COP 4516: Competitive Programming and Problem Solving

Giri Narasimhan & Kip Irvine

Phone: x3748 & x1528

{giri,irvinek}@cs.fiu.edu

Problems to think about!

- What is the least number of comparisons you need to sort a list of 3 elements? 4 elements? 5 elements?
- How to arrange a tennis tournament in order to find the tournament champion with the least number of matches? How many tennis matches are needed? How to arrange a tennis tournament in order to find the runner up to the champion with the least number of matches?
- How to randomize the order of a list?

Sorting Algorithms

- SelectionSort
- InsertionSort
- BubbleSort
- ShakerSort
- MergeSort
- HeapSort
- QuickSort
- Bucket & Radix Sort
- Counting Sort

Data Structure Evolution

- Standard operations on data structures
  - **Search**
  - **Insert**
  - **Delete**
- Linear Lists
  - Implementation: **Arrays** (**Unsorted and Sorted**)
- **Dynamic** Linear Lists
  - Implementation: **Linked Lists**
- **Dynamic** Trees
  - Implementation: **Binary Search Trees**

Data Structures Comparison

Data Structure \ Operation	Search	Insert	Delete
Unsorted Array			
Sorted Array			
Unsorted Linked List			
Sorted Linked List			
Binary Search Trees			
Balanced Binary Search Trees			

BST: Search

```
TreeSearch(node x, key k)
  ▷ Search for key k in subtree rooted at node x
  if ((x = NIL) or (k = key[x]))
      then return x
  if (k < key[x])
      then return TreeSearch(left[x], k)
      else return TreeSearch(right[x], k)
```

Time Complexity: O(h)
h = height of binary search tree

Not O(log n) — Why?

BST: Insert

```
TreeInsert(tree T, node z)
    ▷ Insert node z in tree T
y ← NIL
x ← root[T]
while (x ≠ NIL)
        do y ← x
            if (key[z] < key[x])
                then x ← left[x]
                else   x ← right[x]
p[z] ← y
if (y = NIL)
    then root[T] ← z
    else  if (key[z] < key[y])
                then left[y] ← z
                else   right[y] ← z
```

Time Complexity: O(h)
h = height of binary search tree

Search for x in T

Insert x as leaf in T

BST: Delete

Time Complexity: O(h)
h = height of binary search tree

```
TreeDelete(tree T, node z)
  ▷ Delete node z from tree T
  if ((left[z] = NIL) or (right[z] = NIL))
      then y ← z
      else  y ← Tree-Successor(z)
  if (left[y] ≠ NIL)
      then x ← left[y]
      else  x ← right[y]
  if (x ≠ NIL)
      then p[x] ← p[y]
  if (p[y] = NIL)
      then root[T] ← x
      else  if (y = left[p[y]])
               then left[p[y]] ← x
               else  right[p[y]] ← x
  if (y ≠ z)
      then key[z] ← key[y]
           cop y's satellite data into z
  return y
```

Set y as the node to be deleted. It has at most one child, and let that child be node x

If y has one child, then y is deleted and the parent pointer of x is fixed.

The child pointers of the parent of x is fixed.

The contents of node z are fixed.

Data Structures Comparison

Data Structure \ Operation	Search	Insert	Delete
Unsorted Array	O(n)	O(1)	O(n)
Sorted Array	O(log n)	O(n)	O(n)
Unsorted Linked List	O(n)	O(1)	O(n)
Sorted Linked List	O(n)	O(n)	O(n)
Binary Search Trees	O(h)	O(h)	O(h)
Balanced Binary Search Trees	O(log n)	O(log n)	O(log n)

Animations

- **BST:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/BST-Example.html
- **Rotations:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/index2.html
- **RB-Trees:**
  http://babbage.clarku.edu/~achou/cs160/examples/bst_animation/RedBlackTree-Example.html

Example

- [0,6], [1,4], [2,13], [3,5], [3,8], [5,7], [5,9], [6,10], [8,11], [8,12], [12,14]

- **Simple Greedy Selection**
  - Sort by start time and pick in "greedy" fashion
  - Does not work. WHY?
    - [0,6], [6,10] is the solution you will end up with.
- **Other greedy strategies**
  - Sort by length of interval
  - Does not work. WHY?

Example

- [0,6], [1,4], [2,13], [3,5], [3,8], [5,7], [5,9], [6,10], [8,11], [8,12], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14] -- Sorted by finish times
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]

Greedy Algorithms

- Given a set of activities (s_i, f_i), we want to schedule the maximum number of non-overlapping activities.
- GREEDY-ACTIVITY-SELECTOR (s, f)

1. n = length[s]
2. S = {a_1}
3. i = 1
4. **for** m = 2 **to** n **do**
5. **if** s_m is not before f_i **then**
6. S = S U {a_m}
7. i = m
8. return S

Why does it work?

- THEOREM

  Let A be a set of activities and let a_1 be the activity with the earliest finish time. Then activity a_1 is in some maximum-sized subset of non-overlapping activities.
- PROOF

  Let S' be a solution that does not contain a_1. Let a'_1 be the activity with the earliest finish time in S'. Then replacing a'_1 by a_1 gives a solution S of the same size.

  Why are we allowed to replace? Why is it of the same size?

Then apply induction! How?

Greedy Algorithms – Huffman Coding

- Huffman Coding Problem

  Example: Release 29.1 of 15-Feb-2005 of TrEMBL Protein Database contains **1,614,107** sequence entries, comprising **505,947,503** amino acids. There are 20 possible amino acids. What is the minimum number of bits to store the compressed database?

  ~2.5 G bits or 300MB.
- How to improve this?
- Information: Frequencies are not the same.

Ala (A) 7.72	Gln (Q) 3.91	Leu (L) 9.56	Ser (S) 6.98
Arg (R) 5.24	Glu (E) 6.54	Lys (K) 5.96	Thr (T) 5.52
Asn (N) 4.28	Gly (G) 6.90	Met (M) 2.36	Trp (W) 1.18
Asp (D) 5.28	His (H) 2.26	Phe (F) 4.06	Tyr (Y) 3.13
Cys (C) 1.60	Ile (I) 5.88	Pro (P) 4.87	Val (V) 6.66

- **Idea:** Use shorter codes for more frequent amino acids and longer codes for less frequent ones.

Huffman Coding

2 million characters in file.

A, C, G, T, N, Y, R, S, M

IDEA 1: Use ASCII Code
Each need at least 8 bits,
Total = 16 M bits = 2 MB

IDEA 2: Use 4-bit Codes
Each need at least 4 bits,
Total = 8 M bits = 1 MB

Percentage Frequencies

IDEA 3: Use Variable Length Codes

A	22	11
T	22	10
C	18	011
G	18	010
N	10	001
Y	5	00011
R	4	00010
S	4	00001
M	3	00000

How to Decode?

Need Unique decoding!
Easy for Ideas 1 & 2.
What about Idea 3?

110101101110010001100000000110

110101101110010001100000000110

2 million characters in file.

Length = ?
Expected length = ?
Sum up products of frequency times the code length, i.e.,

(.22x2 + .22x2 + .18x3 + .18x3 + .10x3 + .05x5 + .04x5 + .04x5 + .03x5) x 2 M bits =

3.24 M bits = .4 MB