3-1 Dynamic Programming

(Part I: Examples)

Hengfeng Wei

hfwei@nju.edu.cn

September 17, 2018

Taolu

Steps for Applying DP:

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence
  - larger subproblem $\leftarrow$ # smaller subproblems
  - init. conditions

(IV) Write pseudo-code: filling in “tables” in some order

(V) Analyze the time complexity

(VI) Extract the optimal solution (optionally)

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence

- larger subproblem $\leftarrow$ # smaller subproblems
- init. conditions

(IV) Write pseudo-code: filling in "tables" in some order

(V) Analyze the time complexity

(VI) Extract the optimal solution (optionally)

1D Subproblems

Input: $x_1, x_2, \ldots, x_n$ (array, sequence, string)

Subproblems: $x_1, x_2, \ldots, x_i$ (prefix/suffix)

#: $\Theta(n)$

Examples:

- Rod cutting
- Maximum-sum subarray
- Longest increasing subsequence
- Printing Neatly

2D Subproblems

(I) Input: $x_1, x_2, \ldots, x_m; \quad y_1, y_2, \ldots, y_n$

Subproblems: $x_1, x_2, \ldots, x_i; \quad y_1, y_2, \ldots, y_j$

#: $\Theta(mn)$

Examples: Edit distance, Longest common subsequence

2D Subproblems

(I) Input: $x_1, x_2, \ldots, x_m; \quad y_1, y_2, \ldots, y_n$

Subproblems: $x_1, x_2, \ldots, x_i; \quad y_1, y_2, \ldots, y_j$

#: $\Theta(mn)$

Examples: Edit distance, Longest common subsequence

(II) Input: $x_1, x_2, \ldots, x_n$

Subproblems: $x_i, \ldots, x_j$

#: $\Theta(n^2)$

Examples: Matrix chain multiplication, Optimal BST

3D Subproblems

- Floyd-Warshall algorithm

$$\mathsf{d}(i,j,k) = \min\Big(\mathsf{d}(i,j,k-1), \mathsf{d}(i,k,k-1) + \mathsf{d}(k,j,k-1)\Big)$$

DP on Graphs

(I) On rooted tree

Subproblems: rooted subtrees

(II) On DAG

Subproblems: nodes after/before in the topo. order

DP on Graphs

(I) On rooted tree

Subproblems: rooted subtrees

(II) On DAG

Subproblems: nodes after/before in the topo. order

Knapsack Problem

Subset sum problem, Change-making problem

And Others . . .

And Others . . .

How to identify the recurrence?

How to identify the recurrence?

GUESS

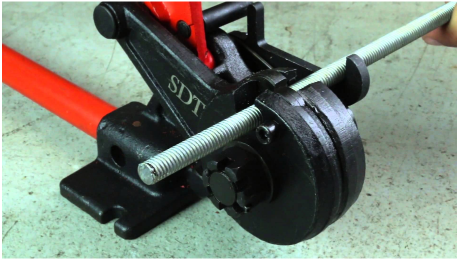
Make Choices by asking yourself the right question

Make Choices by asking yourself the right question

(I) Binary choice
- whether . . .

(II) Multi-way choices
- where to . . .
- which one . . .

Rod Cutting



Rod Cutting Problem

Rod of length n

length i	1	2	3	4	5	$\cdots$
price p_i	1	5	8	9	10	$\cdots$

$$n = i_1 + i_2 + \cdots + i_k$$

$$r_n = p_{i_1} + p_{i_2} + \cdots + p_{i_k}$$

$R(i)$: $\max$ revenue obtained from *cutting a rod of length i*

$$R(n)$$

$R(i)$: $\max$ revenue obtained from *cutting a rod of length i*

$$R(n)$$

Where is the leftmost cut?

$R(i) : \max$ revenue obtained from *cutting a rod of length i*

$$R(n)$$

Where is the leftmost cut?

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j) \Big)$$

$R(i)$: $\max$ revenue obtained from $\boxed{\textit{cutting a rod of length } i}$

$$R(n)$$

Where is the leftmost cut?

