HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

DATA STRUCTURES AND ALGORITHMS

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

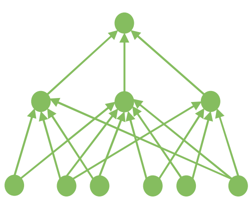
DATA STRUCTURES AND ALGORITHMS

Week 6: Dynamic programming

ONE LOVE. ONE FUTURE.

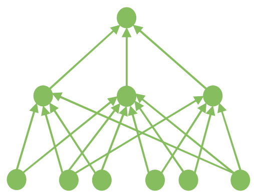
INTRODUCTION

- Dynamic programming was invented by Bellman during World War II. The first name of this algorithm was multi-stage decision process (decision making through many stages).
- The dynamic programming algorithm is a powerful technique for solving optimization problems by dividing them into smaller problems and solving the sub-problems only once.
- Dynamic programming algorithms have many similarities with backtracking and divide and conquer algorithms.



ALGORITHM DIAGRAM

- DIVIDE the starting problem into subproblems that are not necessarily independent of each other
- SOLVE sub-problems from small to large, the solutions are stored in memory (to ensure each problem is only solved correctly once)
  - The smallest subproblem must be solved in a direct, simple way
- COMBINE the solution of the larger problem from the existing solutions of the smaller sub-problems (need to use a recurrence formula)
  - The number of subproblems needed is bounded by a polynomial function of the input data size



ALGORITHM DIAGRAM

```
map<problem, value> Memory;

value DP(problem P) {
    if (is_base_case(P))
        return base_case_value(P);

    if (Memory.find(P) != Memory.end())
        return Memory[P];

    value result = some value;
    for (problem Q in subproblems(P))
        result = Combine(result, DP(Q));

    Memory[P] = result;
    return result;
}
```

Memory, mapping of programs and solutions

The smallest subproblem, corresponding to the basic step of the recursive algorithm

Always check the memory to see if the sub-problem has been solved. Once it have been solved it, don't solve it again. If it haven't been solved it, try solving it.

Solve a subproblem

Solve all subproblems and combine the solutions found to form the solution to the larger problem

Save the subproblem results in memory

ALGORITHM DIAGRAM

```
map<problem, value> Memory;

value DP(problem P) {
    if (is_base_case(P))
        return base_case_value(P);

    if (Memory.find(P) != Memory.end())
        return Memory[P];

    value result = some value;
    for (problem Q in subproblems(P))
        result = Combine(result, DP(Q));

    Memory[P] = result;
    return result;
}
```


AGORITHM DIAGRAM

```
map<problem, value> Memory;

value DP(problem P) {
    if (is_base_case(P))
        return base_case_value(P);

    if (Memory.find(P) != Memory.end())
        return Memory[P];

    value result = some value;
    for (problem Q in subproblems(P))
        result = Combine(result, DP(Q));

    Memory[P] = result;
    return result;
}
```

Memory, mapping of programs and solutions

The smallest subproblem, corresponding to the basic step of the recursive algorithm

Always check the memory to see if the sub-problem has been solved. Once it have been solved it, don't solve it again. If it haven't been solved it, try solving it.

Solve a subproblem

Solve all subproblems and combine the solutions found to form the solution to the larger problem

Save the subproblem results in memory

CHARACTERISTICS

- Compare to the divide and conquer algorithm, the dynamic programming algorithm also has 3 steps: Divide, Solve subproblems and Combine. However, in divide and conquer, the subproblems are independent; In dynamic programming, subproblems overlap or overlap.
- The biggest difficulty in proposing dynamic programming algorithms is the Recursive Formula (other names are Dynamic Programming Formula, Recursive Formula)
- There are 2 approaches: Top-Down and Bottom-Up, in which Top-Down is natural and easy to understand and easy to install.
- Memory design greatly affects the speed of the algorithm
- Memory is still used to trace and explicitly find the optimal solution

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- **Problem**: Given a sequence of n integers $(a_1, a_2, \ldots, a_n)$, find a subsequence consisting of consecutive elements of the sequence such that the sum of the selected elements is maximized.
- **Example**: Given a sequence of 7 intergers:

-16	7	-3	0	-1	5	-4

Subsequence with sum of -12 — **Subsequence with sum of 0**

  - *Optimal solution: (subsequence with the largest sum of 8): 7, -3, 0, -1, 5*

- The exhaustive algorithm has complexity $O(n^2)$, can we do better with the divide and conquer algorithm?

