Data Structures and Algorithms

Prof. Ganesh Ramakrishnan,
Prof. Ajit Diwan,
Prof. D.B. Phatak

Department of Computer Science and Engineering
IIT Bombay

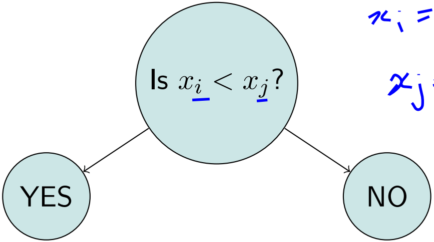
Session: Comparison Based Sorting

Outline

- Comparison based Sorting
- Abstract Data Types for Sorting
  - Selection, Insertion and Heap Sort
  - In-place sorting versions
  - Analysis
- Divide and Conquer Approach to Sorting
  - Merge-Sort and Quick-Sort
  - Analysis

Comparison based Sorting

- Sort by making comparisons between pairs of objects
- Result of each comparison $\Rightarrow$ Yes/No
- A bit later: Lower bound analysis for Comparison based Sorting



x_i = person 1 (age)

x_j = person 2 (age)

Abstract Data Types for Sorting

- Array-based in-place versions
- Loop Invariants
- Complexity Analysis
- Lower bounds on running time

Abstract Data Types for Sorting

- Priority Queue (PQ) is a Natural Choice
- Different implementations of the PQ lead to different sorting algos

Sorting using Priority Queue

Algorithm Sort-PQ(S)
Input: Sequence S
Output: Sequence S sorted in increasing order
Priority Queue: P
repeat
1. $e = front(S)$
2. $insert(e, P)$

until $empty(S)$
repeat
1. $m = indexofMin(P)$
2. $delete(P, m)$

until $empty(P)$

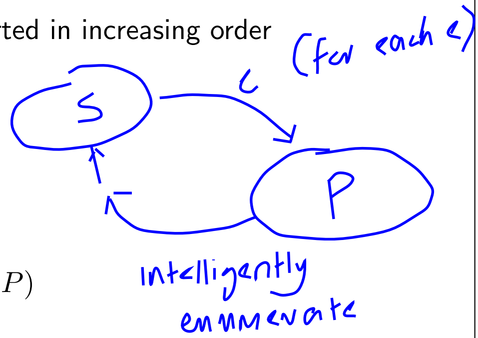


Figure: Sorting using n invocations of Priority Queue

Abstract Data Types for Sorting

- Priority Queue (PQ) is a Natural Choice
- Different implementations of the PQ lead to different sorting algos
  - Unsorted List-based $\Rightarrow$ Selection Sort
  - Sorted List-based $\Rightarrow$ Insertion Sort
  - Heap-based $\Rightarrow$ Heap Sort

Unsorted List-based (Selection Sort)

- $insert(e, P)$: Insert either at the beginning or at the end of the list P
  - $O(1)$
- $indexOfMin(P)$: Scan list P, find the min element and returns the index m
- $delete(P, m)$: Deletes element at position m from list P
  - $1 + 2 + \ldots n = O(n^2)$
- $\Rightarrow O(n^2)$ overall

Unsorted List-based (Selection Sort)

- $insert(e, P)$: Insert either at the beginning or at the end of the list P
  - $O(1)$
- $indexOfMin(P)$: Scan list P, find the min element and returns the index m
- $delete(P, m)$: Deletes element at position m from list P
  - $1 + 2 + \ldots n = O(n^2)$
- $\Rightarrow O(n^2)$ overall
- **Claim:** Best and worst case running times are $\Theta(n^2)$

$f(n) = O(g(n)) \quad \forall n \geq n_0 \quad f(n) \leq M g(n)$

$f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ &

$g(n) = O(f(n))$

Definition for List P: indexOfMin(P) = indexOfMin(P,0)

Algorithm indexOfMin(P,i) starting offset
Input: Sequence of numbers in list P
Output: Return the index of the minimum element in list P

1. $position\ m = i,\ q = i$
2. $min = element\_at(P, i)$
3. $next(q)$

repeat

  if $element\_at(P, q) < min$ **then**

   1. $min = element\_at(P, q)$
   2. $m = q$

  end if

  $next(q)$

until $element\_at(P, q)\ != undefined$

4. $return\ m$

Array-based in-place version of Selection Sort

Algorithm Selection-Sort(S)
Input: Sequence S
Output: Sequence S sorted in increasing order
$i = 1$, $n = length(S)$;
repeat
1. $m = indexOfMin(S, i)$
   //m stores index of min element of S
   //between positions i and n
2. $swap(i, m)$
3. $i = i + 1$

until $i = n - 1$
//Question: Why does it suffice to execute until $i = n - 1$?

Figure: In-place Selection Sort

Selection Sort and Loop Invariant

- At start of each iteration of i^{th} **repeat** loop, the subarray $S[1 \ldots i-1]$ consists of $i-1$ smallest elements of S in ascending order

$S[1 \cdots i-1]$ $S[\underbrace{i \ldots n}_{\text{scan}}]$

Loop Invariant & Algorithm Correctness

Three properties of loop invariant condition:

1. **Initialization:** It holds prior to the first iteration of the loop.
2. **Maintenance:** If it holds before an iteration of the loop, it continues to hold before the next iteration.
3. **Termination:** On termination, the invariant helps show that the algorithm is correct

Selection Sort and Loop Invariant

- At start of each iteration of i^{th} **repeat** loop, the subarray $S[1 \ldots i-1]$ consists of $i-1$ smallest elements of S in ascending order
- Claim: The above is a Loop Invariant for in-place Selection Sort

- Next Session: Sorted List Based $\Rightarrow$ Insertion Sort

Thank you