Discussion 12: Dynamic Programming II

(CS101 Fall 2024)

CS101 Course Team

December 23, 2024

DP Problems

- Structure of States(Subproblems).
  - Euclidean (1D,2D,...) OPT(i) (e.g. Weighted Interval Scheduling, Maximum Subarray), OPT(i,j) (e.g. Knapsack, Min Cost Refueling, LCS)
  - Tree/DAG (e.g. Critical Path, not required in CS101)
  - Subsets
- Structure of Transitions.
  - Linear Choice (e.g. Weighted Interval Scheduling, Knapsack, LCS)
  - Multiple Choices (e.g. House Coloring, Odd Numbers Coin Changing)
  - Interval Choices (e.g. Segmented Least Squares, Max Power)
- States and subproblems.
  - Exactly subproblem: (e.g. Exact Knapsack)
  - Intervals of subproblems: (e.g. Knapsack, LCS)

A Deprecated Problem

Given an array *a* of *n* integer (positive or negative).

- To find a contiguous sub-array whose sum is maximum, we can define a sub-problem for this question:
  Let $OPT_1(i)$ be the maximum sum of any sub-array of x whose **rightmost** index is *i*.
  Give your Bellman equation to solve the sub-problems(OPT_1).

A Deprecated Problem

Given an array *a* of *n* integer (positive or negative).

- To find a contiguous sub-array whose sum is maximum, we can define a sub-problem for this question:
  Let $OPT_1(i)$ be the maximum sum of any sub-array of x whose **rightmost** index is *i*.
  Give your Bellman equation to solve the sub-problems(OPT_1).

Solution

$$OPT_1(i) = \begin{cases} a_i & \text{if } i = 1 \\ max\{a_i, a_i + OPT_1(i-1)\} & \text{if } i > 1 \end{cases}$$

A Deprecated Problem

Given an array *a* of *n* integer (positive or negative).

- We can also define sub-problem:
  Let $OPT_2(i)$ be the maximum sum of any sub-array of x whose **leftmost** index is *i*.
  Give your Bellman equation to solve the sub-problems(OPT_2).

A Deprecated Problem

Given an array *a* of *n* integer (positive or negative).

- We can also define sub-problem:
  Let $OPT_2(i)$ be the maximum sum of any sub-array of x whose **leftmost** index is *i*.
  Give your Bellman equation to solve the sub-problems(OPT_2).

Solution

$$OPT_2(i) = \begin{cases} a_i & \text{if } i = n \\ max\{a_i, a_i + OPT_2(i+1)\} & \text{if } i < n \end{cases}$$

A Deprecated Problem

Given an array *a* of *n* integer (positive or negative).

- If the goal is to find two contiguous (not overlapped) sub-arrays whose sum is maximum. For example: $a[1\dots n] = [12, 5, -1, 31, -61, \underline{59, 26, -53, 58, 97}, -93, -23, \underline{84}, -15, 6]$. What is the answer to this question?

A Deprecated Problem

Given an array a of n integer (positive or negative).

- If the goal is to find two contiguous (not overlapped) sub-arrays whose sum is maximum. For example: $a[1 \dots n] = [12, 5, -1, 31, -61, \underline{59, 26, -53, 58, 97}, -93, -23, \underline{84}, -15, 6]$. What is the answer to this question?

Solution

you can use **both** OPT_1 and OPT_2

$\max\{OPT_1(i) + OPT_2(j) \mid 1 \leq i < j \leq n\}$

Counting

For the 01-Knapsack problem, **Giveout the number of options for the optimal method.**

$$f(i,j) = \max\{f(i-1, j-w_i) + v_i, f(i-1, j)\}$$

Counting

For the 01-Knapsack problem, **Giveout the number of options for the optimal method.**

$$f(i,j) = \max\{f(i-1, j-w_i) + v_i, f(i-1, j)\}$$

- If $f(i,j)$ update with new element:

$$cnt(i,j) \leftarrow cnt(i-1, j-w_i)$$

Counting

For the 01-Knapsack problem, **Giveout the number of options for the optimal method.**

$$f(i,j) = \max\{f(i-1, j-w_i) + v_i, f(i-1,j)\}$$

- If $f(i,j)$ update with new element:

$$cnt(i,j) \leftarrow cnt(i-1, j-w_i)$$

- If $f(i,j)$ not update with new element:

$$cnt(i,j) \leftarrow cnt(i-1,j)$$

Counting

For the 01-Knapsack problem, **Giveout the number of options for the optimal method.**

$$f(i,j) = \max\{f(i-1, j-w_i) + v_i, f(i-1, j)\}$$

- If $f(i,j)$ update with new element:

$$cnt(i,j) \leftarrow cnt(i-1, j-w_i)$$

- If $f(i,j)$ not update with new element:

$$cnt(i,j) \leftarrow cnt(i-1, j)$$

- If the optimal value is same:

$$cnt(i,j) \leftarrow cnt(i-1, j) + cnt(i-1, j-w_i)$$

Drop an egg

You are given some identical eggs and a building. You need to figure out the maximum floor l that you can drop them from without breaking them. Each egg will break if dropped from a floor greater than or equal to l, and will never break when dropped from a floor less than l. Note that once an egg breaks, you cannot use it anymore.

