DYNAMIC PROGRAMMING (DP)

① define subproblems

② derive recursive formula to express opt value ans to subproblem in terms of answers to smaller subproblems

③ evaluate formula bottom-up using a table

Line Break Problem

$\langle 3,3,2,5,10\rangle$, $L=11$
↳ $\langle 3,3,2\rangle$, $\langle 5\rangle$, $\langle 10\rangle$

Given sequence $\langle a_1,\dots,a_n\rangle$ and L,
split into subseqs $\langle a_1,\dots a_{i_1}\rangle$, $\langle a_{i_1+1},\dots a_{i_2}\rangle$,
$\dots$, $\langle a_{i_{k-1}+1},\dots,a_n\rangle$,
$0 = i_0 < i_1 < \dots < i_{k-1} < i_k = n$,
s.t. each subseq has sum $\le L$
to minimize penalty $\sum_{j=1}^{k} \left(L - (a_{i_{j-1}+1}+\dots+a_{i_j})\right)^2$

① For each $i=0,\dots,n$,
define $P(i)$ = min penalty for the input sequence $\langle a_1,\dots,a_i\rangle$

Want $P(n)$.

② Recursive formula:
if last subseq we split into $\langle a_{j+1},\dots,a_i\rangle$,
then $P(i) = P(j) + \left(L - (a_{j+1}+\dots+a_i)\right)^2$

but don't know j, so try all & take min

$$\Rightarrow P(i) = \min_{\substack{j\in\{0,\dots,i-1\}:\\ a_{j+1}+\dots+a_i\le L}} \left(P(j) + \left(L - (a_{j+1}+\dots+a_i)\right)^2 \right)$$

Base case: $P(0) = 0$.

Implement recursively.

base case: $T(0)$

If we evaluate formula recursively,

$$T(i) = T(i-1) + T(i-2) + \cdots + T(0)$$

$\Rightarrow$ exponential!

Instead, evaluate in increas. i & use table

Pseudocode:

$s_0 = 0$
for $i = 1$ to n
  $s_i = s_{i-1} + a_i$

$P[0] = 0$
for $i = 1$ to n
  $P[i] = \min_{j \in \{0,\ldots,i-1\}:\ a_{j+1}+\cdots+a_i \le L} \left(P[j] + \left(L - (a_{j+1}+\cdots+a_i) \right)^2 \right)$

$(s_i - s_j)^2$

$(a_1+\cdots+a_i) - (a_1+\cdots+a_j)$

$pred[i] = j$ that attains the above min

return $P[n]$.

$\Rightarrow O(n^3)$ time

improves to $O(n^2)$ by using prefix sums

$O(n)$ space

How to output opt sol'n:

```
output-sol(i):
    if i=0 return
    j = pred[i], output-sol(j)
    output < a_{j+1}, ..., a_i >.

Call output-sol(n).
```

additional $O(n)$ time

Alternative 1: "forward" version

for $i = 1, \ldots, n+1$,
  define $P(i)$ = min penalty for sequence $\langle a_i, \ldots, a_n \rangle$

for ...

define $P(i)$ = min penalty for sequence $\langle a_i, \ldots, a_n \rangle$

Want $P(1)$.

$$P(i) = \min_{\substack{j \in \{i+1, \ldots, n+1\} \\ a_i + \ldots + a_{j-1} \le L}} \left(P(j) + \left(L - (a_i + \ldots + a_{j-1}) \right)^2 \right)$$

Alternative 2: graph version ...

define graph where

"states" → vertices are $i \in \{1, \ldots, n+1\}$

"transitions" → place edge $i \to j$ of weight $\left(L - (a_i + \ldots + a_{j-1})\right)^2$ if $a_i + \ldots + a_{j-1} \le L$, $j > i$.

Find shortest path from 1 to $n+1$.

DAG

Longest Common Subsequence (LCS) Problem

Given 2 strings $a = a_1 a_2 \cdots a_m$

$b = b_1 b_2 \cdots b_n$

find longest string that is a subseq of both a & b

e.g. ~~0~~ 1 0 ~~3~~ ~~2~~ 1

1 0 3 2 0 1 0 2

1 0 3 2 1

(appl'ns: compare strings/files/DNAs

min # char deletes + # char inserts "edit distance")

Define subproblems: for $i = 0, \ldots, m$, $j = 0, \ldots, n$,

let $C(i,j)$ = length of LCS of $a_1 \cdots a_i$ & $b_1 \cdots b_j$

Want $C(m,n)$.

