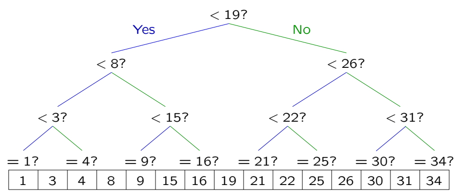
9—Sorting Lower Bound

CSE326 Spring 2002

April 21, 2002

More Than a Data Structure

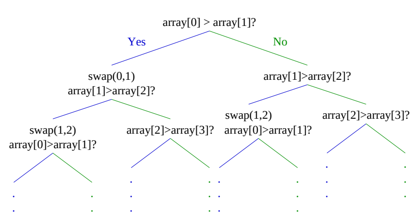


Binary Search as a Tree

Comparison Sorting Algorithms

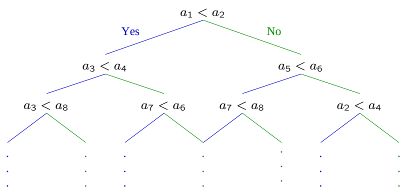
```
void InsertionSort (
        int *array ,
        int n)
{
  for (int i = 1;
       i < n;
       i++) {
    int x = array [ i ];
    for (int j = i;
         j > 0
          && array[j−1] > x;
         j−−)
      array [j] = array [j −1];
    array [j] = x;
  }
}
```

⇒



All Possible Execuations of a Algorithm as a Tree

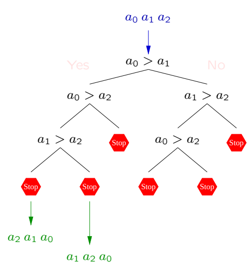
Comparison Algorithms



- Each node is a *comparison*
- Particular input gives a path
- Ignore all the swaps

  We want a lower bound, anyway. . .

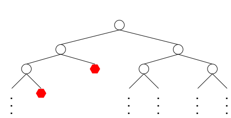
Insertion Sort




```
void InsertionSort (
        int *array ,
        int n)
{
  for (i = 1... n)
    x = array [ i ];
    for (j = i ...1
            && array[j−1] > x)
      array [ j ] = array [ j −1];
    array [ j ] = x;
}
```

- 3 Item Array
- One leaf for each possible permutation

General Sorting Algorithms



- In order to correctly sort, must have one leaf for each permutation of n items
- Smallest *height* when tree is *perfect*

What is $\log n!$ anyway?

Get it from the top

$$
\begin{aligned}
\log n! &= \log\left(n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1\right) \\
&\le \log\left(n \cdot n \cdots n \cdot n\right) \\
&= \log n^n = n \log n
\end{aligned}
$$

So $\log n! \in O(n \log n)$

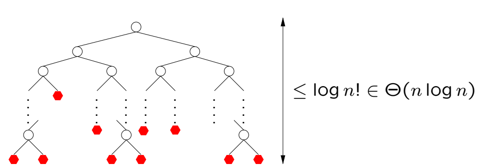
The Bound Underneath

$$
\begin{aligned}
\log n! &= \log\left(n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1\right) \\
&= \log n + \log(n-1) + \cdots \log 2 + \log 1 \\
&\ge \log \frac{n}{2} + \log \frac{n}{2} + \cdots + \log \frac{n}{2} + \log\left(\frac{n}{2} - 1\right) + \log\left(\frac{n}{2} - 2\right) + \cdots + \log 1 \\
&= \frac{n}{2} \log \frac{n}{2} + \log\left(\frac{n}{2} - 1\right) + \log\left(\frac{n}{2} - 2\right) + \cdots + \log 2 + \log 1 \\
&\ge \frac{n}{2} \log \frac{n}{2} \in \Omega(n \log n)
\end{aligned}
$$

$\log n! \in O(n \log n) \cap \Omega(n \log n) = \Theta(n \log n)$

Summing Up

Any Comparison Sorting Algorithm



Must Take at Least $n \log n$ Time