**Final Exam Topics COP3530H, Fall 2011**

**Topics (since or in vicinity of midterm):**

**1. Depth First Search**

Algorithms Based on Depth First Search

 Finding cycles in a directed graph; Topological sort; Reachability problem;

 Connected components; Reflexive transitive closure

Alternative of using Union/Find partitioning to compute connected components.

**2. Greedy**

Min Spanning Tree

Prim's Algorithm

Kruskal's Algorithm

Shortest Path Problem by Dijkstra’s Algorithm

 Greedy based on closest unsettled node

 Use Adjacency Lists for weighted arcs and POT for unsettled – O(M lg N) time

 O(N M lg N) time extension for all shortest paths

 Or use Adjacency Matrix for arcs and List for unsettled – O(N2) time

Huffman Coding

Scheduling Problems

 Greedy heuristics for hard problems (min the max finishing time)

 Cases where greedy works (min the mean finishing time)

**3. NP, NP-Complete and NP-Hard**

Decision vs optimization problems

NP as collection of non-deterministic polynomial time decision problems

NP as collection of deterministically checkable polynomial time decision problems

Polynomial reducibility

NP-Complete class

Show SAT is in NP

Polynomial time reduction of SAT to 3SAT

SAT, 3SAT as canonical NP-Complete problems

Polynomial time reduction of 3SAT to SubsetSum

Show SubsetSum is in NP and hence NP -Complete

Polynomial time equivalence of SubsetSum and Partition

Partition and scheduling

Scheduling problem anomalies

Optimal strategy for UET (unit execution tree or anti-tree)

QSAT (or TQBF) as NP-Hard problem

**4. Flow Typing Problems**

Maximum Network Flow (Ford-Fulkerson)

Using BFS to improve performance

Focusing on the heaviest residual path

Data flow analysis (loops and forward/backward flow)

**5. Relational Data Model**

Relational database is a collection of tables, called **relations**

Each row is called a **tuple** and is an instance of the relation

Each column is a labeled **attribute;** Set of attribute names is **scheme** of the relation

Each relation has a **key**, which is a set of attributes that uniquely identifies each row

Data Structures for Relations:

 BST – sorted by key; Hash, as function of key; List (linked or array) of tuples

Using keys to navigate among relations

**Relational algebra** – The operators:

 Union  Intersection – Difference

C (R) Selection πB1,…Bn (R) Projection

R ∞ Ai=Bj S Join R ∞ S Natural Join

Complexity Analyses of Implementing Relational Operators

**Query optimization** – Algebraic Properties of Relational Operators

**Views** – dynamically derived relations

**6. Divide and Conquer**

Integer multiplication

Tromino Tiling

Closest Points

Doubly Log Max (D&C plus amortization)

**7. Lindenmayer Systems (an aside)**

Parallelism in Lindenmayer Systems

**8. Dynamic Programming**

Cocke-Kasami-Younger

 Use Principle of Optimality – build from pairs of shorter substrings

Let your fingers do the walking – O(N3) time

LCS – longest Common Subsequence – O(N\*M)

 Use Principle of Optimality – build from shorter prefixes

Edit distance – O(N\*M)

 Use Principle of Optimality – build from shorter prefixes

Reflexive Transitive Closure by Warshall’s Algorithm

 Use Principle of Optimality – intermediate (pivot) nodes

 Use a Boolean Adjacency Matrix – O(N3) time

 Comparison to reflexive transitive closure by depth first search

All Shortest Paths Problem by Floyd’s Algorithm

 Use Principle of Optimality – intermediate (pivot) nodes

 Use an Adjacency Matrix to represent weighted arcs – O(N3) time

Knapsack (parameters now N and W, rather than just N)

 Analogy to Radix Sort

**9. Randomized Algorithms**

Las Vegas (Hopefully fast; If succeeds, gives correct answer)

 Ethernet Protocol

 Dining Philosophers

Monte Carlo (Fast; Can give approximation)

Integration

Digests for file transfers

Global Illumination

**10. String matching**

Brute Force (expected n; worst case m\*n)

Rabin-Karp Fingerprint Technique (expected m+n)

 Monte Carlo in nature if accept possible failure (worst case m+n)

 Las Vegas in nature if do check on each candidate match (worst case m\*n)

Knuth-Morris-Pratt: KMP

 Pre-compute pattern shifts against self (expected and worst m+n)

 Uses DFA to compute shift

Horspool’s Algorithm

 Focus on matching right to left

Boyer-Moore

 Similar to Horspool but improves how to shift pattern (can be sublinear)

**11. Halting Problem (another aside)**

Uses diagonalization

**Samples on Database Operations**

 **1.** The following table shows algorithmic costs for **naive** approaches (ones not involving sorts or indices) to relational operations. Fill in the columns associated with the use of **indices**. You may assume constant time index lookup via a hash table. Assume **|R| = n**, **|S| = m**, **t = n+m**, and **|Result| = k**

|  |  |  |
| --- | --- | --- |
|  | **Naive** | **Indexed** |
| **R  S** | **nm** |  |
| **R – S** | **nm** |  |
| **C (R)** | **n** |  |
| **π– (R)** | **n^2** |  |
| **R ∞ S** | **nm** |  |

 **2.** Assume we wish to issue the following query :

**NAME (STATE=FL((DIRECTORS  BORROWERS) ∞ LOCATIONS)))**

Present the tree associated with this query expression.

Now show how the following algebraic rules may be applied to optimize the query by pushing the projection and selection operators as low as possible. Present the new expression and its corresponding tree

**Selection Pushing below Joins**

**(C (R ∞ S))  (C (R) ∞ S) , provided all attributes of C are in R**

**(C (R ∞ S))  (R ∞ C (S)) , provided all attributes of C are in S**

**Projection Pushing below Unions**

**(L (R  S))  (L (R)  L (S))**

**Limited Projection Pushing below Joins**

**(L (R ∞ A=B S))  (L (M (R) ∞ A=B N (S))) , where**

**1) M consists of attributes of L from R followed by attribute A, if it is not in L,**

**2) N consists of attributes of L from S followed by attribute B, if it is not in L.**

**Projection Identity**

**L (R)  R , when L is all the attributes of R**