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j)\Big)$$

$$R(0) = 0$$

$R(i)$: max revenue obtained from *cutting a rod of length i*

$$R(n)$$

Where is the leftmost cut?

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j)\Big)$$

$$R(0) = 0$$

$$O(n^2) = \Theta(n) \cdot O(n)$$

Rod Cutting Problem (Problem 15.1-3)

Each cut incurs a fixed cost of c.

Rod Cutting Problem (Problem 15.1-3)

Each cut incurs a fixed cost of c.

$$R(i) = \max_{1 \le j \le i} \Big(p_j - c + R(i-j) \Big)$$

Rod Cutting Problem (Problem 15.1-3)

Each cut incurs a fixed cost of c.

$$R(i) = \max_{1 \le j \le i} \Big(p_j - c + R(i-j) \Big)$$

$$R(i) = \max \Big(p_i, \max_{1 \le j < i} \left(p_j - c + R(i-j) \right) \Big)$$

Printing Neatly (Problem 15-4)

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

Line width M

$$\textit{extra}[i, j] = M - (j - i) - \sum_{k=i}^{j} l_k$$

Printing Neatly (Problem 15-4)

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

Line width M

$$\textit{extra}[i,j] = M - (j-i) - \sum_{k=i}^{j} l_k$$

To minimize the sum, over all lines *except the last*, of the cubes of the numbers of extra space characters at the ends of lines.

$$C(n) = \min_{L} \sum_{l_{[i,j]} \in L \land j \neq n} \Big(\textit{extra}[i,j]\Big)^3$$


LaTeX
Beginner's Guide
TEX}

$\TeX$ 3.14159265 $\quad \sim \pi$

A $\boxed{\text{sequence}}$ of n words of lengths $l_1, l_2, \cdots, l_n$

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

$C(i)$: $\min$ cost of neatly printing the *first* i words

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

$C(i, j)$: $\min$ cost of neatly printing *words* i *through* j

$C(i) : \min$ cost of neatly printing the *last* words i through n

$C(i) : \min$ cost of neatly printing the *last* words i through n

$\boxed{\textit{How many words to place on the first line?}}$

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \le (n-i+1) \\ \textit{extra}[i, i+k-1] \ge 0}} \Big(\textit{extra}[i, i+k-1]\Big)^3 + C(i+k)$$

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \le (n-i+1) \\ \textit{extra}[i,\, i+k-1] \ge 0}} \Big(\textit{extra}[i, i+k-1]\Big)^3 + C(i+k)$$

$$C(i) = 0, \quad \text{if } \textit{extra}[i, n] \ge 0$$

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \textit{extra}[i,\, i+k-1] \geq 0}} \Big(\textit{extra}[i, i+k-1] \Big)^3 + C(i+k)$$

$$C(i) = 0, \quad \text{if } \textit{extra}[i, n] \geq 0$$

$$O(nM)$$

procedure PRINTING-NEATLY(n)

  for $i \leftarrow n \downarrow 1$ **do**

    if *extra*$[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line

      $C[i] \leftarrow 0$

    else

      $$C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \textit{extra}[i, i+k-1] \geq 0}} \Big(\textit{extra}[i, i+k-1]\Big)^3 + C(i+k)$$

procedure PRINTING-NEATLY(n)

  for $i \leftarrow n \downarrow 1$ **do**

    if $\textit{extra}[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line

      $C[i] \leftarrow 0$

      $B[i] \leftarrow n$

    else

      $C(i) = \min\limits_{\substack{0 < k \leq (n-i+1) \\ \textit{extra}[i, i+k-1] \geq 0}} \Big(\textit{extra}[i, i+k-1]\Big)^3 + C(i+k)$

