Lecture 3

Analysis of Algorithms
CSCI 5620

Topics

- Trees
- Comparison Sorting, Best Time
- Decision Tree
- Linear Expected Sorting Time Algorithm
- Proof of Correctness

Comparison Sorting

- Comparison, example "if a<b then"
- Merge-Sort took $\Theta(n \lg n)$ time.
- How many comparisons need to be done?
- Take $a_1, a_2, \ldots a_n$ all distinct elements.
  - How different ways the items be arranged?
  - Look at the decision tree for 2 items a_1, a_2
  - How many leaves does a decision tree have for n items?
  - What is the depth of the tree?

Comparison Sorting

- Take a_1, a_2, … a_n all distinct elements.
  - How different ways the items be arranged?
  - Look at the decision tree for 2 items a_1, a_2
  - How many leaves does a decision tree have for n items?
  - What is smallest depth of tree with n! leaves?
    - This equals the number of comparisons which must be done to arrive at a leaf.
  - $\lg(n!) \in \Theta(n \lg n)$

Decision Tree Example

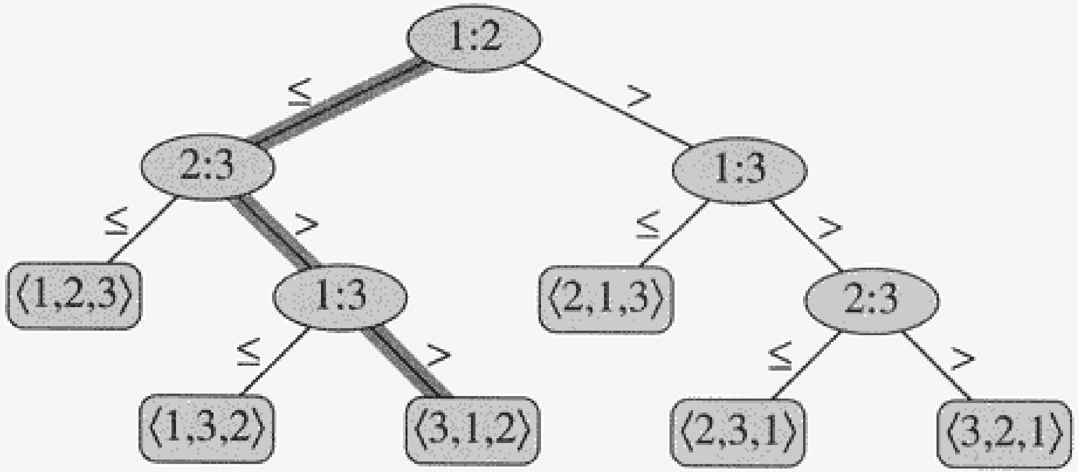

Figure 8.1 The decision tree for insertion sort operating on three elements. An internal node annotated by $i:j$ indicates a comparison between a_i and a_j. A leaf annotated by the permutation $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ indicates the ordering $a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(n)}$. The shaded path indicates the decisions made when sorting the input sequence $\langle a_1 = 6, a_2 = 8, a_3 = 5 \rangle$; the permutation $\langle 3, 1, 2 \rangle$ at the leaf indicates that the sorted ordering is $a_3 = 5 \leq a_1 = 6 \leq a_2 = 8$. There are $3! = 6$ possible permutations of the input elements, so the decision tree must have at least 6 leaves.

