CS 758/858: Algorithms

http://www.cs.unh.edu/~ruml/cs758

Three-dimensional Dynamic Programming

The Context-Free Grammar Parsing Problem

Given:

1. A context-free grammar G in Chomsky Normal Form
   - ϵ-free
   - all rules like:
     - $A \to BC$
     - $A \to a$

     So $A \to BCD$ becomes $A \to BX$ and $X \to CD$
2. String of length n of tokens w_i

Dynamic Programming

Bottom-up dynamic programming:

$$\pi[i, j, A] = \text{can } A \text{ span } i\text{–}j?$$
$$\pi[i, i, A] =$$

Dynamic Programming

Bottom-up dynamic programming:

$$\begin{aligned} \pi[i,j,A] &= \text{can } A \text{ span } i\text{–}j? \\ \pi[i,i,A] &= \text{true iff } A \rightarrow w_i \\ \pi[i,j,A] &= \end{aligned}$$

Dynamic Programming

Bottom-up dynamic programming:

$$
\begin{aligned}
\pi[i,j,A] &= \text{can } A \text{ span } i\text{–}j\text{?} \\
\pi[i,i,A] &= \text{true iff } A \to w_i \\
\pi[i,j,A] &= \bigvee_{A \to BC} \bigvee_{i \leq k < j} \pi[i,k,B] \wedge \pi[k+1,j,C]
\end{aligned}
$$

Pseudo-code

1. initialize π to false everywhere
2. for i from 1 to n
3. foreach nonterminal A
4. $\pi[i, i, A] \leftarrow$ true if$A \rightarrow w_i$
5. for len from 2 to n
6. for i from 1 to $n - (len - 1)$
7. $j \leftarrow i + (len - 1)$
8. for k from i to $j - 1$
9. foreach rule $A \rightarrow BC$
10. if $\pi[i, k, B]$ and $\pi[k + 1, j, C]$ then
11. $\pi[i, j, A] \leftarrow$ true
12. return $\pi[1, n, S]$

Break

- asst 6
- from Steve:
  - read problem carefully, esp what is given
  - don't forget initialization/construction time complexity
  - do sanity checks, eg small example (even for written)

More Parsing

The Cocke-Younger-Kasami Algorithm

Probabilistic CFGs via bottom-up DP:

$\pi[i, j, A]$ = probability of A spanning i–j

$\pi[i, i, A]$ =

The Cocke-Younger-Kasami Algorithm

Probabilistic CFGs via bottom-up DP:

$$\begin{aligned}
\pi[i,j,A] &= \text{probability of } A \text{ spanning } i\text{–}j \\
\pi[i,i,A] &= P(A \to w_i) \\
\pi[i,j,A] &=
\end{aligned}$$

The Cocke-Younger-Kasami Algorithm

Probabilistic CFGs via bottom-up DP:

$$\begin{aligned}
\pi[i,j,A] &= \text{probability of } A \text{ spanning } i\text{–}j \\
\pi[i,i,A] &= P(A \to w_i) \\
\pi[i,j,A] &= \max_{A \to BC} \max_{i \le k < j} \pi[i,k,B] \times \pi[k+1,j,C] \times P(A \to BC)
\end{aligned}$$

Pseudo-code

1. initialize π to all zeros
2. for i from 1 to n
3. foreach nonterminal A
4. $\pi[i, i, A] \leftarrow P(A \rightarrow w_i)$
5. for len from 2 to n
6. for i from 1 to $n - (len - 1)$
7. $j \leftarrow i + (len - 1)$
8. for k from i to $j - 1$
9. foreach rule $A \rightarrow BC$
10. $p \leftarrow \pi[i, k, B] \times \pi[k + 1, j, C] \times P(A \rightarrow BC)$
11. $\pi[i, j, A] = \max(p, \pi[i, j, A])$
12. return $\pi[1, n, S]$

Variations

All parses

Summary of Dynamic Programming

1. optimal substructure: global optimum uses optimal solutions of subproblems
2. ordering over subproblems: solve 'smallest' first, build 'larger' from them
3. 'overlapping' subproblems: polynomial number of subproblems, each possibly used multiple times
4. independent subproblems: optimal solution of one subproblem doesn't affect optimality of another

- top-down: memoization
- bottom-up: compute table, then recover solution

EOLQs

For example:

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!