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- **Problem**: Given a sequence of n integers $(a_1, a_2, \ldots, a_n)$, find a subsequence consisting of consecutive elements of the sequence such that the sum of the selected elements is maximized.
- **Determine the subproblems**: P_i is the problem of finding a sub-segment consisting of consecutive elements with the largest sum whose last element is a_i, for all i = 1,..., n.

Solution of P_2 — *Solution of P_6*

-16	7	-3	0	-1	5	-4

Solution of P_4 — *Solution of P_7*

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- **Problem**: Given a sequence of n integers $(a_1, a_2, \ldots, a_n)$, find a subsequence consisting of consecutive elements of the sequence such that the sum of the selected elements is maximized.
- **Determine the subproblems**: P_i is the problem of finding a sub-segment consisting of consecutive elements with the largest sum whose last element is a_i, for all i = 1,..., n.
- **Dynamic programming formula** (formula combining solutions to sub-problems to obtain solution to parent problem): Let S_i be the sum of elements of the solutions of $P_i, \forall\, i = 1, \ldots, n$.

We have: $S_1 = a_1$,

$$S_i = \begin{cases} s_{i-1} + a_i \ if \ s_{i-1} > 0 \\ a_i \ if \ s_{i-1} \le 0 \end{cases}$$

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- **Example:**

-16	7	-3	0	-1	5	-4

$\boldsymbol{S_1 = -16, S_2 = a_2 = 7, S_3 = S_2 + a_3 = 4, S_4 = S_3 + 0 = 4,}$

$\boldsymbol{S_5 = S_4 + (-1) = 3, S_6 = S_5 + 5 = 8, S_7 = S_6 + (-4) = 4}$

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- **Dynamic programming formula** (formula combining solutions to sub-problems to obtain solution to parent problem): Let S_i be the sum of elements of the solutions of $P_i, \forall\, i = 1, \dots, n$.

We have: $S_1 = a_1$,

$$S_i = \begin{cases} s_{i-1} + a_i \text{ if } s_{i-1} > 0 \\ a_i \text{ if } s_{i-1} \le 0 \end{cases}$$

- **Solution:** The sum of the elements of the sub-segment including consecutive elements of the sequence with the largest sum of selected elements is:

$$\boldsymbol{max(S_1, S_2, \dots, S_n)}$$

- **Complexity:** O(n)

EXAMPLE: SUBSEQUENCE WITH THE LARGEST SUM

- The sum of the elements of the sub-segment including consecutive elements of the sequence with the largest sum of selected elements is:

$$\boldsymbol{max(S_1, S_2, \dots, S_n)}$$

Example:

-16	7	-3	0	-1	5	-4

$\boldsymbol{S_1 = -16, S_2 = a_2 = 7, S_3 = S_2 + a_3 = 4, S_4 = S_3 + 0 = 4,}$

$\boldsymbol{S_5 = S_4 + (-1) = 3, S_6 = S_5 + 5 = 8, S_7 = S_6 + (-4) = 4}$

Solution: $\boldsymbol{max(S_1, S_2, \dots, S_7) = 8}$

EXERCISE: LONGEST INCREASING SUBSEQUENCE

- **Problem:** Given a list of integers $A = (a_1, a_2, \dots, a_n)$ satisfy the condition that the elements are pairwise different $(a_i \neq a_j, \forall i \neq j)$. A subsequence of A is a sequence obtained by deleting some elements in A. A subsequence $B = (b_1, \dots, b_k)$ is called tight increase when $b_i < b_{i+1}, \forall i \in \{1, \dots, k-1\}$. Find the length of the strongly increasing subsequence of A that has the greatest length.
- **Homework:** Submit code to the online system

EXAMPLE: LONGEST COMMON SUBSEQUENCE

- **Problem**: Given 2 lists of characters $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m\}$. A subsequence of A is a sequence obtained by deleting some elements in A. Find the length of the longest common subsequence of X and Y.
- **Example**:
  - $X = "abcb"$ and $Y = "bdcab"$
  - The longest common subsequence is "bcb" with length of 3
- Comment: The exhaustive algorithm, comparing all subsequences of X and Y will have complexity O(2^n×2^m×max(m,n)). Can we solve this problem faster with a dynamic programming algorithm?

