06-29

Monday, June 29, 2020 6:28 PM

① opt / counting / feasibility

② recursion design

③ run time analysis { run time / space

④ { bottom up / top down w/ memoization

- LCS
- Edit distance
- 0-1 Knapsack
- Complete knapsack

Knapsack w/ multiple constraints

- n items
- for each item i. value[i], weight[i], volumn[i]
- knapsack weight cap W volumn cap V
- What is the maximum return [$] ??

(A) 0-1 knapsack

(B) complete knapsack

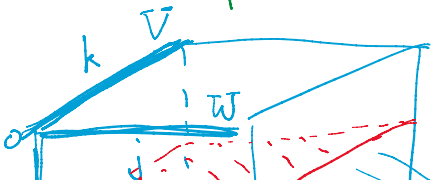
$f(i, j, k)$:= max value given item[1 ... i] and knapsack capacity is j & k (j: weight, k: volumn)

$$f[i,j,k] = \max \begin{cases} value[i] + f(i-1, j-weight[i], k-volumn[i]) \\ f(i-1, j, k) \end{cases}$$

base case $f(0, j, k) = f(i, 0, k) = f(i, j, 0) = 0$

running time ??

exp. fun.

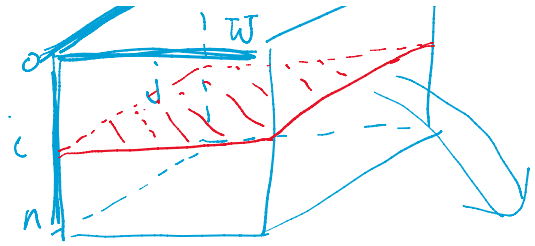


space complexity ??

(A) reduce space bottom up

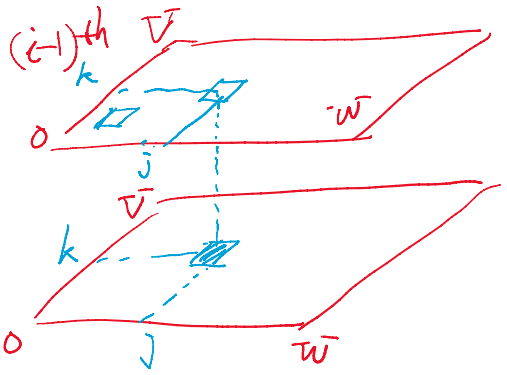
(B) cannot reduce space

$O(nWV)$ complexity



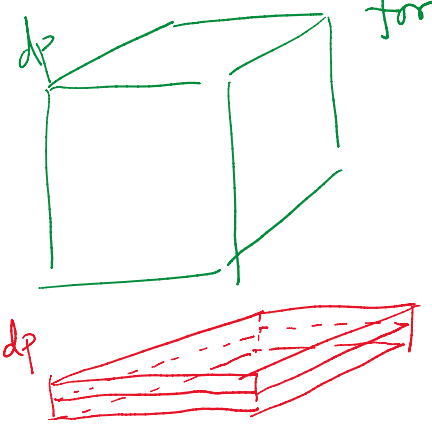
$O(nWV)$ space complexity

$f(i,j,k)$ { $f(i-1, j-\square, k-\square)$; $f(i-1, j, k)$ }



$O(nWV) \Rightarrow O(WV)$ two $O(W \cdot V)$ arrays

```
knapsack ( value[1...n], weight[1...n], volumn[1...n],
           W, V ) {
  for ( i = 1 ... n ) {
    for ( j = 1 ... W ) {
      for ( k = 1 ... V ) {
        if ( weight[i] ≤ j && volumn[i] ≤ k ) {
          dp[i,j,k] = max{ dp[i-1, j, k],
                           value[i] + dp[i-1, j-weight[i], k-volumn[i]] }
        else {
          dp[i,j,k] = dp[i-1, j, k];
        }
      }
    }
  }
  return dp[n, W, V];
}
```



(A) easy (B) okay (C) hard

matrix multiplication [a chain of matrices]

(100,2) (2,100) (100,10)

$A_1 \times A_2 \times A_3$

$A_1 \times A_2 \times A_3$

op1 $(A_1 \times A_2) \times A_3$ (100, 100)

op2 $A_1 \times (A_2 \times A_3)$ (2, 10)

— cost of a single multiplication $i\{[\quad] \times [\quad]\}j = i\{[\quad]\}j$

$O(i \cdot j \cdot k)$

op1 $100 \times 2 \times 100 = 20{,}000$

$100 \times 100 \times 10 = 100{,}000$

$120{,}000$

$2 \times 100 \times 10 = 3{,}000$

$100 \times 2 \times 10 = 2{,}000$

$4{,}000$

$A_1 \times A_2 \times \cdots \times A_n$

$(P_0, P_1)\ (P_1, P_2)\ \cdots\ (P_{n-1}, P_n)$

Q: what's min cost to multiply everything?

