ACM Training
Dynamic Programming

Outlines

- Problems on Recursive Algorithm
- What is Dynamic Programming?
- Examples
- Characteristics
- Exercises
- References

Problems on Recursion

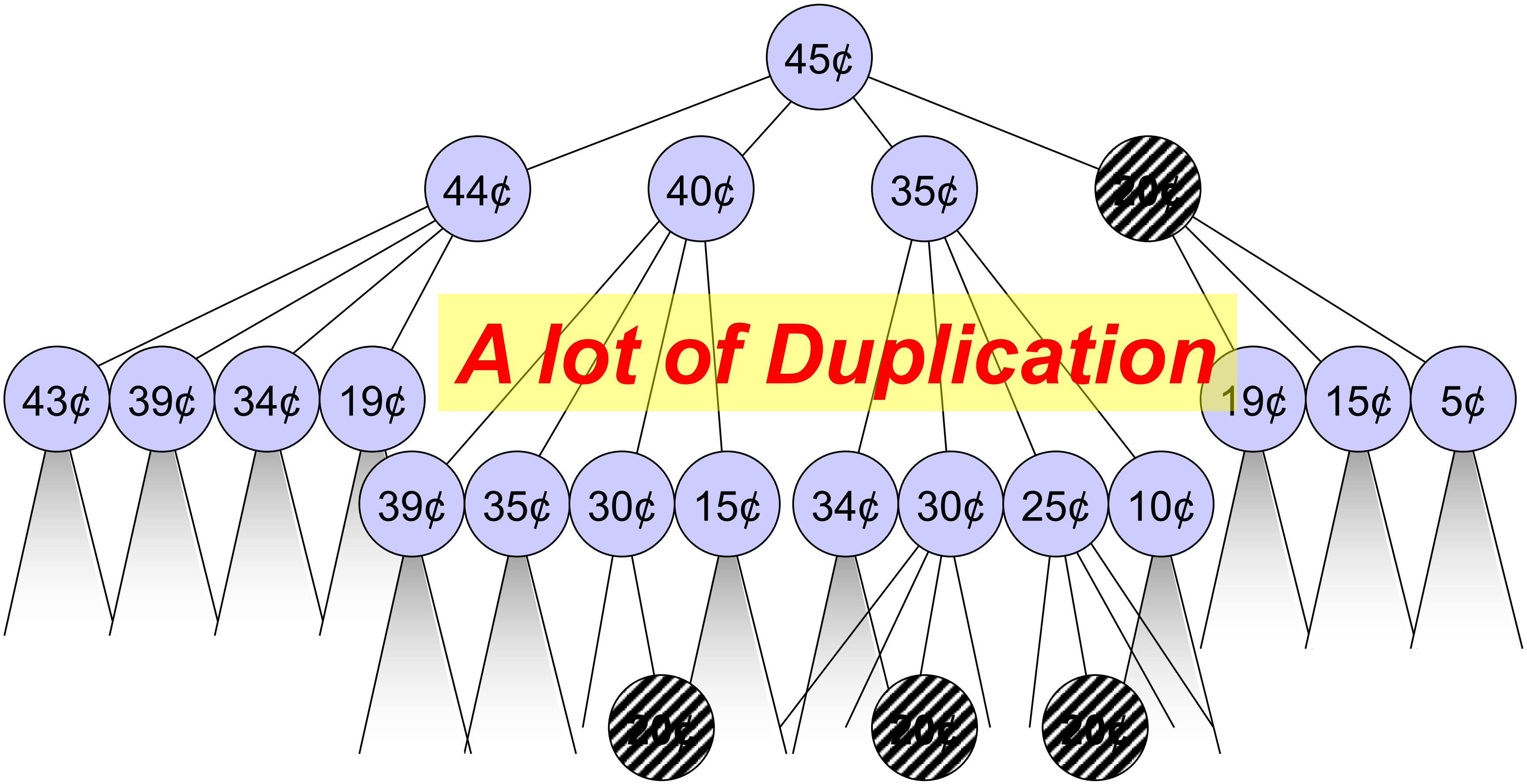
- Excessive Function Calls
  - Performance Suffering related to the depth of recursion
  - N levels of function calls for the recursive version of the Factorial (N)
  - Stack overflow
- Overlapped Sub-Problems
  - Duplicated execution on solving the same sub-problems

Overlapped Sub-Problems

- Minimum Number of Coins
  - Coins[4] = { 1¢ , 5¢ , 10¢ , 25¢ }
  - Find minimum number of coins whose total value equals a specified amount
  - MinCoins (0) = 0
  - MinCoins (m) = min{ MinCoins (m-1¢), MinCoins (m-5¢), MinCoins (m-10¢), MinCoins (m-25¢) } + 1

Revision on Recursion (4)

- Minimum Number of Coins



What is Dynamic Programming?