Drop an egg

You are given some identical eggs and a building. You need to figure out the maximum floor l that you can drop them from without breaking them. Each egg will break if dropped from a floor greater than or equal to l, and will never break when dropped from a floor less than l. Note that once an egg breaks, you cannot use it anymore.

- If you are given only one egg and a 100-story building, what is the strategy to figure out what l is, no matter what value l is? What is the maximum number of drops in the strategy?

Drop an egg

You are given some identical eggs and a building. You need to figure out the maximum floor l that you can drop them from without breaking them. Each egg will break if dropped from a floor greater than or equal to l, and will never break when dropped from a floor less than l. Note that once an egg breaks, you cannot use it anymore.

- If you are given only one egg and a 100-story building, what is the strategy to figure out what l is, no matter what value l is? What is the maximum number of drops in the strategy?

Solution

Drop from the first floor until the egg is broken.

Drop an egg

You are given some identical eggs and a building. You need to figure out the maximum floor l that you can drop them from without breaking them. Each egg will break if dropped from a floor greater than or equal to l, and will never break when dropped from a floor less than l. Note that once an egg breaks, you cannot use it anymore.

- If you are given two eggs and a 10-story building, what is the strategy to determine what l is, which has the minimum number of drops in worst cases?

Drop an egg

You are given some identical eggs and a building. You need to figure out the maximum floor *l* that you can drop them from without breaking them. Each egg will break if dropped from a floor greater than or equal to *l*, and will never break when dropped from a floor less than *l*. Note that once an egg breaks, you cannot use it anymore.

- If you are given two eggs and a 10-story building, what is the strategy to determine what *l* is, which has the minimum number of drops in worst cases?

Solution

First drop at floor 4, if the egg is not broken, then drop at floor 7, then floor 9. If the egg is broken, drop it from the lowest floor we do not know whether the egg will be broken.

Drop an egg

- If you are given k eggs and a n-story building, give the algorithm to figure out the method that figures out what l is with a minimum number of drops in worst cases and the minimum number of drops in worst cases using dynamic programming.

Drop an egg

- If you are given k eggs and a n-story building, give the algorithm to figure out the method that figures out what l is with a minimum number of drops in worst cases and the minimum number of drops in worst cases using dynamic programming.

Solution

Let $A(k, n)$ be the minimum number of drops in worst cases when there are k eggs and n floors. Let $P(k, n)$ be the number of floors we should first test when there are k eggs and n floors. Then the base cases are $A(k, 0) = 0$, $A(k, 1) = 1$ and $A(1, n) = n$ for any k and n. The induction step is the following.

$$A(k, n) = \min_{1 \leq j \leq n} \{\max\{A(k-1, j-1), A(k, n-j)\}\} + 1$$

$$P(k, n) = \arg\min_{1 \leq j \leq n} \{\max\{A(k-1, j-1), A(k, n-j)\} + 1\}$$

Windy Number

Given an interval $[l, r]$, find the number of integers within this range that do not contain leading zeros and have a difference of at least 2 between any two adjacent digits.

Windy Number

Given an interval $[l, r]$, find the number of integers within this range that do not contain leading zeros and have a difference of at least 2 between any two adjacent digits.

Solution

First, we simplify the problem. Let ans_i represent the count of numbers satisfying the conditions in the interval $[1, i]$. The answer is then $ans_r - ans_{l-1}$.
For a number less than n, it must have a digit that is smaller than the corresponding digit in n, with all previous digits being equal to those in n.

Windy Number

Given an interval $[l, r]$, find the number of integers within this range that do not contain leading zeros and have a difference of at least 2 between any two adjacent digits.

Solution

First, we simplify the problem. Let ans_i represent the count of numbers satisfying the conditions in the interval $[1, i]$. The answer is then $ans_r - ans_{l-1}$.
For a number less than n, it must have a digit that is smaller than the corresponding digit in n, with all previous digits being equal to those in n. With this property, we define $f(i, st, op)$ to represent the count of numbers considering the i-th digit from the highest to lowest, where st is the current prefix state and op indicates if the current digits are equal $(op = 1)$ or less $(op = 0)$.

Windy Number

Solution

In this problem, the prefix state is the value of the previous digit, since the current digit must differ from the previous one by at least 2. In other contexts, this state might be the sum of prefix digits, the gcd of all prefix digits, or the remainder of the prefix mod a number.

Windy Number

Solution

In this problem, the prefix state is the value of the previous digit, since the current digit must differ from the previous one by at least 2. In other contexts, this state might be the sum of prefix digits, the gcd of all prefix digits, or the remainder of the prefix mod a number. State transition formula:

$$f(i, st, op) = \sum_{k=1}^{\text{max\_x}} f(i+1, k, \text{op} = 1 \text{ and } k = \text{max\_x}) \quad (|st - k| \geq 2)$$

Windy Number

Solution

In this problem, the prefix state is the value of the previous digit, since the current digit must differ from the previous one by at least 2. In other contexts, this state might be the sum of prefix digits, the gcd of all prefix digits, or the remainder of the prefix mod a number. State transition formula:

$$f(i, st, op) = \sum_{k=1}^{\text{max\_x}} f(i+1, k, \text{op} = 1 \text{ and } k = \text{max\_x}) \quad (|st - k| \geq 2)$$

Here, k is the next digit, and max_x is the maximum possible digit. If $op = 1$, the current digit must not exceed the corresponding digit in n; otherwise, there is no restriction.