**CS311 Yoshii - Week 5 A (Notes-5A) F(n) of Sorting and Radix Sort**

**=====================================================================**

**🡺HW2P2\_Sorting**

**Agenda:**

* **Lower Bound of Comparison-Based Sorting (F(n))**
* **Radix Sort**
* **Summary of algorithm analysis**

**Recall that Binary Search was the optimal algorithm for searching through an ordered list.**

**F(n) = Theta(log N). We used a binary search tree to model the search process to come up with F(n).**

**Lower Bound on Sorting F(n) - Optimality**

When we want to find the lower bound of a problem the first step is to select a modeling tool.

**Selecting a modeling tool:**

**Problem: sorting**

**Class of Algorithms: element-element comparison based sorting**

**Fact:** given any algorithm in this class, a sequence of comparisons will produce a sorted list

**Fact:** each comparison has only 2 outcomes i < j or i > j

Whenever an algorithm is depending on a sequence of binary decisions,

**a decision tree** **is a good model** to use since each node models a binary decision.

**Modeling Tool: a decision tree**

Node: each node in the decision tree will represent a comparison

and there are two branches from each node

**A vs B**

/ \

Means Means

A < B B < A

/ \

**B vs C A vs C**

Path: each path from the root to a leaf will then represent a sequence

of comparisons producing a sorted list

With 3 elements, there are 6 possible

results depending on the relations among the 3 elements.

**==> 6 leaves**

**==> 6 different paths** down the tree to the 6 leaves

**Given A B C**

**Results ABC ACB BAC BCA CBA CAB are the leaves.**

**\*Inter\* Complete the above decision tree.**

**Analyzing the modeling tool:**

**# of leaves is at least # possible results = # permutations of n elements = n!**

e.g. 3! is 6.

But note that no two sequences of comparisons will produce the same

result because each comparison result represents a different ordering.

Therefore, **# of leaves is** **exactly n!**

**Therefore, our decision trees will always have n! leaves.**

THE BEST DECISION trees are the shallowest ones.

The more complete the tree is, the shallower they are.

**Our analyses using the shallowest Decision Tree represents the lower bound of sorting algorithms!!**

**Lower Bound of Sorting (for the Worst Case)**

**You have the shallowest decision tree (i.e. a complete tree if possible).**

**We already know that if there are N nodes in the tree, we have logN levels.**

**What if we have N! leaves? How many levels?**

**In the worst case, the longest path will be followed.**

root --- O ---- O -- ......-- O leaf at level B

**# of comparisons on the longest path = B** **Solve for B.**

#leaves = 2^B (if a complete tree i.e. the shallowest)

B = log(#leaves) but since B is an integer

B = ceil(log(#leaves))

B = ceil(log(n!)) (n is the number of elements to sort)

**Note log(n!)** = Sum of log j as j varies from 1 to n == **nlogn – 1.5n**

Therefore, **B = ceil(nlogn-1.5n)** (for a complete tree; the shallowest)

**The shallowest decision tree == best sorting algorithm**

**== # of comparisons is at most NlogN down a path to level B**

**The best sorting algorithm does at most NlogN comparisons ҉ ҉**

**Therefore, Merge Sort is Optimal in terms of Time but…..**

**Summary of F(n)s: ҉҉**

* **F(n) of searching an unordered list is Theta(n) comparisons**
* **F(n) of searching an ordered list is Theta(logn) comparisons**
* **F(n) of sorting when only one inversion/comparison is Theta(n^2) comparisons**
* **F(n) of comparison based sorting is Theta(nlogn) comparisons**

**Can you name the optimal algorithm for each?**

**Questions Relating the Above to the Real World Cases**

**Make sure you can answer these before you submit HW2P2**

1) You have discovered the W(n) for your sorting algorithm to be Theta(n^2).

Would you try to find a better algorithm? Under what circumstance?

2) You have discovered the W(n) for your sorting algorithm to be Theta(nlogn).

And it sorts in place. Would you try to find a better algorithm?

Why? Why not?

**Radix Sort - Breaking the NLOGN barrier !!!!!!!!????**

Radix Sort is also a Divide and Conquer algorithm which

was used in old days to sort punch cards.

**Initial Idea:**

If you had words to sort,

and there were 26 bins : a - z

Look at the 1st letter of each word and

put them in the corresponding bin.

For each bin, prepare 26 sub bins: a - z

Now, look at the 2nd letter of each word and

put them in the corresponding sub bins.

If each word had 3 letters, we would need

26 + 26\*26 + 26\*26\*26 bins!!!

