make history.

THE UNIVERSITY of ADELAIDE

Iterative sorting algorithms

Dr. Anna Kalenkova

Math Learning Center: University of Adelaide

https://www.adelaide.edu.au/mathslearning/

Sorting algorithms

Comparison-based sorts:

Iterative sorts: Selection sort, Insertion sort, Bubble sort.

Recursive sorts: Merge sort, Quick sort.

Distribution sorts: Bucket sort.

Selection sort

```
void selectionSort(vector<int>& array) {
     for (int i = 0; i < array.size(); i++) {
          // Find min element from i to n-1
          for(int j = i + 1; j < array.size(); j++) {
               …
          }
          // Swap elements at index i and min elements
          …
     }
}
```

Time complexity is $O(n^2)$ in best, worst and average cases

Insertion sort

```
void insertionSort(vector<int>& array) {
      for (int i = 1; i < array.size(); i++) {
          for(int j = i; j >= 0; j--) {
               if(array.at(j) < array.at(j-1)) {
                    // swap array.at(j+1) and array.at(j)
               }
               else break;
          }
     }
}
```

Time complexity: $O(n^2)$ in worst and average cases and $O(n)$ in the best case.

THE UNIVERSITY of ADELAIDE

Bubble sort

```
void bubbleSort(vector<int>& array) {
     for (int i = array.size() - 1; i > 1; i--) {
          for(int j = 0; j < i; j++) {
          if(array.at(j) > array.at(j+1)) {
               // swap array.at(j) and swap array.at(j+1)
          }
     }
}
```

Time complexity: $O(n^2)$ in worst, best and average cases.

Comparison-based sorting algorithms

- Suppose we need to sort an array: a_1, a_2,…,$a_{n.}$

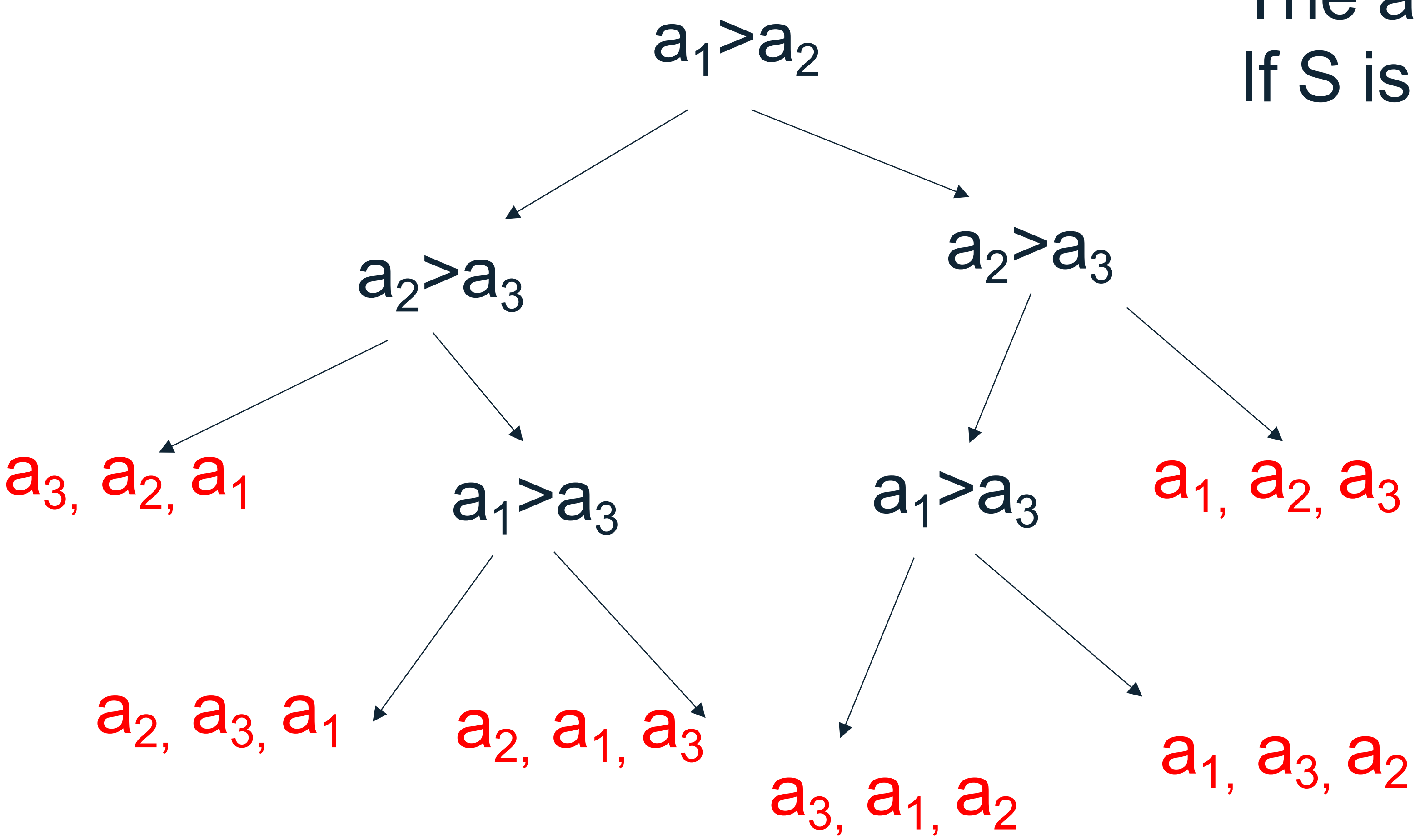
- If all the elements are distinct, there n! possible results of sorting (all possible permutations), but only one is correct!

