DYNAMIC PROGRAMMING (DP)

- define subproblems
- derive recursive formula to express ans to subproblem in terms of ans to smaller subproblems
- evaluate formula bottom-up using a table

"Line Break" Problem

"if is a good algorithm."

→ "if is a
good
algorithm."

$\underbrace{\quad\quad}_{L}$

$\langle 3, 3, 2, | 5, 11 \rangle$

$P(5) = P(4) + f(L-11)$

$\cancel{P(5) = P(3) + f(L-5-11)}$

Given sequence of numbers $\langle a_1, .., a_n \rangle$ and L,

split into contiguous subsequences

$\langle a_1, \ldots, a_{i_1} \rangle, \langle a_{i_1+1}, .., a_{i_2} \rangle, .., \langle a_{i_{k-1}+1}, .., a_n \rangle$

$0 = i_0 < i_1 < i_2 < \cdots < \cdots i_{k-1} < i_k = n.$

s.t. $a_{i_{j-1}+1} + \ldots + a_{i_j} \leq L \quad \forall j = 1, .., k.$

to minimizing penalty $\sum_{j=1}^{k} f(L - a_{i_{j-1}+1} - \cdots - a_{i_j})$

e.g. $f(x) = x^2$. ←

- Define subproblems: ⇐

For each $i = 0, .., n$,

define $P(i)$ = min penalty for the input sequence $\langle a_1, \ldots, a_i \rangle$

Want $P(n)$.

- Recursive formula: To solve $P(i)$:

if last subsequence we split into $\langle a_{j+1}, \ldots, a_i \rangle$ ←

- Recursive formula: to solve $P(i)$:

if last subsequence we split into $\langle a_{j+1}, \dots, a_i \rangle$

then $P(i) = P(j) + f(L - a_{j+1} - \dots - a_i)$.

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

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

if we evaluate formula recursively,

runtime $T(i) \geq T(i-1) + T(i-2) + \dots$

$\Rightarrow$ exponential!

Instead, evaluate in increas. i & use table.

Pseudocode:

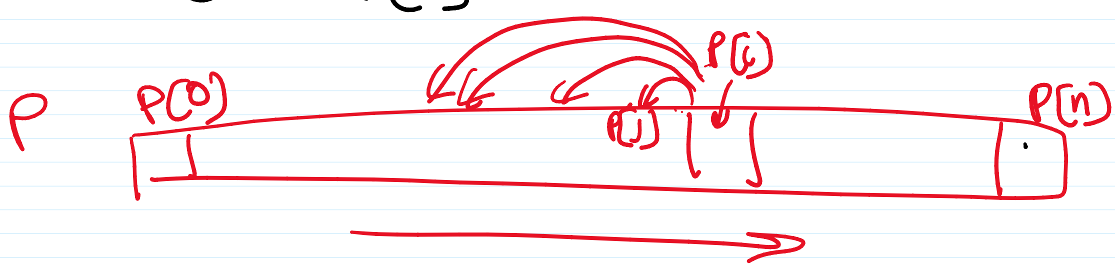
$P[0] = 0$

for $i = 1$ to n do

$P[i] = \min_{\substack{j \in \{0,\dots,i-1\}: \\ a_{j+1}+\dots+a_i \leq L}} \left(P[j] + f(L - a_{j+1} - \dots - a_i) \right)$

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

return $P[n]$



runtime $O(n \cdot \underbrace{n \cdot n}) = \boxed{O(n^3)}$

#subprobs i.e. #table entries

cost to evaluate formula for one subprob.

lowered to

$O(n \cdot$ can be lowered to $O(n)$ by maintaining running sum for $a_{j+1} + \dots + a_i$ $) = \boxed{O(n^2)}$

another way: precompute $s_i = a_1 + \dots + a_i$
afterwards, $a_{j+1} + \dots + a_i = s_i - s_j$

space $\boxed{O(n)}$

How to output opt sol'n (rather than opt value):

```
output-sol(i):
    if i=0 return
    j = pred[i], output j.
    output-sol(j)
Call output-sol(n).
```

$O(n)$ additional time

Alternative 1: forward version

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

want $P(1)$.

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

evaluate decreas order of i.

$j \in \{i+1, .., n\}:$
$a_i + \cdots + a_{j-1} \le L.$

evaluate decreas order of i.

Alternative 2: graph version

define weighted dir. graph where ← a DAG (dir. acyclic graph)

Current state → vertices $i \in \{1, .., n+1\}$

place edge from i to j with weight $f(L - a_i - \cdots - a_{j-1})$ if $a_i + \cdots + a_{j-1} \le L$ & $j > i$

find shortest path from vertex 1 to $n+1$

Longest Common Subsequence Problem (LCS)

Given 2 strings $a = a_1 a_2 \cdots a_m$
$b = b_1 b_2 \cdots b_n$

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

e.g. 0 1 0 1 3 1 2 1   10312
1 0 3 2 0 1 0 2

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

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

Want $C(m,n)$.

Recursive formula:

Case 1. sol'n not use a_i: $\Rightarrow$ $C(i,j) = C(i-1,j)$
Case 2. sol'n not use b_j $\Rightarrow$ $C(i,j) = C(i,j-1)$
Case 3. sol'n uses a_i & b_j with $a_i = b_j$
$\Rightarrow$ $C(i,j) = C(i-1,j-1) + 1$

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

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