(P_0, P_{n-1}) (P_2, P_n)

$A_1 \times A_2 \times A_3 \times \cdots \times A_{n-1} \times A_n$

(P_0, P_{n-1}) (P_{n-1}, P_n)

which is our "last" multiplication ??

$n-1$ multiplication, which one is last one??

$$f(1,n) = \min \begin{cases} f(1,2) + f(3,n) + P_0 \times P_2 \times P_n \\ f(1,n-1) + f(n,n) + P_0 \times P_{n-1} \times P_n \\ \cdots \end{cases}$$

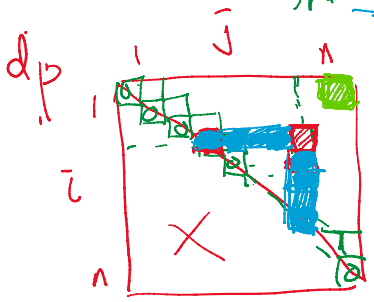
($f(n,n) = 0$)

$A_1 \times A_2 \times \cdots \times A_i \times A_{i+1} \times \cdots \times A_j \times A_{j+1} \times \cdots \times A_n$

$$f(i,j) = \min_{j-i-1} \begin{cases} f(i,i) + f(i+1,j) + P_{i-1} \times P_i \times P_j \\ f(i,i+1) + f(i+2,j) + P_{i-1} \times P_{i+1} \times P_j \\ \cdots \\ f(i,j-1) + f(j,j) + P_{i-1} \times P_{j-1} \times P_j \end{cases}$$

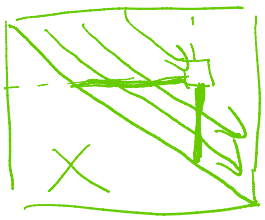
$$\left(f(i,j-1) + f(j,j) \right) + P_{i-1} \times P_{j-1} \times P_j \Big)$$

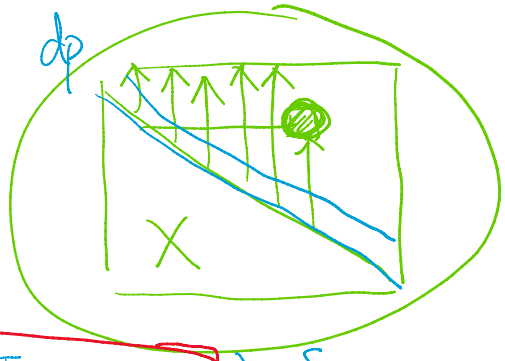
$$= \min_{k=i,\dots,j-1} \{ f(i,k) + f(k+1,j) + P_{i-1} \times P_k \times P_j \}$$



$f(i,j)$ $\quad i \le j$

$f(i,i) = 0$, $\quad f(i,i+1) = P_{i-1} \times P_i \times P_{i+1}$

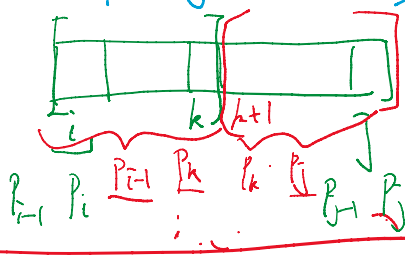





```
MEM ( P[0.... n] ) {
  [ for (i = 1... n)   dp[i,i] = 0;
    for (i = 1... n-1) dp[i,i+1] = P[i-1] · P[i] · P[i+1];

  for (j = 3 ... n) {
    for (i = j-2 .... 1) {
      dp[i,j] = ∞;
      for (k = i, ... j-1) {
        dp[i,j] = min( dp[i,j],
                       dp[i,k] + dp[k+1,j] + P[i-1] P[k] P[j] )
      }
    }
  }
  return dp[1, n];
}
```

$k = i \dots j-1$



$O(n^3)$ run time

space $O(n^2)$

string palindrome

e.g. ana

anna

hannah

....

banana — split into a list of palindromes

- [b, ana, n, a] 4
- [b, a, n, a, n, a] 6
- [b, anana] 2

Q: what's the min # of palindromes

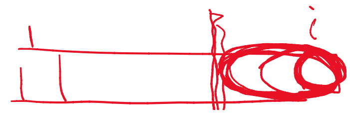
input string A 

$f(i)$:= min # of palindromes given $A[1 \ldots i]$

1	2	3	4	5	6
b	a	n	a	n	a

A (above) — "last" palindrome in the list

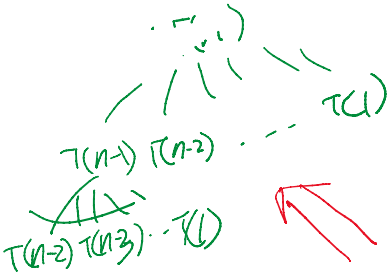
$f(6) = \min \begin{cases} f(5) + 1 & \text{"a" as last} \\ f(3) + 1 & \text{"ana" as last} \\ f(1) + 1 & \text{"anana" as last} \end{cases}$