EXAMPLE: LONGEST COMMON SUBSEQUENCE

- **Problem**: Given 2 lists of characters $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m\}$. A subsequence of A is a sequence obtained by deleting some elements from A. Find the length of the longest common subsequence of X and Y.
- **Determine the subproblem:** Let S(i,j) be the length of the longest common subsequence of the two sequences, subsequences of X are $X_i = (x_1, \ldots, x_i)$ with $i \in \{1, \ldots, n\}$ and subsequences of Y are $Y_j = (y_1, \ldots, y_j)$ with $j \in \{1, \ldots, m\}$.
- **Base problems (smallest subproblems):**

$$S(i,0) = 0, \forall i \in \{1, \ldots, n\}$$
$$S(0,j) = 0, \forall j \in \{1, \ldots, m\}$$

EXAMPLE: LONGEST COMMON SUBSEQUENCE

- **Determine the subproblem:** Let S(i,j) be the length of the longest common subsequence of the two sequences, subsequences of X are $X_i = (x_1, \ldots, x_i)$ with $i \in \{1, \ldots, n\}$ and subsequences of Y are $Y_j = (y_1, \ldots, y_j)$ with $j \in \{1, \ldots, m\}$.
- **Base problems (smallest subproblems)**:

$$S(i,0) = 0, \forall i \in \{1, \ldots, n\}$$
$$S(0,j) = 0, \forall j \in \{1, \ldots, m\}$$

- **Dynamic programming formula:**

$$S(i,j) = max \begin{cases} S(i-1,j-1)\ if\ \ x_i = y_j \\ S(i-1,j) \\ S(i,j-1) \end{cases}$$

EXAMPLE: LONGEST COMMON SUBSEQUENCE

- Dynamic programming formula

$$S(i,j) = max \begin{cases} S(i-1,j-1)\ if\ \ x_i = y_j \\ S(i-1,j) \\ S(i,j-1) \end{cases}$$

X	3	7	2	5	1	4	9			
Y	4	3	2	3	6	1	5	4	9	7

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1	1	2
3	0	1	2	2	2	2	2	2	2	2
4	0	1	2	2	2	2	3	3	3	3
5	0	1	2	2	2	3	3	3	3	3
6	1	1	2	2	2	3	3	4	4	4
7	1	1	2	2	2	3	3	4	5	5

EXAMPLE: LONGEST COMMON SUBSEQUENCE

- **Determine the subproblem:** Let S(i,j) be the length of the longest common subsequence of the two sequences, subsequences of X are $X_i = (x_1, \dots, x_i)$ with $i \in \{1, \dots, n\}$ and subsequences of Y are $Y_j = (y_1, \dots, y_j)$ with $j \in \{1, \dots, m\}$.
- **Base problems (smallest subproblems):**

$$S(i,0) = 0, \forall i \in \{1, \dots, n\}$$
$$S(0,j) = 0, \forall j \in \{1, \dots, m\}$$

- **Dynamic programming formula:**

$$S(i,j) = max \begin{cases} S(i-1,j-1)\ if\ \ x_i = y_j \\ S(i-1,j) \\ S(i,j-1) \end{cases}$$

- **Complexity:** $O(n \times m)$

EXERCISE: LONGEST INCREASING SUBSEQUENCE

- **Problem**: Given a sequence $A = (a_1, a_2, \dots, an)$. A subsequence of the sequence A is a sequence obtained by deleting some elements from A. Find the length of the subsequence of A that is a progression with a step of 1 and has the largest length.
- **Homework**: Submit code to the online system

HUST

THANK YOU !

hust.edu.vn fb.com/dhbkhn