamic Programming
An Alg Design Technique

Thousands of Applications

Kai-Fu Lee's Speech Alg

Genome Seq

Clark-Bryant Hardware checking

We will do

1) Classic Ex
2) Not so Classic one.

Your solutions should have a special form.

4-Steps

Optimal Binary Search Trees

Input:

Keys	K_1	K_2		K_n
Freq Prob	P_1	P_2		P_n

Goal: BST s.t.

Cost Search(K_i) = Depth(K_i) eg Depth of root = 1

$$\text{Expected Cost} = \sum_{i=1}^{n} P_i \cdot \text{Depth}(K_i)$$

BST with min expected cost.

Naive Alg: Try all possible trees

Side Question: How may trees(Binary) BT.

We will use Dyn Prog to compute # tree

Counting #BT

Step 1 Give a definition of subproblems being solved

eg $T(n)$ = #BST with n nodes

Step 2 Give a recurrence over subproblems

Include base cases!

eg $T(1)=1$ $T(2)=2$ or $T(0)=1$

$$T(n+1) = \sum_{i=0}^{n} T(i) \cdot T(n-i)$$

Step 3 Prove recurrence correct by induction.

eg Let S_n = set of all n node BTs.

Partition S_n by #nodes in left subtree.

$S_n^0, S_n^1, \ldots, S_n^{n-1}$

Claim $|S_n^i| = T(i) \cdot T(n-i-1)$ by induction

Step 4 Determine run time.

eg Subprobs $T(1), \longrightarrow T(n)$

$C_i \equiv$ cost to compute $T(i)$ from $T(1), \dots, T(i-1)$

$C_i = O(i)$

$\therefore$ Total Cost $= \sum_{i=1}^{n} Ci = O(n^2)$

Finding Opt-BST

Input: $p_1 \cdots p_n$

Trick 1 Compute expect-cost (find tree later)

Trick 2 Solve for more than asked.

eg Solve $C_{ij} \equiv$ expect cost for opt for $p_i \cdots p_j$ $i \le j$

(Step 1) Note: $\sum_{i=k}^{j} p_k < 1$

Step 2 Recurrence relation

Base Case $C_{ij} = p_i$ if $i=j$ ($C_{ij} = 0$ if $j<i$)

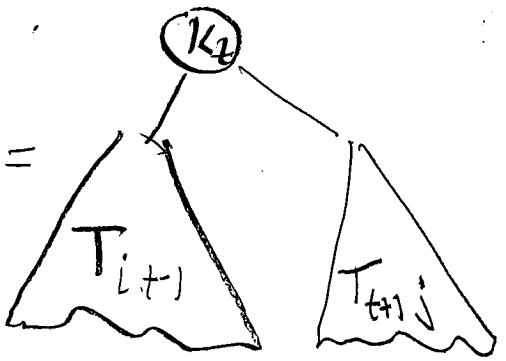
Def $w_{ij} = p_i + \cdots + p_j$

$$C_{ij} = \min_{i \le t \le j} \{ C_{i\,t-1} + C_{t+1\,j} \} + w_{ij} \qquad (*)$$

Step 3 Correctness

Let $T \equiv$ opt tree subject to condition root k_t.

$T =$



$T_L = T_{i\,t-1}$

$T_R = T_{t+1\,j}$

$$Cost(T) = P_t + \sum_{l=i}^{t-1} P_l \left(Depth_T(k_l)\right) + \sum_{l=t+1}^{j} P_l \left(Depth_T(k_l)\right)$$

$$= P_t + \sum_{l \neq t} P_l + \sum P_l \, Depth_{T_L}(k_l) + \sum_{l=t+1}^{j} P_l \, Depth_{T_R}(k_l)$$

$$= W_{ij} + Cost(T_L) + Cost(T_R)$$

Claim $Opt_{ij} \leq C_{ij}$

We could return a tree with cost $= C_{ij}$

the root will be k_t for t min (*)

Claim $C_{ij} \leq Opt_{ij}$

pf induction of j-i

Let T_{ij} be opt tree

Suppose root is k_t.

$$C_{ij} \leq C_{i\,t-1} + C_{t+1\,j} + W_{ij}$$

$$\leq Opt_{i\,t-1} + Opt_{t+1\,j} + W_{ij}$$

$$= Cost(T_{ij}) = Opt_{ij}$$

Recurrence as code (memoization)

Procedure $C(i,j)$

if $j<i$ return 0

else if $i=j$ return P_j

else if $hash(i,j) \neq 0$ return $hash(i,j)$

else

$$hash(i,j) = \min_{i \leq t \leq j} \{ C(i,t-1) + C(t+1,j) \} + W_{ij}$$

return $hash(i,j)$

Timing Table size $= O(n^2)$

Cost per entry $= O(n)$

Total $O(n^3)$

Table Method

$p = .2, .1, .7$

i \ j	1	2	3	4
3	1.4	.9	.7	0
2	.4	.1	0	0
1	.2	0	0	0
0	0	0	0	0

$.4 = .3 + \min\{.1, .2\}$

$.9 = .8 + \min\{.1, .7\}$

$1.4 = 1 + \min\{.9+0, .2+.7, 0+.4\}$

Can we do better? Yes $O(n^2)$ time

Def $r(i,j) \equiv$ index of root of an opt tree

Claim $r(i, j-1) \leq r(i,j) \leq r(i+1, j)$