Dynamic Programming

these are the reasons.....

greedy ~~~ X

backtracking ~~~ X

D&C lucky or unlucky to pick breaking points

DP figures out breaking points, bottom-up

Fibonacci #'s

rec. $F(n) = F(n-1) + F(n-2)$ top → bottom

repeating a lot of work

iter. fill an array bottom → up

Same idea with DP. Find recurrence, then find an iterative version to fill an array.

↖ usually a more complex recurrence than in D&C

string reconstruction

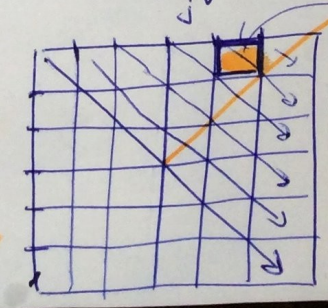
problem $S[1 \ldots n]$

rec
- subproblem $S[i \ldots j]$ is a collection of words
- dictionary of words – constant time
- on strings
- 2 cases $S[i \ldots j] \in$ Dict ← Base case
- or $\exists\, k$, s.t. $S[i \ldots k], S[k+1 \ldots j]$ are both collection of words

$i \le k \le j-1$

Array formulation

iter
- bottom-up
- figure dependencies



S[i,j]

S[1,1], S[2,j]

S[1,2], S[3,j]

S[1,3], S[4,j]

```
for d = 1 to n                          } Build from small to large strings
  for i = 1 to n - d + 1
    j = i + d - 1                                  O(n^3)
    if S[i, j] ∈ Dict                  ↑ O(n^2) lookups in dictionary
      A[i, j] = T
    for k = i to j - 1
      if A[i, k] = T and A[k+1, j] = T
        A[i, j] = T
```

info to keep for reconstruction : keep k, the breaking point

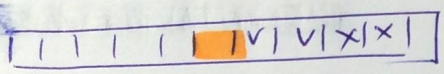
Correctness induction on d

Set up a better recursion, 2D to 1D array

$S[i, n]$ is a collection of words ?

rec. on strings: $\exists k$ $S[i, k] \in$ Dict and $S[k+1, n]$ is a collection of words, $i \le k \le n-1$

iter



or in the opposite direction

$O(n^2)$ ← not tight

$O(n \cdot L)$ L = length of the longest word in Dict

info to keep for reconstruction : keep breaking point k or keep all k's for all possible ways

Edit Distance

how close are two sequences

Hamming distance

1 1 0 1 1 1 0
0 1 0 1 1 0 0

= 2

 # places where sequences do not match

bad for shifts/insertions

1 1 0 1 1 1 0 4
1 0 1 1 1 0 _

edit distance **# of edit operations to transform one string to another**

ops	costs
– insert	c_i
– delete	c_d
– replace	c_r

ex activ ate
ca veat

ops D RD I D

5 ops, if all costs are 1 then edit distance is 5

A[1 ... n]
B[1 ... m]

subproblem A[i ... j] → B[k ... l] NO

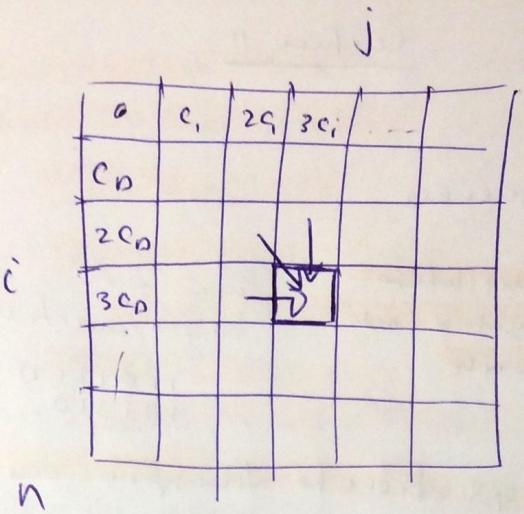
A[1 ... i] → B[1 ... j]

ex: activ A[1 5]
cave B[1 4]

$$C(i,j) = \min \begin{cases} C(i-1,j-1) & \text{if } A[i] = B[j] \text{ (match)} \\ C(i-1,j-1) + C_R & \text{if } A[i] \neq B[j] \\ C(i,j-1) + C_I \\ C(i-1,j) + C_D \end{cases}$$

rec on cost function

iter



array $O(nm)$

$O(1)$ to fill a cell

$\Rightarrow O(nm)$

to reconstruct keep info of pointer that provide min in the recurrence

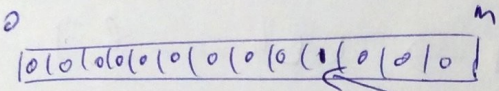
---

Review

Bloom filter data structure

correctness vs. space

allow false positives, in return get small space



$x_1, x_2 \ldots x_n \quad h_i(x_j)$

$h_1, h_2 \ldots h_k$

hash into the same array or into separate arrays

↳ asymptotically the same

$$k = (\ln 2)\frac{m}{n}$$

optimum