(26 + 676 + 17576 = 18278)

**\*Inter\* In general, if each word had K letters, how many**

**bins will we need?**

Hint: Sum-of r^i for i = 0 to k = [r^(k+1) - 1] / (r - 1)

***Radix Sort == How can we cut down on the number of bins??***

**Let's assume we have N words all K letters long!**

**e.g. rat mop car map tar cat N=6 K=3**

**PASS1**

Start with the **very last letter of the words**.

Put the words into separate lists based on this letter.

We don't need 26 lists as you will see below.

**e.g. mop car rat map tar cat**

**list1 = mop map (p list)**

**list2 = car tar (r list)**

**list3 = rat cat (t list)**

Note that lists are ordered alphabetically (p,r,t).

Now create one list by appending these lists.

Note that this list has been sorted by the last letters.

**e.g. mop map car tar rat cat**

**PASS2**

Based on the 2nd to last character,

put the words into separate lists based on this letter.

**e.g. list1 = map car tar rat cat (a list)**

**list2 = mop (o list)**

Note that within each list, it's been sorted already by the very last letter

e.g. p-words before r-words before t-words in list1

Now append the lists

**e.g. map car tar rat cat mop**

**PASS3**

And now with the 1st character of each word,

**list1 = car cat (c-list)**

**list2 = map mop (m-list)**

**list3 = rat (r-list)**

**list4 = tar (t-list)**

Within each list, it has already been sorted according to the last 2 letters.

And finally, combining the lists

**e.g. car cat map mop rat tar**

**It works only if you maintain the original order within each sub-list**

**when you append the sub-lists back into one.**

***Space Complexity : How Many Lists Are Needed***

If we used a-list all the way to z-list in each pass,

we will still need 26 lists per pass.

3 letter words

26 sub-lists pass 1

26 sub-lists pass 2

26 sub-lists pass 3

= 26\*3 sub-lists

26\*K lists for K letter words and we also need 1 list per pass for the appended result.

If lists are arrays, we will need 26\*N slots per pass where N is

the number of words. Many of these slots are not used. Inefficient.

If lists are linked lists, we can append the linked lists without creating the new one.

If lists are linked lists, we can delete nodes and we put nodes into the sub-lists.

**🡺 At any point we have only N nodes 🡺 Sorts in place**

**\*Inter\* Does the above algorithm work if words are of different lengths?**

**Illustrate with an example.**

**e.g. rat map car mop ta ca**

**How do you make it work? Hint: padding**

***Time Complexity Analysis of Radix Sort:***

We subdivide into sub-lists K times where **K is the number of characters**

in a word. i.e. **K PASSES**

**For each pass, all N words must be looked at.**

Thus, **Theta(N\*K)** **for looking at the words**.

N=1024 K=10

K = logN

N\*K = NlogN

N=2000 K=10

K < logN

N\*K < NlogN

N=10,000 K=10

K <<< logN

N\*K <<< NlogN

== N

The fastest algorithm we know so far is Theta(NlogN).

If K (the #characters) is close to logN, Radix Sort is good.

**If K is much less than logN because words are very short**

**but the list is long, then it is Theta(N)**

**It breaks the NlogN barrier!!!!**

**No!!!!! It is not a comparison based sorting algorithm.**

**Conclusion on Radix Sort:**

**Space: The linked lists at any point never exceeds total of N nodes. Theta(N).**

**Time: Theta(N) if N is much bigger than K (i.e. short words) ҉ ҉**

**\*Inter\* For what other types of elements would you use Radix Sort???**

**Give a specific case.**

**EC1 Extra Credit Presentation Idea:**

**- Read about other sorting algorithms**

**\*\* Go to the String Matching file now! (NEW TOPIC)**

**!!! Email me now if you have any concerns about the content so far !!!**

**In-Class Exercise (Week5A) – Math Exercise – (MAX 2 people per team) 5 pts**

**\*\* Demonstrate the Choices** [**Tutor**](http://cgi.csusm.edu/ryoshii/mathex/mathex4/math.html) **\*\* (link)**

**Do the Week 5A Math Exercise**

**-------------------------------- break ---------------------------------------------------------------**

**\*\* Go over Analysis Summary Handout now.**

**Be working on HW2P2 \_sorting! \*\* GO OVER IT NOW \*\***

**-----------------------------------------------------------------------------------------------------------**

**The End of Part 2. Analysis:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***®Summarize here what you have learned in your own words and also write down your own thoughts/reactions/questions.***

***Email me now if you have any questions about what you read in this file.***