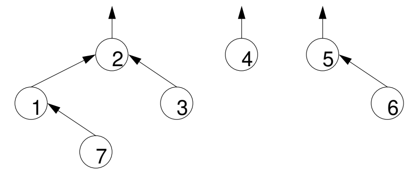
CSE 326
Lecture 28

Spring Quarter 1995

Weighted Union



	1	2	3	4	5	6	7
up	2	0	2	0	0	5	1
weight	-	4	-	1	2	-	-

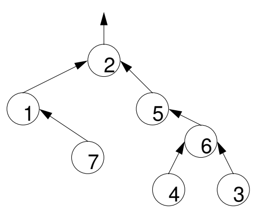
- W-Union(m,n) {

```
if weight(m) ≤ weight(n) then
    {up[m] := n; weight[n] := weight[n] + weight[m]}
else
    {up[n] := m; weight[m] := weight[m] + weight[n]}}
```


Path Compression Code

- Find(i) {

```
local int r,t;
r := i
while not(up[r] = 0) do r := up[r];
t := up[i];
if not(t=0) then
  while not(t=r) do {up[i] := r; i := t; t := up[t]}
return(r) }
```



Complexity of Weighted Union and Path Compression

- Tarjan proved m weighted union and path compression find operations on a set of n elements has worst case time complexity O(m α(m,n)).

  α(m,n) is called "inverse ackermann's function". It is very, very slow growing. Indeed, $\alpha(m,n) \leq 4$ for all practical values of m and n.
- For all practical purposes implementing dynamic equivalence with weighted union and path compression find yields an essentially amortized constant time per operation data structure.
- In many practical cases neither weighted union nor path compression find are needed. When you need dynamic equivalence you have to decide if long paths can develop. If they can in your application then you need to introduce either weighted union or path compression find or both.

Summary of the the Course

- Lists, Stacks, and Queues
  - Basic multipurpose data structures
  - Managing a private store
  - Cursor implementation of pointers
  - Applications
    > polynomials
    > text editing *
    > adjacency list representation of a graph and depth first search (Section 9.1.1, page 320 and sections 9.6.1 and 9.6.4, pages 357 and 366)
    > sparse matrices *

* indicates that the topic was not covered in the book.

Summary Continued

- Trees
  - Pointer representation of multi-way trees
  - Tree traversals
    > preorder
    > postorder
    > inorder
  - Simple binary search
  - Binary search trees
    > Generic binary search trees
    > AVL trees
    > Splay trees
  - B-trees
  - Multidimensional search trees
    > Quad trees *
    > k-d trees *
  - Optimum binary search tree (Section 10.3.3, page 422)

Summary Continued

- Hashing
  - Choosing a hash function
    > division method
    > mulitplication method *
    > universal hash functions *
  - Open hashing (chaining)
  - Closed hashing
    > linear probing
    > quadratic probing
    > double hashing
  - Rehashing

Summary Continued

- Priority queues
  - Binary heap
  - d-heap
  - Leftist trees
  - Skew heaps
  - Calendar Queue *
  - Applications
    > Discrete event simulation *
    > N-body problem *
- Dynamic equivalence
  - Up-tree implementation
  - Weighted Union
  - Path Compression
  - Application
    > Minimum spanning tree, Kruskal's algorithm (Section 9.5.2 page 354)

Summary Continued

- Programming Methods
  - Recursion
  - Call by reference vs. call by value
  - Memoization
  - Dynamic programming (Section 10.3, page 416)
  - Converting a recursive algorithm to an iterative one.
    > Eliminating end recursion
    > Using a stack
    > Observing the recursion as a two-pass iteration, as an example the AVL tree insertion or deletion.
  - Balancing to achieve good performance (AVL, weighted union)
  - Divide and conquer (merge sort, fast polynomial multiplication, binary search, ...)
  - Looking for good amortized performance (Splay, Skew, Rehash)
  - General familiarity with more advanced data structures and their applications.

Summary Concluded

- Analysis
  - Analyzing basic loops
  - Developing and solving recurrences which arise when using recursion.
  - Upper bound vs. lower bound analysis
  - Worst case vs. average case analysis
  - Amortized analysis vs. worst case per operation analysis
  - Correctness proofs
- Problem solving practice
  - Most homework involved problem solving rather that rote practice. Problem solving skills are among the most valuable skills you can have.
- Programming practice
  - More familiarty with object oriented programming
  - Strengthening of C++ experience.
  - Introduction to Unix
  - Programs required a combination of problems solving and coding known solutions. This is what many of you will do in your future employment.

Text Coverage

- Chapters 1-4 and 8 all sections
- Chapter 5 all except section 5.6.
- Chapter 6 all except section 6.8
- Also parts of:
  - Section 7.5 Heapsort
  - Section 7.6 Mergesort
  - Section 9.5.2 Kruskal's algorithm
  - Section 9.1.1 Representation of graphs
  - Sections 9.6.1 and 9.6.4 Depth-first search
  - Section 10.3.3 Optimal binary search trees

Topics Covered Which are in the Book

- Sparse matrices
- Text editing data structures
- Multiplication method hash function and universal hash functions
- Calendar queue

Final Exam

- Wednesday, June 7, 2:30 - 4:20
- Exam starts promptly at 2:30 and all papers must be turned in by 4:20.
- Sit in alternate seats. Several end seats reserved for left handers.
- Exam covers all the material in the course.
- Must use 8.5 x 11 inch blue book, 32 cents at the HUB consession stand.
- You may prepare the first 4 pages (2 sheets of paper, 2 sides each) of the blue book as a help sheet. Your exam will start on page 5.
- No other notes or books may be used during the exam. Calculators can be used but are not required. There may be some simple calculations to do, but they can be done by hand.
- Content of the Exam:
  1. Understand the methods and algorithms well. Drill yourself this.
  2. Problem solving skill. The homeworks were practice for this.

Good Luck

- Best of luck on the final exam
- Have a great summer