Discrete Mathematics

CS204: Spring, 2008

Jong C. Park
Computer Science Division, KAIST

Today's Topics

Equivalence Relations

Matrices of Relations

Relations

Matrices of Relations

- Example
  - Let R_1 be the relation from $X = \{1,2,3\}$ to $Y = \{a,b\}$ defined by $R_1 = \{(1,a), (2,b), (3,a), (3,b)\}$, and let R_2 be the relation from Y to $Z = \{x,y,z\}$ defined by $R_2 = \{(a,x), (a,y), (b,y), (b,z)\}$.
  - The matrix of R_1 relative to the orderings 1, 2, 3 and a, b is

$$A_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{cc} a & b \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{array}$$

Matrices of Relations

– and the matrix of R_2 relative to the orderings a, b and x, y, z is

$$A_2 = \begin{array}{c|ccc} & x & Y & z \\ \hline a & 1 & 1 & 0 \\ b & 0 & 1 & 1 \end{array}$$

– The product of these matrices is

$$A_1A_2 = \begin{array}{c|ccc} & x & y & z \\ \hline 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 3 & 1 & 2 & 1 \end{array}$$

Matrices of Relations

- Interpretation
  - The ikth entry in A_1A_2 is computed as

$$\begin{array}{ccc} & a & b \\ i & s & t \end{array} \qquad \begin{array}{c} k \\ u \\ v \end{array} = su + tv$$

  - If this value is nonzero, then either su or tv is nonzero.
  - Suppose that $su \neq 0$. (The argument is similar if $tv \neq 0$.) Then $s \neq 0$ and $u \neq 0$. This means that $(i,a) \in R_1$ and $(a,k) \in R_2$. This implies that $(i,k) \in R_2 \circ R_1$. We have shown that if the ikth entry in A_1A_2 is nonzero, then $(i,k) \in R_2 \circ R_1$.

Matrices of Relations

– The converse is also true. Assume that $(i,k) \in R_2 \circ R_1$. Then, either

1. $(i,a) \in R_1$ and $(a,k) \in R_2$ or
2. $(i,b) \in R_1$ and $(b,k) \in R_2$.

– If 1 holds, then $s = 1$ and $u = 1$, so $su = 1$ and $su + tv$ is nonzero. Similarly, if 2 holds, $tv = 1$ and again we have $su + tv$ nonzero. We have shown that if $(i,k) \in R_2 \circ R_1$, then the ikth entry in A_1A_2 is nonzero.

Matrices of Relations

- Theorem
  - Let R_1 be a relation from X to Y and let R_2 be a relation from Y to Z. Choose orderings of X, Y, and Z.
  - Let A_1 be the matrix of R_1 and let A_2 be the matrix of R_2 with respect to the orderings selected.
  - The matrix of the relation $R_2 \circ R_1$ with respect to the orderings selected is obtained by replacing each nonzero term in the matrix product A_1A_2 by 1.
  - Proof.
    - Explained earlier through the interpretation.
      - That is, the ikth entry in A_1A_2 is nonzero if and only if $(i,k) \in R_2 \circ R_1$.

Matrices of Relations

- How to determine whether a relation R on a set X is
  - transitive?
    - If A is the matrix of R (relative to some ordering), we compute A^2. We then compare A and A^2. The relation R is transitive if and only if whenever entry i, j in A^2 is nonzero, entry i, j in A is also nonzero. The reason is that entry i, j in A^2 is nonzero if and only if there are elements (i,k) and (k,j) in R. Now R is transitive if and only if whenever (i,k) and (k,j) are in R, then (i,j) is in R. But (i,j) is in R if and only if entry i, j in A is nonzero.
    - Therefore, R is transitive if and only if whenever entry i, j in A^2 is nonzero, entry i, j in A is also nonzero.

Matrices of Relations

- Example
  - The matrix of the relation $R = \{(a,a), (b,b), (c,c), (d,d), (b,c), (c,b)\}$ on $\{a,b,c,d\}$, relative to the ordering a, b, c, d, is

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Its square is

$$A^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - We see that whenever entry i, j in A^2 is nonzero, entry i, j in A is also nonzero. Therefore, R is transitive.

Matrices of Relations

- Example
  - The matrix of the relation $R = \{(a,a), (b,b), (c,c), (d,d), (a,c), (c,b)\}$ on $\{a,b,c,d\}$, relative to the ordering a, b, c, d, is

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Its square is

$$A^2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - The entry in row 1, column 2 of A^2 is nonzero, but the corresponding entry in A is zero. Therefore, R is not transitive.

Today's Topics

Introduction

Examples of Algorithms

Analysis of Algorithms

Algorithms

Introduction

- Algorithm
  - a finite sequence of instructions
- Characteristics of an algorithm
  - Input
    - It receives input.
  - Output
    - It produces output.
  - Precision
    - The steps are precisely stated.

Introduction

- Characteristics of an algorithm (continued)
  - Determinism
    - The intermediate results of each step of execution are unique and are determined only by the inputs and the results of the preceding steps.
  - Finiteness
    - It terminates; that is, it stops after finitely many instructions have been executed.
  - Correctness
    - The output produced by the algorithm is correct; that is, the algorithm correctly solves the problem.
  - Generality
    - It applies to a set of inputs.

