1/27/09

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

Your solutions should have a special form.

Optimal Binary Search Trees

Input:

Keys	K_1	K_2		K_n
Freq Prob	P_1	P_2		P_n

Goal: BST s.t.

$$\text{Cost Search}(K_i) = \text{Depth}(K_i) \text{ ie Depth of root} = 1$$

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

BST with min expected cost.

Naive Alg: Try all possible trees

Side Question: How may trees?

We will use Dyn Prog.

<u>Step 1</u> Give a definition of subproblems being solved

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

<u>Step 2</u> Give a recurance over subproblems Include base cases.

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

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

<u>Step 3</u> Prove recurance correct by induction.

<u>eg</u> Let S_n = set of all n node BSTs.

Partition S_n by # nodes in left subtree.

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

<u>Claim</u> $|S_n^i| = T(i) \cdot T(n-i)$ by induction

<u>Step 4</u> Determine run time.

<u>eg</u> Subprob $T(1), \ldots, T(n)$

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

$C_i = O(i)$

$\therefore$ Total Cost $= O(n^2)$

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)

Step 2 Recurance relation

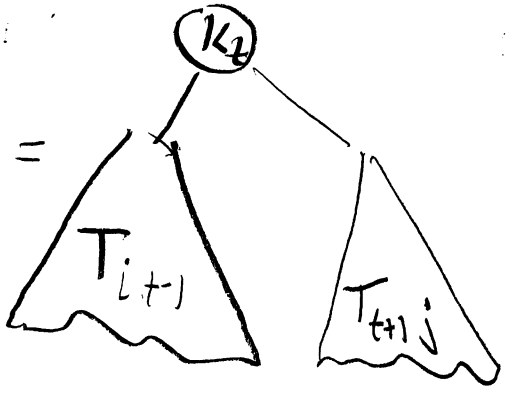
Base Case $C_{ij} = p_i$ if $i = j$ ($C_{ij} = 0$ if $j < i$)

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

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

Step 3 Correctness

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

$T =$ 

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

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

$$\mathrm{Cost}(T) = P_t + \sum_{l=i}^{t-1} P_l \left(\mathrm{Depth}_T(K_l)\right) + \sum_{l=t+1}^{j} P_l \left(\mathrm{Depth}_T(K_l)\right)$$

$$= P_l + \sum_{l \neq t} P_l + \sum P_l \, \mathrm{Depth}_{T_L}(K_l) + \sum_{l=t+1}^{j} P_l \, \mathrm{Depth}_{T_R}(K_l)$$

$$W_{ij} + \mathrm{Cost}(T_L) + \mathrm{Cost}(T_R)$$

<u>Claim</u> $Opt_{ij} \le C_{ij}$

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

the root will be k_t for t min $(*)$

<u>Claim</u> $C_{ij} \le Opt_{ij}$

pf induction of $j-i$

Let T_{ij} be opt tree

Suppose root is k_t

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

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

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

Recurrance as code (memoization)

Procedure $C(i,j)$

if $k<j$ return 0

else if $i=j$ return P_j

elseif $hash(i,j) \neq 0$ return $hash(i,j)$

else

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

return $hash(i,j)$

<u>Timing</u> Table size $= O(n^2)$

Cost per entry $= O(n)$

Total $O(n^3)$

Table Method

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

1.4	.9	.7
.4	.1	
.2		

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

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

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

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

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

<u>Claim</u> $r(i, t-1) \le r(i,j) \le r(t+1, j)$