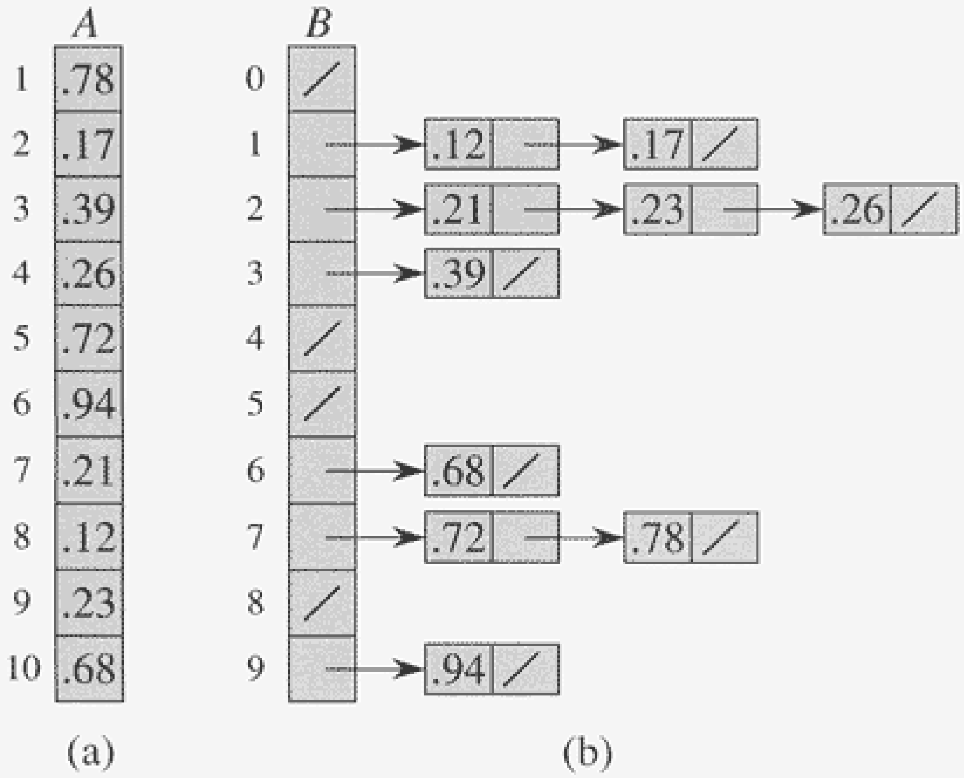

Figure 8.4 The operation of BUCKET-SORT. **(a)** The input array $A[1..10]$. **(b)** The array $B[0..9]$ of sorted lists (buckets) after line 5 of the algorithm. Bucket i holds values in the half-open interval $[i/10, (i+1)/10)$. The sorted output consists of a concatenation in order of the lists $B[0], B[1], \ldots, B[9]$.

Linear Time

- Bucket Sort
  - Assumes data if uniformly distributed over the interval [0,1)
  - Divides the [0,1) into n equal-sized subintervals, (or buckets)
  - Distributes the n input numbers into buckets.
  - Since we are uniform [0,1) each bucket has few items.

- Bucket-Sort(A)
  - n <---- length (A) 1
  - For i = 1 to n 2
    - Insert A[i] into linked list of bucket B[floor (n*A[i]] 3
  - For i= 0 to n-1 4
    - Sort B[i] with a Comparison Sorting algorithm 5
  - Concatenate B[0],B[1],…B[n-1] together 6
- Showing Bucket Sort is linear average running time (or expected running time not worse case)
  - Not a proof (we would need more probability)
  - Uniform random, B[i] expected size of B[i] is 1, i.e. $\Theta(1)$.
  - Line 3 take constant time.
  - For loop 2 with 3 is $\Theta(n)=\Theta(n)$
  - Line 5 take Size (B[i]) log (Size B[i]]) time, expected time $\Theta(1)$
  - For loop 2 with 3 is $\Theta(n*1)=\Theta(n)$
  - Line 6 is $\Theta(n)$

Proof of Correctness

- Loop Invariants
  - Conditions and relationships that are satisfied by the variables and data structures at the start of each iteration of the loop
  - Used to show why an algorithm is correct
  - Used for Loops (for, while, etc)
- Induction to show loop invariants
  - Initialization: It is true prior to the first iteration of the loop (BASE CASE)
  - Maintenance: If it is true before an iteration of the loop, it remain true before the next iteration. $P(k) \Rightarrow P(k+1)$
  - Termination: When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Sequential Search

- Input array A=<a1,…an> value v
- Output index i if v=A[i] or -1 otherwise
  - Index:=1
  - While index<=n and L[index] $\neq$ v
    - Index:=index+1
  - If (index>n) index:=-1

- Loop invariants
  - k^{th} iteration
  - index:=k & L[i] $\neq$ v for 1<=i<k
- Initialization When k:=1 index=1
- Maintenance: Suppose at the kth iteration index:=k & L[i] $\neq$ v for 1<=i<k.
  - At the k+1st iteration we know that L[k] $\neq$ v.
  - So L[i] $\neq$ v for 1<=i <k+1. Now if L[k+1]=v, the loop terminates and it is the first occurrence of v.
  - Otherwise index := index+1, or k+2 and

    L[i] $\neq$ v for 1<=i <k+1
- Termination: So if the loop terminates with k<=n, then index=k L[k]=v, otherwise the index=-1 and v is not in L

Pg 40 2-2

- Bubblesort (A)

  For i = 1 to A.length -1

  For j = A.length downto i+1

  If A[j]<A[j-1]

  Exchange A[j] and A[j-1]

- What is the loop invariant of the inner for loop?
  - Hint: What is the inner loop doing (in j terms)?
- What is the loop invariant of the outer for loop?
  - Hint: What do we have in terms of i?