- Not directly related to any existing programming language
- Solve recursive algorithms with the non-recursive ways
- Typically applied to optimization problems

Development Steps

1. Characterize the structure of optimal solution
   - The optimal solution must come from optimal solutions of the sub-problems
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a **bottom-up** fashion
   - Start from solving the smallest sub-problems first

Development Steps

4. Construct an optimal solution from computed information
   - Not required for every problem
     - e.g. *Minimum Number of Coins*
   - Need to carefully handle the use of ***memory space*** and its ***structure***
   - Try to re-use the memory space
     - Sub-sub-problem may not need to be kept when the solution of the sub-problem is found

Essential Components

- State space (table entry definition)
- State Transition Function
- Complexity=A*B*C
  - A: Number of states
  - B: Number of combinations (branches) in deciding the optimal for each state
  - C: Cost in calculating the value of each branch

Example2 LCS

- Longest Common Sub-sequence
  - Given 2 character sequences (A, B)
  - Sub-sequence
    - Pattern of characters in A appears in B

```
Seq_A    = CAGT TACGC AGGT A
Sub_Seq = CA T    CGC A      A
Seq_B    = CA TG   CGCTA    TAA
```

    - Many possible solutions
    - Some of the solutions are the optimal
  - Answer: the maximum length
  - Answer: the optimal sub-sequence(s)

- Longest Common Sub-sequence
  - Recursive Solution

```
int lcslen (char* A, char* B)
{
  if (*A == 0 || *B == 0)
    return 0;

  else if (*A == *B)
    return lcslen (A+1, B+1) + 1;

  else
    return max(lcslen (A+1, B),
               lcslen (A, B+1));
}
```

Longest Common Sub-sequence

```
LCS-LENGTH(X,Y)
    m ← length [X]
    n ← length [Y]
    for i ← 1 to m
      do c[i,0] ← 0
    for j ← 0 to n
      do c[0,j] ← 0
    for i ← 1 to m
      do for j ← 1 to n
        do if x[i] = y[j]
             then c[i,j] ← c[i-1,j-1] + 1
                  b[i,j] ← "↖"
             else if c[i-1,j] ≥ c[i, j - 1]
                  then c[i,j] ← c[i - 1, j]
                       b[i,j] ← "↑"
                  else c[i,j] ← c[i, j -1]
                       b[i,j] ← "←"
 return c and b
```

