Dynamic Programming

Step 1 → Recursive Formulation of problem → count # of distinct subproblems

Step 2 → Create a table of subproblem → fill from bottom up

Running time → (size of table) · (time for each entry)

e.g. → **RNA Seq**

RNA Secondary Structure

- What you're given:
  - RNA sequence $B = b_1, b_2, \ldots, b_n$ (a sequence of ***bases***)
  - $b_i \in \{A, C, G, U\}$
- $S \subset [n]^2$ is a set of ***pairings***; $[n] = \{1, 2, \ldots, n\}$.
- The rules:
  - If $(i, j) \in S$ then $\{b_i, b_j\}$ is either $\{A, U\}$ and $\{C, G\}$.
  - No base is in more than one pair.
  - If $(i, j) \in S$ then $i < j - 4$. *(the "no sharp turns" condition)*
  - If $(i, j), (k, l) \in S$ then we cannot have $i < k < j < l$. *(the "non-crossing" condition)*

→ $b_1, b_2, \ldots b_{n-1}, b_n$

1. b_n paired with eligible $b_1, \ldots b_{n-5}$
2. b_n unpaired (then its just b_{n-1} optimal) or trivial

$\underbrace{b_1, b_2 \ldots b_{i-1}}, b_i, \underbrace{b_{i+1}, \ldots b_{n-1}}, b_n$ — distinct due to no crossing

$OPT(i,j)$ = max # of base pairings from $b_i, \ldots, b_j$

$$OPT(i,j) = \max\{opt(i, j-1),\ 1 + \max_{i \le k < j-4} \{opt(i,k-1) + opt(k+1, j-1)\}\}$$

so b_k, b_j compatible

base case $OPT(i,j) = 0$ if $j \le i+4$

Complexity → # subproblems → $O(n^2)$
time for rec → $O(n)$ ish for the max

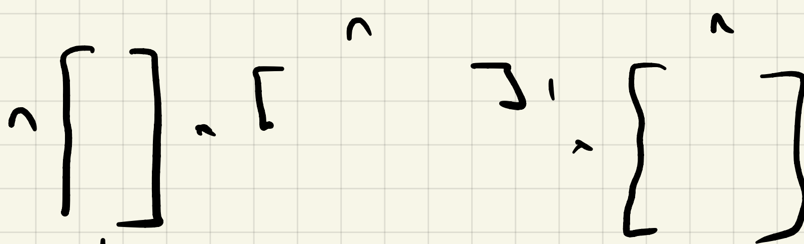
⟹ $O(n^3)$

Multiple Matrix Mult

Multiple Matrix Multiplication

- Matrix multiplication is ***associative*** but ***not commutative***
  - $A(BC) = (AB)C$
  - $AB \neq BA$ (not necessarily equal)
- Cost of mult. $m \times p$ and $p \times n$ matrices is $O(mnp)$.
- The Problem:
  - Given matrices $A_1, A_2, \ldots, A_k$
  - A_i is an $n_i \times n_{i+1}$ matrix where $n_1, \ldots, n_{k+1}$ are ints.
  - Compute $A_1 A_2 \cdots A_k$ via a sequence of matrix mults.
  - $(A_1 A_2 \cdots A_k)$ is an $n_1 \times n_{k+1}$ matrix

n^3

eg 

$(AB)C$ → n^2 then n^3 → $O(n^3)$

$A(BC)$ → n^2 then n^2 → $O(n^2)$

Strat → pick pivot

$(A_i \ldots A_k)(A_{k+1} \ldots A_j)$ min of this

→ base $OPT(i,i) = 0$

$$OPT(i,j) = \min_{i \le k \le j-1} \{OPT(i,k) + OPT(k+1,j)\} + n_i n_{k+1} n_{j+1}$$ → merge

Optimal BST Problem

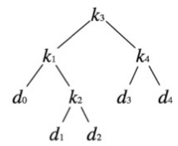
Optimal Binary Search Trees

- Input: Given sequence $(k_j)_{1 \le j \le n}$
  - Keys in sorted order: $k_1 < k_2 < \cdots < k_n$
  - p_j = probability of searching for key k_j
  - "Dummy keys" $d_0 < d_1 < \cdots < d_n$ → interval?
  - q_j = probability of searching for d_j (searching in (k_j, k_{j+1}))
  - $\Sigma_j\, p_j + \Sigma_j\, q_j = 1$.
- Problem: design an ***optimal binary search*** tree that minimizes expected search time
  - $\text{depth}_T(k_j)$ = depth of k_j in tree T
  - Search time for $k_j = \text{depth}_T(k_j) + 1$
  - Expected search time in T is…?

Example with 4 keys

- $p_1 = 0.2$ $\quad q_0 = 0.3$
- $p_2 = 0.1$ $\quad q_1 = 0.1$
- $p_3 = 0.05$ $\quad q_2 = 0.1$
- $p_4 = 0.05$ $\quad q_3 = 0.05$
- $\quad q_4 = 0.05$

$$E[\text{search time in } T] = \Sigma_j\, p_j\,(\text{depth}_T(k_j)+1) + \Sigma_j\, q_j\,(\text{depth}_T(d_j)+1)$$



$$\begin{aligned} E[\text{search time}] &= 1\cdot p_3 + 2\cdot(p_1+p_4) + 3\cdot(q_0+p_2+q_3+q_4) + 4\cdot(q_1+q_2) \\ &= 0.05 + 2\cdot(0.25) + 3\cdot(0.5) + 4\cdot(0.2) \\ &= 2.85 \end{aligned}$$

- $w(i,j)$: the total probability mass in a subtree containing $k_i, \ldots, k_j$, i.e., $q_{i-1} + p_i + \cdots + p_j + q_j$.
  - $w(i, i-1) = q_{i-1}$.
  - $w(i,j) = w(i, j-1) + p_j + q_j$.
- $e(i,j)$: expected number of nodes $\{k_i, \ldots, k_j\}$ touched in a search, if $\{k_i, \ldots, k_j\}$ are arranged optimally in a subtree.
  - $e(i, i-1) = w(i, i-1) = q_{i-1}$.
  - $e(i,j) = \min_{i \le r \le j}\{e(i, r-1) + e(r+1, j) + w(i,j)\}$ → sum of weight of subtree
  - $\text{root}(i,j)$ = the "r" minimizing the eqn. above.

In this case. Each subtree is optimal if the entire tree is optimal!