More DP

"Try all possibilities"

Recursive Alg'n → Memorize → Evaluation order ↓ Iterative alg'n

Problem: Max Weight Independent Set (MWIS) on trees.

Given:\ T = (V, E) acyclic
\ \ \ \ \ \ v \in V, \ w(v) \geq 0
Find \ S \subseteq V \text{ that maximizes } \sum_{v \in S} w(v)
\ s.t. \ \forall u, v \in S, (u, v) \notin E.

8 + 1 + 1 + 7 + 6 + 4 = 27

Observations:
- \ v = root
- either \ v \ is in opt. soln or not.
- v not in opt soln
  recurse on subhees rooted at children of v.
- v is in opt soln
  recurse on subhees rooted at grandchildren of v.
1 Define subproblems: \( V \subseteq V \)

\[ A(v) = \text{weight of MWIS in the subtree rooted at } v. \]

Ans: \( A(\text{root}) \).

2 Base case: if \( v \) is a leaf \( A(v) = w(v) \)

Otherwise:

**Recursive formula:**

\[
A(v) = \max \left\{ \sum_{\text{child } u} A(u), \ w(v) + \sum_{\text{grandchild } u} A(u) \right\}
\]

3 Evaluation Order:

in Post order traversal of vertices.

Psuedocode:

\[ m maximize \{ E_v, \ldots, E_{v_n} \} = \text{post-order traversal } T. \]
1. \( E_{V_1 \ldots V_m} = \text{post-order} \)

2. for \( i = 1 \) to \( n \)
   
3. if \( u_i \) is a leaf then \( A[u_i] = w(u_i) \)

4. else
   
   \[
   O(m) \{ A[u_i] = \max \{ \sum_{\text{child of } u_i} A[v_j], w(u_i) + \sum_{\text{grandchild of } u_i} A[v_k] \} \}
   
   5. Return \( A[u_n] \)

Run Time: \# subproblems = \( O(m) \)

Time per subproblem = \( O(n) \)

\( \Rightarrow O(m^2) \) Time.

Better Analysis: How many time \( A[u] \) appears on RHS. for a given \( u \in V \)

Twice! Once for the parent \( v \) of \( u \) and once for the grandparent of \( u \).

\( \Rightarrow O(m) \)

Amortized analysis.

Problem: Subset Sum
Problem: Subset Sum

Given numbers $a_1, a_2, \ldots, a_n$ & $T$
check if $\exists S \subseteq \{1, \ldots, n\}$ s.t. $\sum_{i \in S} a_i = T$.

Op: TRUE or FALSE.

e.g. $1, 10, 5, 15, 20, 8, 30, 13 \quad T = 22$

TRUE

1. Define subproblem: $0 \leq i \leq n, 0 \leq t \leq T$

   $SS(i,t)$ = true if $a_1, \ldots, a_i$ has a subset summing to $t$.

Base case:

$SS(0,0) = true$

$SS(0,t) = false \quad \forall t \geq 1$.

Recursive formula:

$$SS(i,t) = \begin{cases} SS(i-1,t) \lor SS(i-1,t-a_i) & \text{if } a_i \leq t \\ SS(i-1,t) & \text{otherwise} \end{cases}$$
Evaluation order:
in increasing order or i.
for each i, any order ok t.

Run Time: # subproblem: \(O(mt)\)
time per subproblem: \(O(1)\)
\[\Rightarrow \quad O(mt)\] time.
i/p size: \(O(n \cdot \log T)\)
\[T = 2^{\log T}\]

Rank: \(T\) can be very large unfortunately.

poly in \(n \cdot \log T\)?
Big open que

\(P = NP?\)

Variant:

Problem 3: Parsing:

Given a CFG \(G = (V, T, P, S)\) &
sentence \(x = a_1 \ldots a_n \in T^*\)
check if \(x \in L(G)\)?
E.g.

\[ S \rightarrow AS \mid AB \]
\[ A \rightarrow BC \]
\[ B \rightarrow SA \mid AC \mid \lambda \]
\[ C \rightarrow A \lambda \mid CC \mid \lambda \]

\[ x = (011011) \text{ ELG} \]

**CNF** 

Assume: every production rule in \( P \) is of type:

\[ A \rightarrow BC \]

\[ A \rightarrow a \]

**CKY Alg’m ( Cocke – Younger – Kassami ’70)**

1. **Define Subprob.** \( 1 \leq i \leq j \leq n, \quad A \in V \)

\[ f(i, j, A) = \text{true if } a_i \ldots a_j \text{ can be generated starting at } A. \]

**Ans:** \( f(1, n, S) \)

**Care:** \( a_i \) be generated by \( A \).
Base case: \( a_i \) be generated by \( P \):

\[
f(i, i, A) = \text{true iff } (A \rightarrow a_i) \in P.
\]

= false o.w.

Recursive formula: \( i < j \)

\[
f(i, j, A) = \frac{a_i \ldots a_j}{B} \text{ be gen. by starting at } A.
\]

Clauses:
- Which production rule to use \( A \rightarrow BC \)
- Split \( a_i \ldots a_k \frac{a_{k+1} \ldots a_j}{B} \), \( i \leq k \leq j-1 \)

\[
f(i, j, A) = \bigvee_{A \rightarrow BC} \bigvee_{K \in \{i, \ldots, j-1\}} \bigvee_{\text{im } P} \left( f(i, k, B) \land f(k+1, j, C) \right)
\]

Evaluation order:

Increasing order \( B(C) \rightarrow (j-i) \)
Run time:

- # sub problems: $O(n^2 \cdot |V|)$
- Time per subprob: $O(n \cdot |P|)$

$\Rightarrow O(n^3 \cdot |V| \cdot |P|)$

Rmk: Valiant '75 $O(|P| \cdot n^{2.373})$

For most PL there are linear-time passing alg'm.