      $B[i] = i + k$ ▷ line break

```
1: procedure PRINTING-NEATLY(n)
2:     for i ← n ↓ 1 do
3:         if extra[i, n] ≥ 0 then                    ▷ put w_i through w_n on a line
4:             C[i] ← 0
5:             B[i] ← n
6:         else
7:             C(i) = min_{0 < k ≤ (n−i+1), extra[i, i+k−1] ≥ 0} (extra[i, i+k−1])^3 + C(i+k)
8:             B[i] = i + k                                ▷ line break
```

$B[1]$,

procedure PRINTING-NEATLY(n)
  for $i \leftarrow n \downarrow 1$ **do**
    if *extra*$[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line
      $C[i] \leftarrow 0$
      $B[i] \leftarrow n$
    else
      $C(i) = \min\limits_{\substack{0 < k \leq (n-i+1) \\ \textit{extra}[i,i+k-1] \geq 0}} \Big(\textit{extra}[i, i+k-1]\Big)^3 + C(i+k)$
      $B[i] = i + k$ ▷ line break

$$B[1], \quad B[B[1]+1], \quad \cdots$$

Neatness: $c[i, j] = \textit{extra}[i, j]^{1}$

— *Tianyun Zhang*

Neatness: $c[i,j] = \textit{extra}[i,j]$[1]

— *Tianyun Zhang*

Neatness: $c[i, j] = \textit{extra}[i, j]^{1}$

— *Tianyun Zhang*

Proof: Exchange Argument

Longest Increasing Subsequence (Problem 15.4-5)

$$A[1 \dots n]$$

$$5,\ 2,\ 8,\ 6,\ 3,\ 6,\ 9,\ 7$$

Find (the length of) a longest increasing (non-decreasing) subseqence.

$$5,\ \color{red}{2},\ 8,\ 6,\ \color{red}{3},\ \color{red}{6},\ \color{red}{9},\ 7$$

$L(i)$: $\boxed{\text{the length of an LIS of } A[1 \ldots i]}$

$$L(n)$$

$L(i)$: $\boxed{\text{the length of an LIS of } A[1 \ldots i]}$

$$L(n)$$

$\boxed{\textit{Is } A[i] \textit{ in this LIS of } A[1 \cdots i]\textit{?}}$

$L(i)$: $\boxed{\text{the length of an LIS of } A[1 \ldots i]}$

$$L(n)$$

$\boxed{\textit{Is } A[i] \textit{ in this LIS of } A[1 \cdots i]\textit{?}}$

$$L(i) = \max \Big(\underbrace{L(i-1)}_{\text{NO}}, \qquad\qquad\qquad \Big)$$

$L(i)$: $\boxed{\text{the length of an LIS of } A[1 \ldots i]}$

$$L(n)$$

$$\boxed{\textit{Is } A[i] \textit{ in this LIS of } A[1 \cdots i]\textit{?}}$$

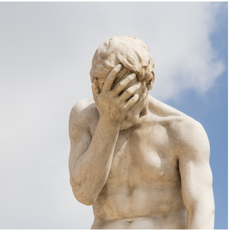
$$L(i) = \max \Big(\underbrace{L(i-1)}_{\text{NO}}, \underbrace{1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)}_{\text{YES}} \Big)$$

$L(i)$: $\boxed{\text{the length of an LIS of } A[1 \ldots i]}$

$$L(n)$$

$\boxed{\textit{Is } A[i] \textit{ in this LIS of } A[1 \cdots i]\textit{?}}$

$$L(i) = \max \Big(\underbrace{L(i-1)}_{\text{NO}}, \underbrace{1 + \max_{j<i \,\land\, A[j] \le A[i]} L(j)}_{\text{YES}} \Big)$$



$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j < i \,\wedge\, A[j] \le A[i]} L(j)$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)$$

$$L(1) = 1$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)$$

$$L(1) = 1$$

$$O(n^2) = \Theta(n) \cdot O(n)$$

$$\mathsf{LIS(A)} = \mathsf{LCS}\Big(A, \textsc{Sort}(A)\Big)$$

$$\mathsf{LIS(A)} = \mathsf{LCS}\Big(A, \textsc{Sort}(A)\Big)$$

$$O(n^2) = O(n \log n) + O(n^2)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