- Each comparison-based sorting algorithm builds a decision tree. Let's consider such a tree.

THE UNIVERSITY of ADELAIDE

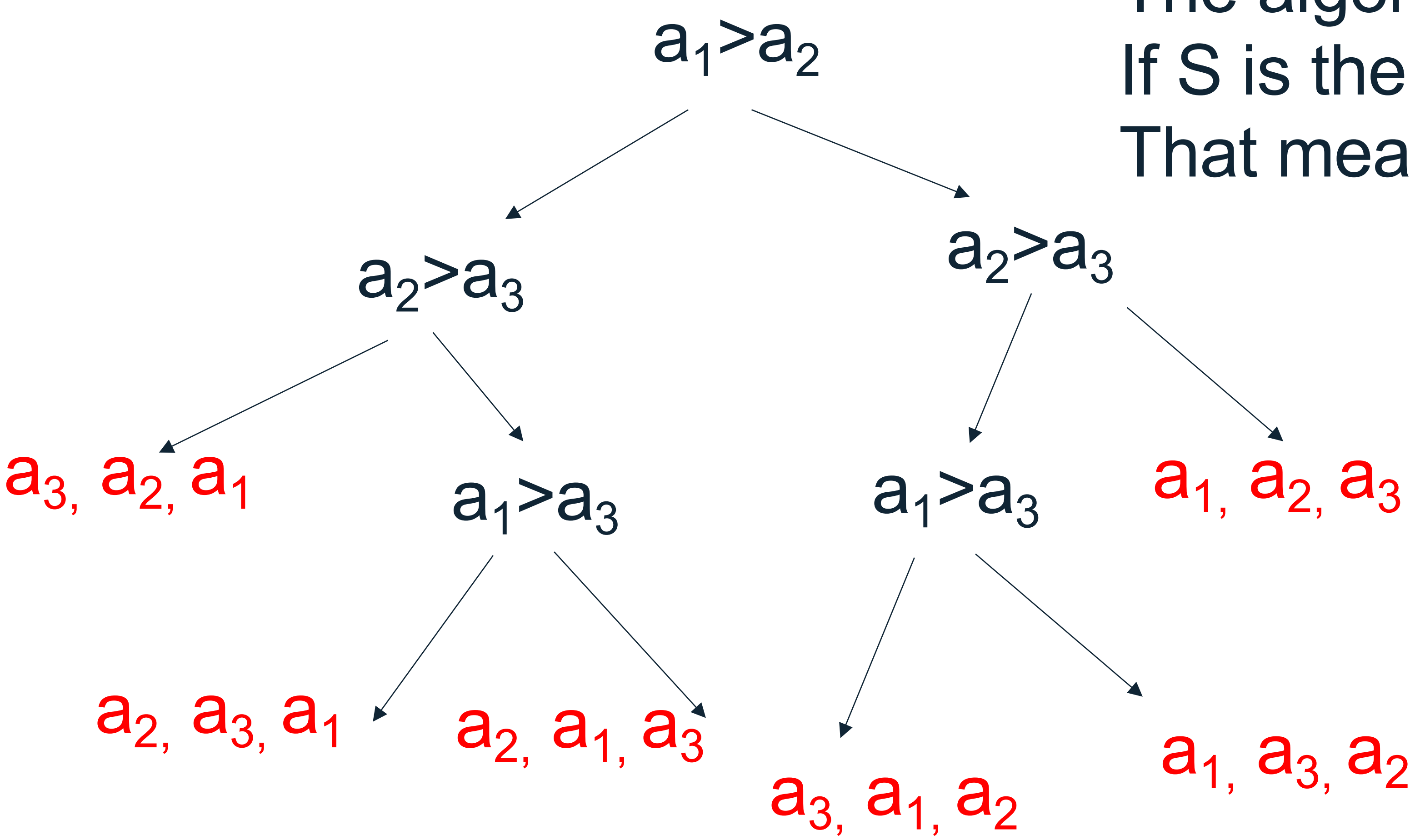
Comparison-based sorting algorithms

The algorithm selects a branch each time .
If S is the initial set of problems, we select S' > S/2.