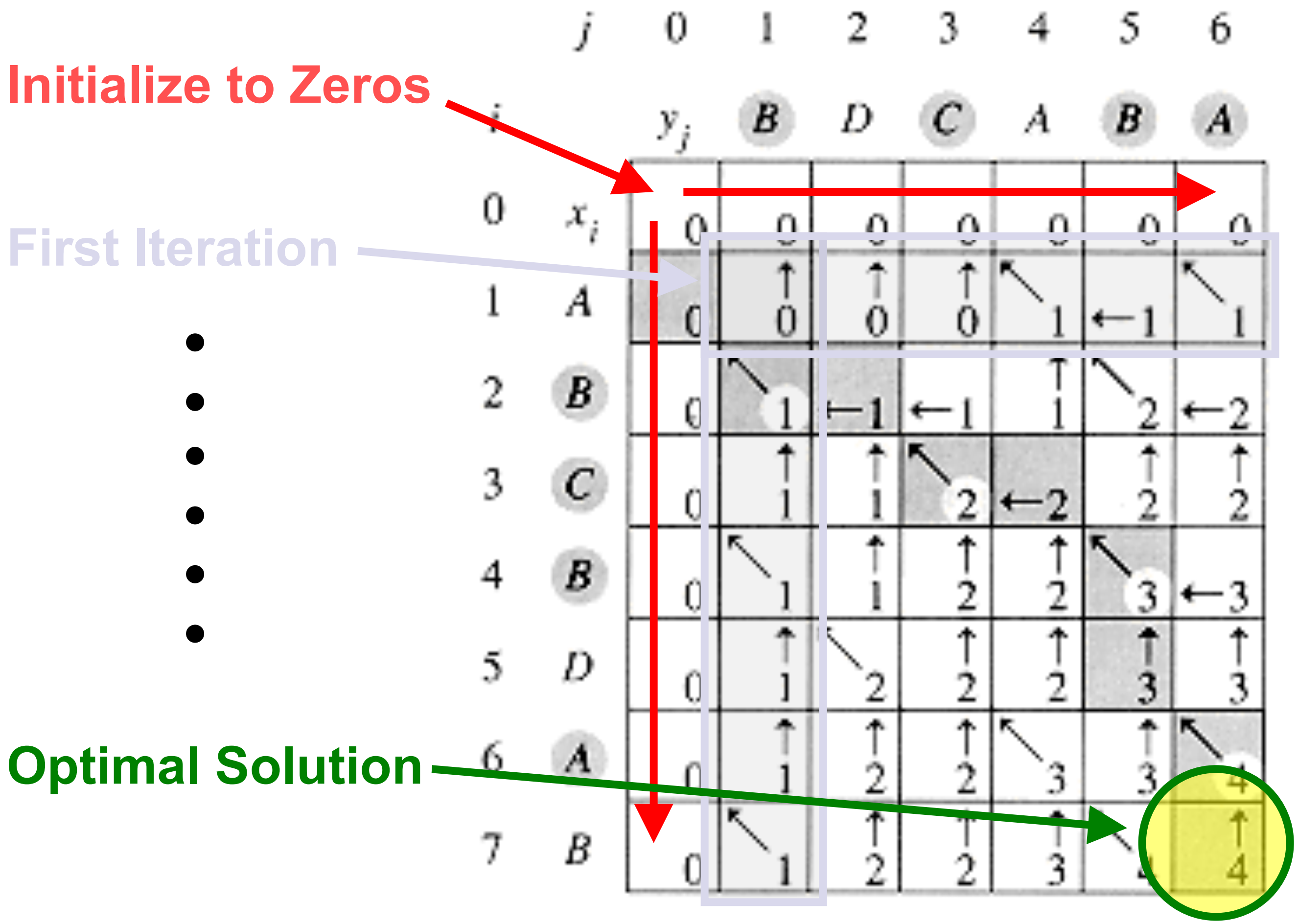
Dynamic Programming Version

- Longest Common Sub-sequence

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

BCBA

Longest Common Sub-sequence

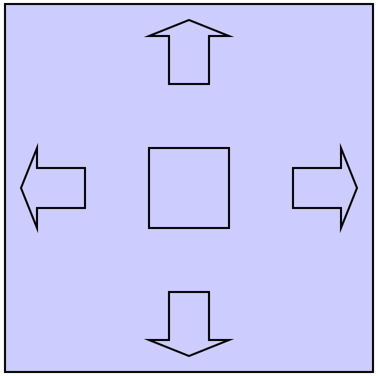


Hands-on Exercise #1

- Online judge 10405 – Longest Common Subsequence

Example 2—Dance Dance Revolution

- Mr. White is very fat but like DDR so much
- He’d like to use least energy to correctly follow a melody
- Rules:
  - When a new symbol comes, he should move one of the feet to that symbol while keeping the other in the original place
- How to calculate energy?
  - Moving cost:
    - Adjacent move: 3
    - Opposite move: 4
    - Center to any: 2
    - Tap the same position: 1



How to choose suitable state?

- Enough to use d[i] where i is the index for the steps?
- We should choose d[i, j, k] where j and k represents the positions of left foot and right foot respectively
- Criteria
  - Future decisions only depend on the current state

Hands-on exercise #2

- Online judge 357 – Let Me Count the Ways

Dynamic Programming Approach

- Top-Down Approach (Memorizing)
  - Like the recursive algorithm
  - Look up the table before computing the solution
- Bottom-Up Approach
  - Usually more efficient

Dynamic Programming Approach

- Maintaining the table structure
  - Can be multiple dimensions according to the problem set
    - `char table[MTHS][DAYS][MAX_LEN]`
  - Save sub-results for later use
  - Avoid the evaluation of the same sub-problems *(like Minimum Number of Coins)*

Different Models

- Linear Model
- String Model
- Interval Model
- State Compression Model
- Tree Model

Interval Model

- Jumping frog
  - There are n stones (1<n<1000) in the pond. All the stones are on the same circle.
  - A frog wants to jump to every stone exactly once.
  - Try to design a route so that the total distance covered by the frog is minimized

Tree Model

- Finding Maximum Comfortable Group on Tree
  - Input: a tree
  - Output: a comfortable group of maximum size
- What is comfortable group?
  - A set of nodes with no edge between any pair
- How to define state (subproblem)?

Other Examples on Trees

- Minimize longest path in a tree

Exercises

- 231: Testing the CATCHER
- http://online-judge.uva.es/p/v2/231.html
- 165: Stamps
- http://online-judge.uva.es/p/v1/165.html
- 108: Maximum Sum
- http://online-judge.uva.es/p/v1/108.html
- http://icpcres.ecs.baylor.edu/onlinejudge/index.php?option=com_onlinejudge&Itemid=8&category=25&page=show_problem&problem=2389

Exercises

- ACM ICPC 2001 Taejon
  - Problem D, Human Gene Functions
  - Problem F, Moving Tables

    http://icpc.baylor.edu/past/icpc2002/regionals/Taejon01/problems.html
- ACM ICPC 1997 World Final
  - Problem B, Jill Rides Again

    http://icpc.baylor.edu/past/icpc97/Finals/ProblemsF97.pdf
- ACM ICPC 1998 World Final
  - Problem B, Flight Planning

    http://icpc.baylor.edu/past/icpc98/Finals/Report/Problems/Problems98.pdf

References

- Books
  - **Introduction to Algorithms (Ch 16)**
    by Thomas H.C., Charles E.L., Ronald L.R.
  - **Data Structure and Algorithm Analysis (Ch 10.3)**
    by Mark Allen Weiss
- Web
  - **Dynamic Programming**
    http://www.csc.liv.ac.uk/~ped/teachadmin/algor/dyprog.html
  - **A Tutorial on Dynamic Programming**
    http://mat.gsia.cmu.edu/classes/dynamic/dynamic.html
  - **Dynamic Programming in DARWIN**
    http://www.inf.ethz.ch/personal/cgina//courses/compbio/uebung/demo3.html
  - **Dynamic Optimization Notes Index**
    http://agecon2.tamu.edu/people/faculty/woodward-richard/637/notes/default.htm