Introduction

- Example
  - An algorithm to find the maximum of three numbers *a*, *b*, and *c*:
    1. *large* = *a*.
    2. If *b* > *large*, then *large* = *b*.
    3. If *c* > *large*, then *large* = *c*.
  - Properties
    - Input
    - Output
    - Precision
    - Determinism
    - Finiteness
    - Correctness
    - Generality

Introduction

- Example
  - Pseudocode

Algorithm 4.1.1: Finding the Maximum of Three Numbers

Input: a, b, c
Output: *large* (the largest of a, b, and c)

```
max3(a, b, c) {
    large = a
    // if b is larger than large, update large
    if (b > large)
        large = b
    // if c is larger than large, update large
    if (c > large)
        large = c
    return large
}
```

Introduction

- Another example
  - An algorithm to find the largest value in a sequence

Algorithm 4.1.2: Finding the Maximum Value in a Sequence

Input: s, n

Output: $large$ (the largest value in the sequence s)

```
max(s, n) {
   large = s_1
   for i = 2 to n
      if (s_i > large)
         large = s_i
   return large
}
```

Examples of Algorithms

- Searching
- Sorting
- Time and Space for Algorithms
- Randomized Algorithms

Searching

Algorithm 4.2.1: Text Search

Input: p (indexed from 1 to m), m, t (indexed from 1 to n), n

Output: i

```
text_search(p, m, t, n) {
  for i = 1 to n − m + 1 {
    j = 1

    // i is the index in t of the first character of the
    // substring to compare with p, and j is the index in p

    // the while loop compares t_i ··· t_{i+m−1} and p_1 ··· p_m
    while (t_{i+j−1} == p_j) {
      j = j + 1
      if (j > m)
        return i
    }
  }
  return 0
}
```

Sorting

Algorithm 4.2.3: Insertion Sort

Input: s, n
Output: s (sorted)

$insertion\_sort(s, n)$ {
  for $i = 2$ to n {
    // save s_i so it can be inserted into the correct place
    $val = s_i$
    $j = i - 1$
    // if $val < s_j$, move s_j right to make room for s_i
    while ($j \geq 1 \ \wedge val < s_j$) {
      $s_{j+1} = s_j$
      $j = j - 1$
    }
    $s_{j+1} = val$ // insert val
  }
}

Time and Space for Algorithms

- Resources
  - Time
    - the number of steps
    - best-case time
    - worst-case time
    - average-case time
  - Space
    - the number of variables, length of the sequences involved

Randomized Algorithms

- Relaxing the requirements of an algorithm
  - Relaxing Finiteness
    - an operating system
  - Relaxing Determinism
    - those written for more than one processor
      - for a multiprocessor machine
      - for a distributed environment
    - making random decisions
  - Relaxing Generality and Correctness
    - solutions for practical problems

Randomized Algorithms

- Example
  - shuffling the values in the sequence $a_1, \ldots, a_n$.
  - *rand*(*i*,*j*): returns a random integer between *i* and *j*, inclusive.

Algorithm 4.2.4: Shuffle

Input: a, n

Output: a (shuffled)

$shuffle(a, n)$ {
  for $i = 1$ to $n - 1$
    $swap(a_i, a_{rand(i,n)})$
}

Analysis of Algorithms

- Analysis of an algorithm
  - a process of deriving estimates for the time and space needed to execute the algorithm
- Example
  - Given a set X of n elements, some labeled “red” and some labeled “black,” we want to find the number of subsets of X that contain at least one red item.
  - Since a set that has n elements has 2^n subsets, the program, if it chooses to examine every subset, would require at least 2^n units of time to execute.

Analysis of Algorithms

- Issues
  - The time needed to execute an algorithm is a function of the input.
  - But it is difficult to obtain an explicit formula for this function.
  - We choose to use parameters that characterize the size of the input.
    - Example
      - If the input is a set containing *n* elements, we would say that the size of the input is *n*.
    - best-case, worst-case, average-case time

Analysis of Algorithms

- Definition
  - Let f and g be functions with domain {1, 2, 3, ...}.
  - We write

    $$f(n) = \mathcal{O}(g(n))$$

    and say that $f(n)$ is of order at most $g(n)$ or $f(n)$ is big oh of $g(n)$ if there exists a positive constant C_1 such that

    $$|f(n)| \leq C_1|g(n)|$$

    for all but finitely many positive integers n.
  - We say that g is an asymptotic upper bound for f.

Analysis of Algorithms

- We write

$$f(n) = \Omega(g(n))$$

  and say that $f(n)$ is of order at least $g(n)$ or $f(n)$ is omega of $g(n)$ if there exists a positive constant C_2 such that

$$|f(n)| \geq C_2|g(n)|$$

  for all but finitely many positive integers n.
- We say that g is an asymptotic lower bound for f.
- We write

$$f(n) = \Theta(g(n))$$

  and say that $f(n)$ is of order $g(n)$ or $f(n)$ is theta of $g(n)$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- We say that g is an asymptotic tight bound for f.

Analysis of Algorithms

- Examples
  - Since $60n^2 + 5n + 1 \leq 60n^2 + 5n^2 + n^2 = 66n^2$ for all n $\geq$ 1, we may take $C_1 = 66$ to obtain $60n^2 + 5n + 1 = O(n^2)$.
  - Since $60n^2 + 5n + 1 \geq 60n^2$ for all n $\geq$ 1, we may take $C_2 = 60$ to obtain $60n^2 + 5n + 1 = \Omega(n^2)$.
  - Since $60n^2 + 5n + 1 = O(n^2)$ and $60n^2 + 5n + 1 = \Omega(n^2)$, $60n^2 + 5n + 1 = \Theta(n^2)$

Summary

- Equivalence Relations
- Matrices of Relations
- Algorithms
  - Introduction
  - Examples of Algorithms
  - Analysis of Algorithms