Algorithm Design Techniques

Dynamic Programming : String / Text Problems: Examples

- Parsing

2/4/2016 Sundar B. CSIS, BITS, Pilani

Problem: Parsing

- Given a context free grammar G in Chomsky Normal Form (CNF), and a string w, verify whether w is in L(G).
- Note:
  - This is not an optimization problem.

CONTEXT FREE GRAMMARS

- Recall:
  - A Context Free Grammar **G** is a quadruple (**V,T,P,S**) where
    - **V** is a set of non-terminal symbols
    - **T** is a set of terminal symbols
    - **P** is a set of rules of the form
      - **A --> α**
        - where **A** is a non-terminal i.e. **A** $\in$ **V**
        - and α is a string of terminals and non-terminals i.e. $\alpha \in$ **(V U T)***
    - **S** $\in$ **V** is the start symbol

CFGs – Chomsky Normal Form

- Recall:
  - A Context Free Grammar **G = (V,T,P,S)** is in Chomsky Normal Form
  - if every rule in P is in one of the following forms:
    - **A --> BC** for non-terminals **A**, **B** , and **C**.
    - **A --> a** for non-terminal **A**, and terminal **a**.
    - **S --> ε** where **ε** is the empty string.

Problem - Parsing (for CNF grammars) : Analysis

- Note that the parsing problem:
  - Is **w in L(G)**?

where G = (V,T,P,S) is essentially a decision problem asking

  - can **w** be derived from **S** using rules in **P**?
- How do you divide this problem into sub-problems?
  - Note that grammar rules are implicitly defined using structural induction – e.g. in our case:
    - A rule of the form **A --> BC** can be read as :
      - a string α can be derived from **A** if
        - a string β can be derived from **B** and
        - a string γ can be derived from **C** and
        - $\alpha = \beta . \gamma$
    - Rules of the other two forms are base cases of induction.

Problem - Parsing (for CNF grammars): Analysis [2]

- Formulation of sub-problems:
  - Denote a string as **w[i..j]**
  - If there is a rule of the form **A --> BC** and if **w[i..j]** can be derived from **A**
    - then there must be a **k** such that
      - **i <= k <= j** and
      - **w[i..k]** can be derived from **B** and
      - **w[k+1..j]** can be derived from **C**
- Now formulate the parsing problem as:
  - can **w[1..n]** be derived from **S**?
    - for **G = (V,T,P,S)**

PROBLEM – PARSING: RECURRENCE RELATION

- Define:
  - **Sym[i,j]** for **1<= i <= j <= n**, where **n is |w|**, as
    - the set of all symbols that can derive the string **w[i..j]** from the rules of the grammar.
- Recurrence relation:
  - Sym[i,j] =
    - { A | A --> BC $\in$ P, B $\in$ Sym[i,k], and C $\in$ Sym[k+1, j] for some k s.t. i <= k < j } if i < j
    - { A | A --> a $\in$ P and w[i]=a } if i=j
- Then the parsing problem can be decided by answering
  - Is S $\in$ Sym[i,j]?

PROBLEM – PARSING: DYNAMIC PROGRAMMING

- The recurrence from the previous slide can be implemented as a DP algorithm:
  - Referred to as **Cocke-Younger-Kasami** algorithm (*see your text book for* **Theory of Computation**).
- Exercise:
  - Analyze the time and space complexity of CYK algorithm.