06-22
Monday, June 22, 2020 6:29 PM

6:30 ~ 8 pm DP
8:15 ~ 9:30 pm Quiz 2

- LCS
- 0-1 knapsack

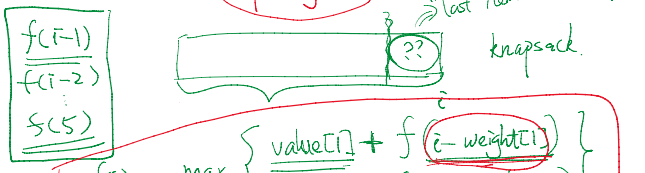
DP ① optimization / counting / feasible
② design recursion
③ [examing recursion] run time bad if imp naively
④ two approaches { bottom up / top down w/ memoization

Unbounded / complete knapsack

- n types items
- for each type of items, there is inf copies
- for each item of type i, $\frac{value[i]}{weight[i]}$
- capacity of the knapsack is C

Goal: max return

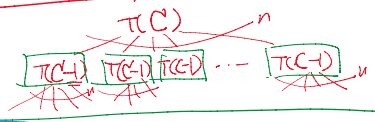
f(i) := max return given knapsack w/ capacity i

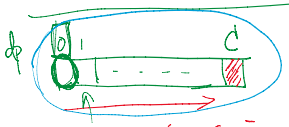


$$f(i) = \max \begin{cases} value[1] + f(i - weight[1]) \\ value[2] + f(i - weight[2]) \\ \cdots \\ value[n] + f(i - weight[n]) \end{cases}$$

base case: $f(0) = 0$, $i < \min weight[k]$

weight[i] = 1





dp[ ] initialized to 0

```
for ( i = 1 ... C ) {
    best = 0;
    for ( k = 1 .... n ) {
        if ( weight[k] ≤ i ) {
            best = max{best, value[k] + dp[i - weight[k]]}
        }
    }
    dp[i] = best;
}
return dp[C];
```

try all n types of items

O(nC)

weights
int
i = 1
<$1, 1$>

Edit Distance

- given two strings
- e.g. start = "hello"
  target = "hell"
- allowed operations
  1. insert "hello" ⇒ (insert b) "helblo"
  2. delete "hello" ⇒ (delete l) "helo"
  3. replace "hello" ⇒ (replace h w/ g) "gello"

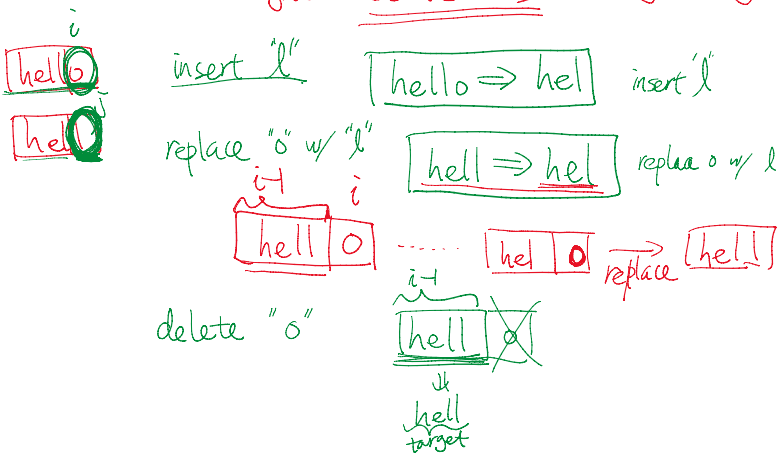
hello ⇒ ␣ ⇒ ␣ ---- ⇒ hell

hello ⇒ hellll ⇒ hell
(replace, delete)

hello ⇒ hell
(delete o)

"best" shortest seq of operations
min numbers of operations

f(i, j) := min number of operations given start[1...i] & target[1...j]



---

start = "rabbit"
target = "bunny"

① rabbit → bunn (insert y) ⇒ bunny

② rabbi → bunn (replace)
(7 ops)

f(i, j) = f(i, j-1) + 1

② rabbi → bunn f(i,j) = f(i,j-1) + 1

rabbi + → bunn + → (replace r/y) bunny

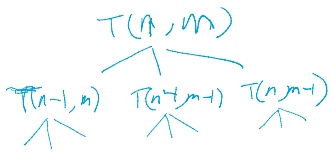
f(i,j) = f(i-1,j-1) + 1 / 0 (replace)

③ rabbi → bunny

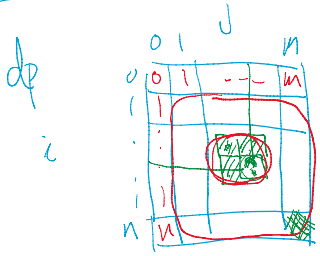
rabbi + → bunny + → (delete) bunny

f(i,j) = f(i-1,j) + 1

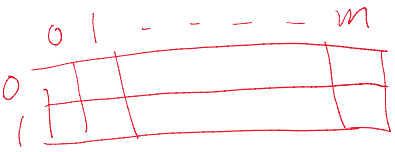
$$f(i,j) = \min \begin{cases} f(i-1,j)+1 \\ f(i,j-1)+1 \\ f(i-1,j-1)+0/1 \end{cases}$$



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




```
edit_distance (A[1...n], B[1...m]) {
    for (i=0...n) dp[i,0] = i;
    for (j=0...m) dp[0,j] = j;
    for (i=1....n) {
        for (j=1...m) {
            dp[i,j] = min( dp[i-1,j]+1,
                           dp[i,j-1]+1);
            if (A[i] == B[j])
                dp[i,j] = min ( dp[i,j],
                                dp[i-1,j-1]);
            else dp[i,j] = min (dp[i,j],
                                dp[i-1,j-1]+1);
        }
    }
    return dp[n,m];
}
```



dp[i%2, j] = min(dp[i-1%2, j]+1, dp[i%2, j-1]+1)

r.t O(nm)

space O(nm) ⇓ O(m)

Quiz start at 8:15