Lecture 2021-03-16

Tuesday, 16 March, 2021 10:56

Dynamic Programming

1. Recursive Backtracking
   (which call to return?)
2. Identify overlapping recursive subproblems
3. Memoize into data structure
   (what is the evaluation order?)

Today: More examples

Text Segmentation w/ **fixed # of segments**

is $w \in L^*$? is $w \in L^k$?

Text Seg:
picked a prefix, and if prefix $\in L$, recurse on rest.
i.e. $w \in L^*$ iff $w = uv$ where $u \in L$, $v \in L^*$. (non-base cases)

w/ fixed # segments?

$w \in L^k$ iff $w = uv$ where $u \in L$, $v \in L^{k-1}$ ←

for input $w(1..n)$

Define $\text{IsStringInL}^k(i,h)$ = True if $w[i..n]$ is in L^h.

$$= \begin{cases} \text{TRUE} & \text{if } h=0,\ i>n \\ \text{FALSE} & \text{if } h=0,\ i\le n \ /\ i>n,\ h\neq 0 \\ \text{IsString}(w[i..n]) & \text{if } h=1 \\ \bigvee_{j=i}^{n} \left(\text{IsString}(w[i..j]) \wedge \text{IsStringInL}^k(j+1,h-1)\right) & \text{o.w.} \end{cases}$$

Call $\text{IsStringInL}^k(1,k)$ to solve original problem

→ $O(nk)$ distinct subproblems.

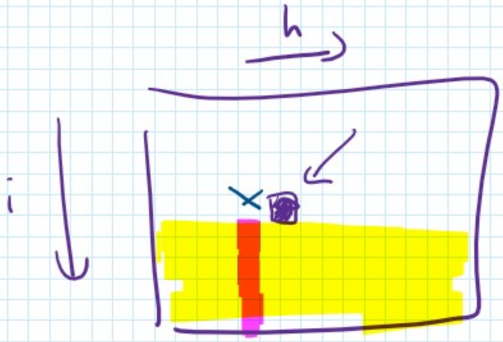
$1 \le i \le n+1$

⟹ O(nk) distinct ...

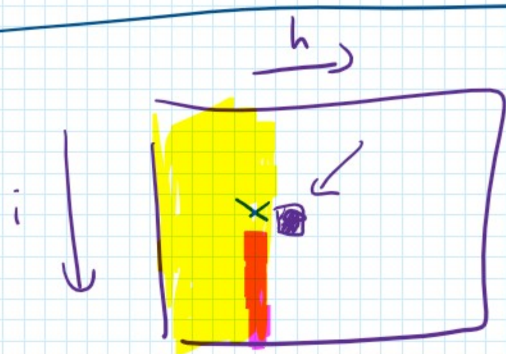
$1 \le i \le n+1$
$0 \le h \le k$.

memoize into 2d array indexed by i & h.
IsSInL^k

IsSInL^k[i][h] depends on... j > i & h−1.



one possible order:
for i from n+1 down to 1:
for h ~~from 1 to k~~: in any order?
// fill in i, h entry



another possible order:
for h from 1 to k:
for i ~~from n+1 down to 1~~ in any order?
// fill in i, h entry.

DPIsStringInL^k(w[1..n], k):

IsStrInL^k ← 2d array indexed 1.. n+1 & 0..k.

IsStrInL^k[n+1][0] ← TRUE
for i from 1 to n
IsStrInL^k[i][0] ← FALSE
for h from 1 to k:
for i from n+1 down to 1:
IsStrInL^k[i][h] ← FALSE
for j from i to n
if IsString(w[i..j]) and
IsStrInL^k[j+1][h−1]:

if IsString$(w[i..j])$ and
    IsStrInL$^k[j+1][h-1]$:
        IsStrInL$^k[i][h] \leftarrow$ TRUE
return IsStrInL$^k[1][k]$

Edit Distance

— allowed ops:

- replace char   MON → FON
- insert char   MON → MOON
- delete char   MONE → MON

MONEY
delete E! MONY
delete Y! MON
insert O MOON
replace N w/D MOOD
replace M w/F FOOD

Q: Given two strings $x[1..m]$ & $y[1..n]$
fewest # of edits to get from x to y?
"edit distance" between x & y.

Lets look at the last column in optimal edit sequence

3 cases

- $x[1..m-1]$ / $y[1..n-1]$ | $x[m]$ / $y[n]$ replace
- $x[1..m]$ / $y[1..n-1]$ | $y[n]$ insert
- $x[1..m-1]$ / $y[1..n]$ | $x[m]$ delete

Edit(i,j) = edit distance between $x[1..i]$ & $y[1..j]$

Goal: Edit(m,n)

Edit(i,j) = edit distance between x[1..i] & y[1..j]

return: Edit(m,n)

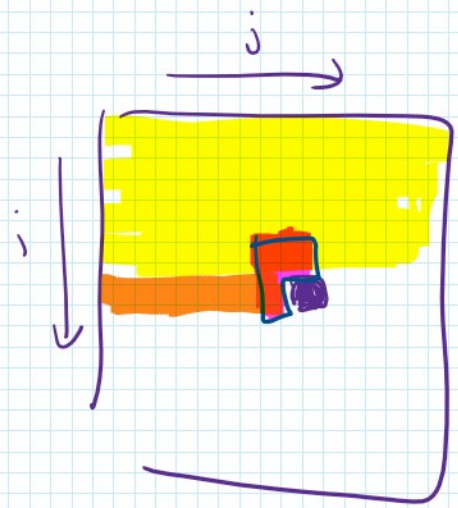
$$\text{Edit}(i,j) = \begin{cases} j \quad \textcolor{red}{(j \text{ inserts})} & i=0 \quad \textcolor{red}{(x[1..0]=\varepsilon)} \\ i \quad \textcolor{red}{(i \text{ deletes})} & j=0 \quad \textcolor{red}{(y[1..0]=\varepsilon)} \\ \min \begin{cases} \mathbb{1}[x[i] \neq y[j]] + \text{Edit}(i-1,j-1) \\ 1 + \text{Edit}(i,j-1) \\ 1 + \text{Edit}(i-1,j) \end{cases} & \text{otherwise} \end{cases}$$

$\mathbb{1}[\text{TRUE}] = 1$

$\mathbb{1}[\text{FALSE}] = 0$

\# subproblems? $O(mn)$

memoization structure? 2d array $[0..m] \times [0..n]$



one possibility:

for i from 1 to m
 for j from 1 to n
 // fill in i,j entry.