CPSC 500 Fundamentals of Algorithm Design and Analysis

Lecturer: Prof. David Kirkpatrick **Lecture Date: Oct. 9, 2013** **Scribe: Ben Zhu**

1. Overview

At the beginning of this lecture, we reviewed our discussion on lower bounds and Selection Problem. Afterwards we focus our discussion on Dictionary Problem, the operations, advantages and disadvantages. Then we take a further look at how to deal with the disadvantages and to improve the algorithms.

2. Review of last lecture

Lower bounds
We talked about two creative ways to formulating lower bounds on finding median, decision tree and adversary argument.

Return to Selection
We can use adversary argument as an easy way to reduce the 2-partitioning part of selection. It can be considered as an anti-algorithm bad instance that we choose bad input to preserve the uncertainty of selection.

3. Dictionary Problem

3.1 General Discussion

We have dictionaries over $U = \{0, \ldots, m\text{-}1\}$

Dictionary

- a set $S \subseteq U$, with $|S| = n$
- operations
  - member
  - closest
  - insert/ delete
  - max/ min
- Note: $\Omega(\lg|S|)$ lower bound for member on comparison-based model

Advantages

- $O(1)$ time member, insert/ delete

Disadvantages

- preprocessing (initialization)
- closest (predecessor/ successor)
- space!   key space is the size of universe

3.2 Discussion on preprocessing

Claim: We can augment a basic direct access table (using $\theta(|U|)$ space), so that it can be initialized (from an arbitrary initial configuration) in $\theta(|S|)$ time without losing any advantages.

Method

We use an array A with size $|U|$ to record the access table. Then we put some elements of S into the table with signature.

- So if we get a 1 signed from A, it is the element of S.
- If we get a 1 unsigned from A, it is actually a 0.

Signature control

We set up another array B , each contains a pointer to A.

- put a pointer to B instead of 1 in A
- for a random element in A , interpret the value and $\mathrm{mod}\ |S|$, find the related element in B, if the one in B does not point back, it is not an element of S
- B can be an empty set at beginning, add elements into S by adding more space to B
- the space usage is larger, since we change A from bit array to address array

3.3 Discussion on predecessor/ successor

Naive methods	successor	insert/ delete
Linear search	$O(m)$	$O(1)$
(Query a position and walk down)		
Maintain pointers	$O(1)$	$O(m)$
(Each element in array has a pointer to next element in set)		

Build some auxiliary structure

We set up a tree over the array (on top of leaves), mark leaves that in set and mark their parents.

- For an element in array, walk up until a marked node, find predecessor/ successor in that subtree as a marked leaf. (For a 1 in array, find marked sister node since a leaf level element in set have pointer for predecessor/ successor)
- Cost of binary tree with marked nodes is $O(\log m)$ to find the lowest marked ancestor.
- Also $O(\log m)$ for insert/ delete.
- We can store pointers for the leftmost/ rightmost marked leaf of its subtree for internal nodes.

Represent auxiliary tree like a heap with implicit pointers

Indexing:

parent (i) = $\lfloor i/2 \rfloor$

l.child (j) = 2 j

r.child (j) = 2 j +1

r-ancestor (i) = $\lfloor i/2^r \rfloor$ (r level above)

• This allows us to find the lowest marked ancestor using binary search on the ancestor path in $O(\log\log m)$ time.

• Unfortunately, insert/ delete still require looking at all of the nodes on the path to the lowest marked ancestor.

4. Conclusion

In this lecture, we mainly focused on Dictionary Problem. We talked about the operations, advantages and disadvantages of dictionary problem. Then we further discussed on how to deal with the disadvantages without losing any advantages. We started from discussion on preprocessing that we can augment a basic direct access table (using $\theta(|U|)$ space) to get it initialized in $\theta(|S|)$ time. Afterwards, we discussed on building auxiliary tree and representing auxiliary tree like a heap with implicit pointers to improve the algorithm to find the lowest marked ancestor. Furthermore, we will talk about using X-fast tree to achieve $O(\log\log m)$ time for all operations and space problems next lecture.

References

[1] Wikipedia, "upper and lower bounds," http://en.wikipedia.org/wiki/Upper_and_lower_bounds

[2] Wikipedia, "Selection algorithm," http://en.wikipedia.org/wiki/Selection_algorithm

[3] Wikipedia, "Algorithm," http://en.wikipedia.org/wiki/Algorithm