Answer by Eugene Yarovoi @ Quora

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

procedure *LIS*(n)

$E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$

for $i \leftarrow 1 \uparrow n$ **do**

$\forall 1 \le l \le i : E[l] \leftarrow$

$\{\textit{all inc. subseq. of length } l\}$

return

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

procedure *LIS*(n)
  $E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$
  for $i \leftarrow 1 \uparrow n$ **do**
    $\forall 1 \le l \le i : E[l] \leftarrow$
      $\{\textit{all inc. subseq. of length } l\}$
  return $\max \{l \mid E[l] \neq \emptyset\}$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

---

procedure *LIS*(n)
    $E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$
    for $i \leftarrow 1 \uparrow n$ **do**
        $\forall 1 \le l \le i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*
            $\{\textit{all inc. subseq. of length } l\}$
    return $\max\{l \mid E[l] \neq \emptyset\}$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$E[l] = \{$*all increasing subsequences of length* $l\}$

$$\langle 3 \quad 9 \quad 11 \quad 13 \rangle$$

$$\langle 4 \quad 6 \quad 10 \quad 15 \rangle$$

$$\langle 2 \quad 5 \quad 12 \quad 18 \rangle$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\langle 3 \quad 9 \quad 11 \quad 13 \rangle$$

$$\langle 4 \quad 6 \quad 10 \quad 15 \rangle$$

$$\langle 2 \quad 5 \quad 12 \quad 18 \rangle$$

$\Big|E[l]\Big| = 1$: *the one with the smallest ending number*

procedure *LIS*(n)

$E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$

for $i \leftarrow 1 \uparrow n$ **do**

$\forall 1 \le l \le i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*

the inc. subseq. of length l with the smallest ending number

return $\max \{l \mid E[l] \neq \emptyset\}$

$E[l] = \{$*all increasing subsequences of length* $l\}$

$\Big|E[l]\Big| = 1$: *the one with the smallest ending number*

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\Big|E[l]\Big| = 1 : \textit{the one with the smallest ending number}$$

$$E[3] = \{\langle 2 \quad 8 \quad 15 \rangle\}$$

$$E[4] = \{\langle 4 \quad 6 \quad 11 \quad 13 \rangle\}$$

$$E[5] = \{\langle 3 \quad 9 \quad 11 \quad 12 \quad 13 \rangle\}$$

$E[l] = \{\textit{all increasing subsequences of length } l\}$

$\Big|E[l]\Big| = 1$: *the one with the smallest ending number*

$$E[3] = \{\langle 2 \quad 8 \quad 15 \rangle\}$$

$$E[4] = \{\langle 4 \quad 6 \quad 11 \quad 13 \rangle\}$$

$$E[5] = \{\langle 3 \quad 9 \quad 11 \quad 12 \quad 13 \rangle\}$$

$\forall i < j$: *the ending number of* $E[i]$ $<$ *the ending number of* $E[j]$

procedure *LIS*(n)

  $E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$

  for $i \leftarrow 1 \uparrow n$ **do**

    $\forall 1 \leq l \leq i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*

      the inc. subseq. of length l with the smallest ending number

  return $\max \{l \mid E[l] \neq \emptyset\}$

procedure *LIS*(n)

$E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$

for $i \leftarrow 1 \uparrow n$ **do**

$\forall 1 \leq l \leq i : E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*

the inc. subseq. of length l *with the smallest ending number*

return $\max \{l \mid E[l] \neq \emptyset\}$

Extending

Extending

by looking at only the ending number of the inc. subseq

Extending

by looking at only the ending number of the inc. subseq

procedure *LIS*(n)
    $E[l] \leftarrow \qquad \forall 1 \le i \le n$
    for $i \leftarrow 1 \uparrow n$ **do**
        $E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*
            the smallest ending number for the inc. subseq. of length l
    return

Extending

by looking at only the ending number of the inc. subseq

---

procedure *LIS*(n)

