Algorithms & Models of Computation

CS/ECE 374, Fall 2020

More Dynamic Programming

Lecture 14

Tuesday, October 13, 2020

LaTeXed: October 17, 2020 16:50

14.1
Review of dynamic programming and some new problems

What is the running time of the following?

Consider computing $f(x, y)$ by recursive function + memoization.

$$f(x, y) = \sum_{i=1}^{x+y-1} x * f(x + y - i, i - 1),$$

$$f(0, y) = y \qquad f(x, 0) = x.$$

The resulting algorithm when computing $f(n, n)$ would take:

- $O(n)$
- $O(n \log n)$
- $O(n^2)$
- $O(n^3)$
- The function is ill defined - it can not be computed.

Recipe for Dynamic Programming

1. Develop a recursive backtracking style algorithm $\mathcal{A}$ for given problem.
2. Identify structure of subproblems generated by $\mathcal{A}$ on an instance I of size n
   1. Estimate number of different subproblems generated as a function of n. Is it polynomial or exponential in n?
   2. If the number of problems is "small" (polynomial) then they typically have some "clean" structure.
3. Rewrite subproblems in a compact fashion.
4. Rewrite recursive algorithm in terms of notation for subproblems.
5. Convert to iterative algorithm by bottom up evaluation in an appropriate order.
6. Optimize further with data structures and/or additional ideas.

14.1.1
Is in L^k?

L^k

A variation

Input A string $w \in \Sigma^*$ and access to a language $L \subseteq \Sigma^*$ via function **IsStringinL**(*string* x) that decides whether x is in L, and non-negative integer k

Goal Decide if $w \in L^k$ using **IsStringinL**(*string* x) as a black box sub-routine

Example 14.1.

Suppose L is ***English*** and we have a procedure to check whether a string/word is in the ***English*** dictionary.

- Is the string "isthisanenglishsentence" in $\textit{English}^5$?
- Is the string "isthisanenglishsentence" in $\textit{English}^4$?
- Is "asinineat" in $\textit{English}^2$?
- Is "asinineat" in $\textit{English}^4$?
- Is "zibzzzad" in $\textit{English}^1$?

Recursive Solution

When is $w \in L^k$?

$k = 0$: $w \in L^k$ iff $w = \epsilon$

$k = 1$: $w \in L^k$ iff $w \in L$

$k > 1$: $w \in L^k$ if $w = uv$ with $u \in L^{k-1}$ and $v \in L$

Assume w is stored in array $A[1..n]$

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

Recursive Solution

When is $w \in L^k$?

$k = 0$: $w \in L^k$ iff $w = \epsilon$

$k = 1$: $w \in L^k$ iff $w \in L$

$k > 1$: $w \in L^k$ if $w = uv$ with $u \in L^{k-1}$ and $v \in L$

Assume w is stored in array $A[1..n]$

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

Recursive Solution

When is $w \in L^k$?

$k = 0$: $w \in L^k$ iff $w = \epsilon$

$k = 1$: $w \in L^k$ iff $w \in L$

$k > 1$: $w \in L^k$ if $w = uv$ with $u \in L^{k-1}$ and $v \in L$

Assume w is stored in array $A[1..n]$

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by **IsStringinLk**$(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2 k)$

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by **IsStringinLk**$(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2 k)$

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by $\textsf{IsStringinLk}(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2 k)$

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by **IsStringinLk**$(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2 k)$

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by $\textsf{IsStringinLk}(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2 k)$

Analysis

```
IsStringinLk(A[1...i], k):
    if k = 0 and i = 0 then return YES
    if k = 0 then return NO   // i > 0
    if k = 1 then
        return IsStringinL(A[1...i])

    for ℓ = 1...i − 1 do
        if IsStringinLk(A[1...ℓ], k − 1) and IsStringinL(A[ℓ + 1...i]) then
            return YES

    return NO
```

- How many distinct sub-problems are generated by $\textsf{IsStringinLk}(A[1..n], k)$? $O(nk)$
- How much space? $O(nk)$
- Running time if we use memoization? $O(n^2k)$

Another variant

Question: What if we want to check if $w \in L^i$ for some $0 \leq i \leq k$? That is, is $w \in \cup_{i=0}^{k} L^i$?

Exercise

Definition 14.2.

A string is a palindrome if $w = w^R$.
Examples: ***I***, ***RACECAR***, ***MALAYALAM***, ***DOOFFOOD***

Problem: Given a string w find the longest subsequence of w that is a palindrome.

Example 14.3.

MAHDYNAMICPROGRAMZLETMESHOWYOUTHEM has ***MHYMRORMYHM*** as a palindromic subsequence

Exercise

Definition 14.2.

A string is a palindrome if $w = w^R$.
Examples: ***I***, ***RACECAR***, ***MALAYALAM***, ***DOOFFOOD***

Problem: Given a string w find the longest subsequence of w that is a palindrome.

Example 14.3.

MAHDYNAMICPROGRAMZLETMESHOWYOUTHEM has ***MHYMRORMYHM*** as a palindromic subsequence

Exercise

Assume $\boldsymbol{w}$ is stored in an array $\boldsymbol{A}[\boldsymbol{1..n}]$

$\boldsymbol{LPS}(\boldsymbol{A}[\boldsymbol{1..n}])$: length of longest palindromic subsequence of $\boldsymbol{A}$.

Recursive expression/code?

THE END

...

(for now)