Base cases. $C(i,0) = 0 \quad \forall i$

$C(0,j) = 0 \quad \forall j$

Recursive formula:

Case 1. soln not use a_i

$\Rightarrow C(i,j) = C(i-1,j)$.

Case 2. soln not use b_j

$\Rightarrow C(i,j) = C(i,j-1)$

Case 3. soln uses a_i & uses b_j with $a_i = b_j$.

$\Rightarrow C(i,j) = C(i-1,j-1) + 1$

but don't know which case,
so try all & take max

$$\Rightarrow C(i,j) = \begin{cases} \max\{C(i-1,j), C(i,j-1), C(i-1,j-1)+1\} & \text{if } a_i = b_j \\ \max\{C(i-1,j), C(i,j-1)\} & \text{if } a_i \neq b_j \end{cases}$$

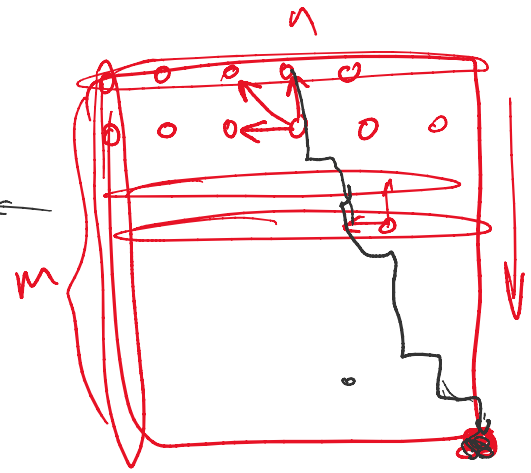
Evaluation order: { increas. order of i
same i: increas order of j

for i = 1 to m
  for j = 1 to n
    ~ ~

table entries = $O(mn)$
time per entry = $O(1)$

$\Rightarrow$ $\boxed{O(mn)}$ time

$O(mn)$ space



Rmk - can improve space to $O(n)$
by remembering only last 2 rows

How to output opt soln?

```
output-sol(i,j):
  if i=0 or j=0 return
  if C[i,j] = C[i-1,j-1]+1 & a_i=b_j
      output-sol(i-1,j-1), output a_i
  else if C[i,j] = C[i-1,j]
      output-sol(i-1,j)
  else output-sol(i,j-1)
call output-sol(m,n)
```

$O(m+n)$ time

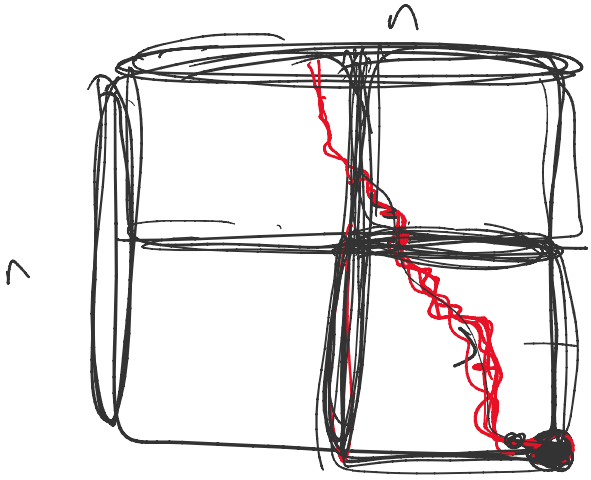
but space goes back up to $O(mn)$
unfortunately!

Space improvement: (Hirschberg '75 / Ramachandran '06)

Space Improvement: (Hirschberg '75 / Chowdhury, Ramachandran '06)

Suppose $m = n$.

idea - divide & conquer!



DP

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n^2)$$

$$\Rightarrow T(n) = O(n^2)$$

$$S(n) = S\left(\frac{n}{2}\right) + O(n)$$

$$\Rightarrow S(n) = O(n)$$