$E[l] \leftarrow \infty \quad \forall 1 \le i \le n$

for $i \leftarrow 1 \uparrow n$ **do**

$E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*

the smallest ending number for the inc. subseq. of length l

return

Extending

by looking at only the ending number of the inc. subseq

procedure *LIS*(n)

$E[l] \leftarrow \infty \quad \forall 1 \le i \le n$

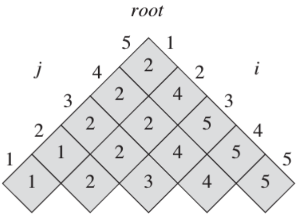
for $i \leftarrow 1 \uparrow n$ **do**

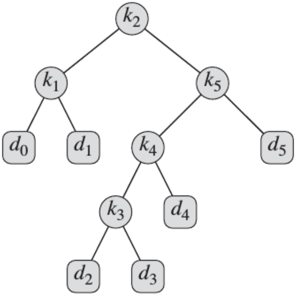
$E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*

the smallest ending number for the inc. subseq. of length l

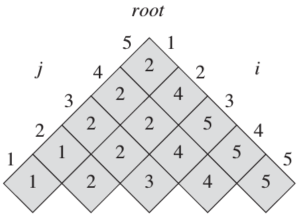
return $\max \{l \mid E[l] < \infty\}$

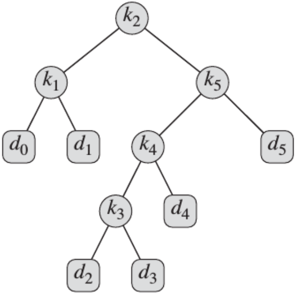
CONSTRUCT-OPTIMAL-BST(root) (Problem 15.5-1)





CONSTRUCT-OPTIMAL-BST(root) (Problem 15.5-1)





What about $d_0, d_1, \cdots, d_n$?

CONSTRUCT-OPTIMAL-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

Construct-Optimal-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i, r-1] + e[r+1, j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

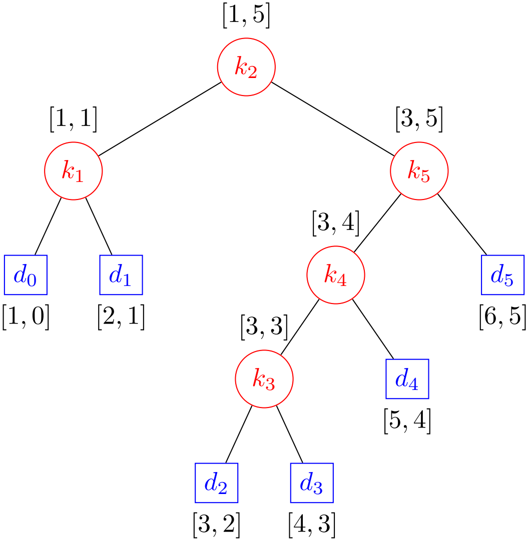
```
procedure CONSTRUCT-OPTIMAL-BST(root, i, j)
    if j = i − 1 then
        return node with d_{i−1}

    rn ← node with key k_{root[i,j]}
    rn.left ← CONSTRUCT-OPTIMAL-BST(root, i, root[i, j] − 1)
    rn.right ← CONSTRUCT-OPTIMAL-BST(root, root[i, j] + 1, j)

    return rn
```

Construct-Optimal-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i, r-1] + e[r+1, j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

1: **procedure** Construct-Optimal-BST($root$, i, j)
2:     **if** $j = i - 1$ **then**
3:         **return** node with d_{i-1}
4:     $rn \leftarrow$ node with key $k_{root[i,j]}$
5:     $rn.left \leftarrow$ Construct-Optimal-BST$(root, i, root[i,j] - 1)$
6:     $rn.right \leftarrow$ Construct-Optimal-BST$(root, root[i,j] + 1, j)$
7:     **return** rn