The set of solutions is split each time.

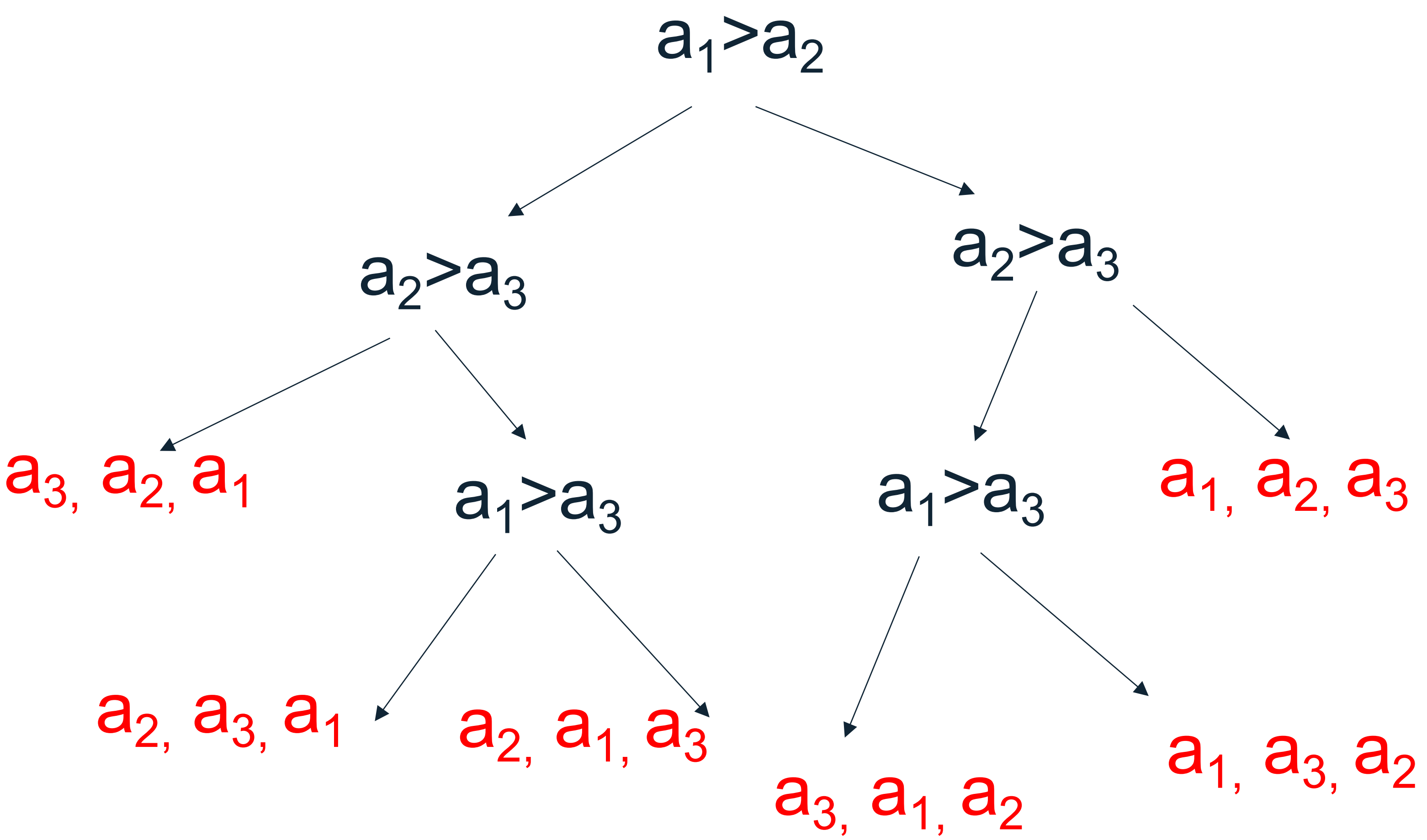
Comparison-based sorting algorithms



The algorithm selects a branch each time.
If S is the initial set of problems, we select S' > S/2.
That means we halving the set of solutions is the best option.

The number of leaves (possible solutions) is **n!**. Let the height of this tree (the number of comparisons) is $\mathbf{\log_2(n!)}$,
because we halve the set of solutions each time.

Comparison-based sorting algorithms



$\log_2(n!) = log_2(n) + log_2(n-1) + \cdots + log_2(2) \geq log_2(n) + log_2(n-1) + \ldots + \log_2\left(\frac{n}{2}\right) \geq nlog_2\left(\frac{n}{2}\right) = n(\log_2(n) - 1)$. Hence, $\log_2(n!) = \Omega(n\log(n))$.