$f(i) = \min \{$ 

$$f(i) = \min_{k=1\ldots i} \{ f(k-1) + 1 \text{ if } A[k\ldots i] \text{ is palindrome} \}$$

$f(0) = 0 \qquad f(1) = 1$


```
min_palindrome ( A[1...n]){
    dp[0]=0 ;
    for ( i=1.....n) {
        dp[i] = i ;
        for ( k= 1.....i ) {
            if ( is_palindrome(A[k:i]))
                dp[i]=min ( dp[i],
                            1+dp[k-1])
        }
    }
    return dp[n];
```

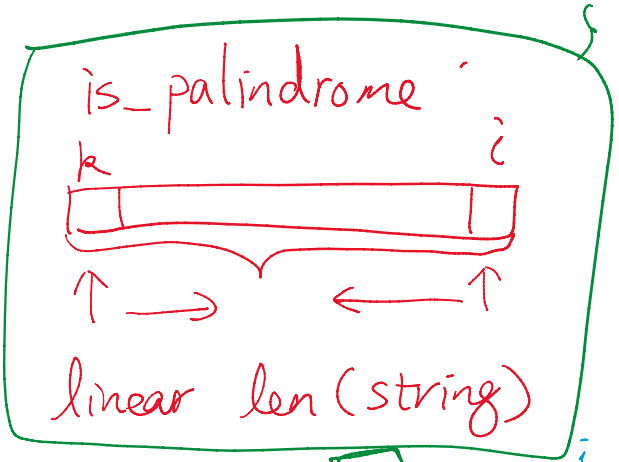


$O(n^3)$

$O(n^2)$

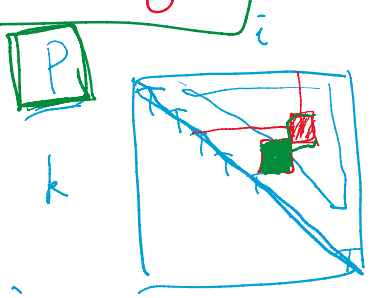
banana

b, a, n, n, a, n, a



Q: can we make it constant time operation?

A [k, i]



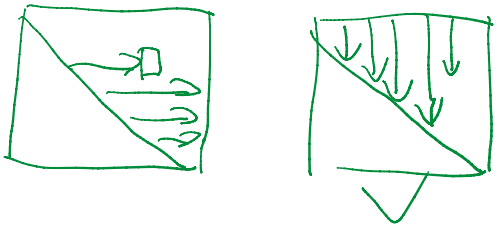
$P[k,i] = \begin{cases} \text{True} & \text{means } A[k\ldots i] \text{ is palindrome} \\ \text{False} & \text{means } A[k\ldots i] \text{ is not palindrome} \end{cases}$

$k \le i$

k

$$P[k,i] = (A[k] == A[i]) \;\&\&\; P[k+1, i-1]$$

$P[k,i] = (A[k]==A[i]) \;\&\&\; P[k+1,i-1]$



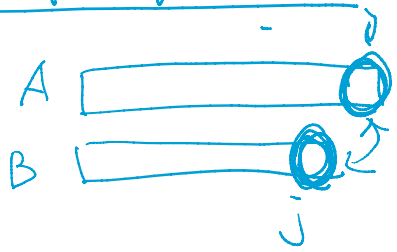
$O(n^2)$

```
for (k=1...n) P[k,k]=1;
for (k=1...n-1) P[k,k+1]=(A[k]==A[k+1])
for (i=3...n) {
    for (k=1.....i-2){
        P[k,i]= (A[k]==A[i]) &&
                P[k+1, i-1]
}
}
```

(A) easy (B) okay (C) hard

recursion structure

① pair of sequences [LCS, Edit distance]



$f(i,j)$ $f(i-1,j)$ $f(i,j-1)$ $f(i-1,j-1)$ ⋮

② single sequence [string palindrome]

$f(i)$ $f(i-1)$ $f(i-2)$ ⋮ prefixes

③ knapsack [(0-1) complete . multiple constraints]

prefixes

③ knapsack [0-1 complete, multiple constraints]

e.g. [final exam question
coin change]

④ intervals [matrix multiplication]