Construct-Optimal-BST$(root, 1, n)$


[1, 5]
k_2
[1, 1]
k_1
[3, 5]
k_5
[3, 4]
k_4
d_0
d_1
d_5
[1, 0]
[2, 1]
[6, 5]
[3, 3]
k_3
d_4
[5, 4]
d_2
d_3
[3, 2]
[4, 3]

procedure CONSTRUCT-OPTIMAL-BST($root$, i, j, r) ▷ r : *parent*

  if $r = i$ **then**

    print "d_{i-1} is the *left* child of k_r" **return** node with d_{i-1}

  if $r = j$ **then**

    print "d_{i-1} is the *right* child of k_r" **return** node with d_{i-1}

  $r' \leftarrow root[i, j]$ $rn \leftarrow$ node with key $k_{r'}$

  if $r = 0$ **then**

    print "$k_{r'}$ is the root"

  else if $j < r$ **then**

    print "$k_{r'}$ is the *left* child of k_r"

  else if $i > r$ **then**

    print "$k_{r'}$ is the *right* child of k_r"

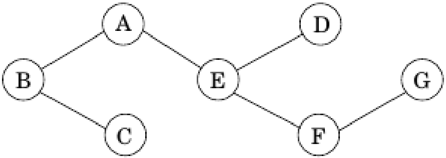
  rn.*left* $\leftarrow$ CONSTRUCT-OPTIMAL-BST($root, i, r' - 1, r'$)

  rn.*right* $\leftarrow$ CONSTRUCT-OPTIMAL-BST($root, r' + 1, j, r'$)

  return rn

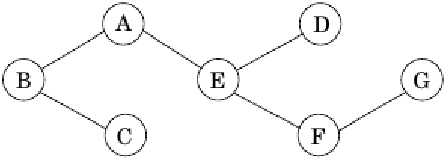
Minimum Vertex Cover on Trees (Additional Problem)

- Undirected tree $T = (V, E)$; *No designated root*
- Compute (the size of) a minimum vertex cover (MVC) of T



Minimum Vertex Cover on Trees (Additional Problem)

- Undirected tree $T = (V, E)$; *No designated root*
- Compute (the size of) a minimum vertex cover (MVC) of T



Rooted T at any node r.

$I(u)$: the size of an MVC of subtree T_u rooted at u

$$I(r)$$

$I(u)$: the size of an MVC of subtree T_u rooted at u

$I(r)$

Is u in MVC$[u]$?

$I(u)$: the size of an MVC of $\boxed{\text{subtree } T_u \text{ rooted at } u}$

$$I(r)$$

$\boxed{\textit{Is } u \textit{ in MVC}[u]\textit{?}}$

$$I(u) = \min \Big\{ \underbrace{1 + \sum_{v:\ \text{children of } u} I(v)}_{\in}, \underbrace{\#\ \text{children of } u + \sum_{v:\ \text{grandchildren of } u} I(v)}_{\notin} \Big\}$$

$I(u)$: the size of an MVC of $\boxed{\text{subtree } T_u \text{ rooted at } u}$

$$I(r)$$

$\boxed{\textit{Is } u \textit{ in MVC}[u]\textit{?}}$

$$I(u) = \min \Big\{ \underbrace{1 + \sum_{v:\ \text{children of } u} I(v)}_{\in}, \underbrace{\#\ \text{children of } u + \sum_{v:\ \text{grandchildren of } u} I(v)}_{\notin} \Big\}$$

$$I(u) = 0, \text{ if } u \text{ is a leave}$$


#10
IS GREED
GOOD?

There is an MVC which contains no leaves.

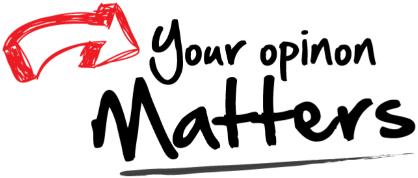
There is an MVC which contains no leaves.

Proof: Exchange Argument

There is an MVC which contains no leaves.

Proof: Inductive Exchange Argument


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn