CMPSC 250
Analysis of algorithms

Dr. Aravind Mohan
Allegheny College

February 12, 2018
Lecture 09 (1) - Shell Sort
What is Shell Sort?

- It is a generalization of insertion sort.
- In this sorting algorithm, we compare elements that are distant apart rather than adjacent.
- In-place comparison sort.
- We start by comparing elements that are at a certain distance apart. So if there are N elements then we start with a value \textit{gap} < N.
- In each pass, we keep reducing the value of gap till we reach the last pass when gap is 1.
- In the last pass, shell sort is like insertion sort.
Some background about Shell Sort

- Invented by Donald L. Shell
Shell Sort Example

Unsorted array

| 14 | 18 | 19 | 37 | 23 | 40 | 29 | 30 | 11 |
Shell Sort Example

- First Pass: $gap_1 = \text{round}(N/2)$ \quad N = 9
- First Pass: $gap_1 = \text{round}(9/2)$
- First Pass: $gap_1 = \text{round}(4.5)$
- First Pass: $gap_1 = 4$
- Pairs: (0,4), (1,5), (2,6), (3,7), (4,8), (5,9)

|   11 |   18 |   19 |   30 |   14 |   40 |   29 |   37 |   23 |
Shell Sort Example

- Second Pass: \( gap_2 = \text{round}(gap_1/2) \) \( gap_1 = 4 \)
- Second Pass: \( gap_2 = \text{round}(4/2) \)
- Second Pass: \( gap_2 = \text{round}(2) \)
- Second Pass: \( gap_2 = 2 \)
- Pairs: (0,2), (1,3), (2,4), (3,5), (4,6), (5,7), (6,8), (7,9)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>18</td>
<td>14</td>
<td>30</td>
<td>19</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Shell Sort Example

- Third Pass: $gap_3 = \text{round}(gap_2/2)$  $gap_2 = 2$
- Third Pass: $gap_3 = \text{round}(2/2)$
- Third Pass: $gap_3 = \text{round}(1)$
- Third Pass: $gap_3 = 1$
- Pairs: (0,1), (1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9)

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>14</th>
<th>18</th>
<th>19</th>
<th>23</th>
<th>29</th>
<th>30</th>
<th>37</th>
<th>40</th>
</tr>
</thead>
</table>

Dr. Aravind Mohan  (Allegheny College)  CMPSC 250  February 12, 2018  8 / 18
Shell Sort Pseudocode

```
function SHELLSORT(an array a of length n)
    gap ← round(n/2)
    while gap > 0 do
        for i = gap to n-1 do
            temp ← a[i]
            j ← i
            while j ≥ gap and a[j - gap] > temp do
                a[j] ← a[j - gap]
                j ← j - gap
            end while
            a[j] ← temp
        end for
        gap ← round(gap/2)
    end while
end function
```
Complexity of Shell Sort

- Best-case: $O(N)$
- Average-case: $O(N \times \log_2(N))$
- Worst-case: $O(N \times \log_2(N))$
So far: Complexity of Sorting Algorithms

- Insertion Sort: Best-case: $O(N)$ ; Average-case: $O(N^2)$ ; Worst-case: $O(N^2)$
- Bubble Sort: Best-case: $O(N)$ ; Average-case: $O(N^2)$ ; Worst-case: $O(N^2)$
- Selection Sort: Best-case: $O(N^2)$ ; Average-case: $O(N^2)$ ; Worst-case: $O(N^2)$
Let us try another example

Unsorted array

| 15 | 19 | 20 | 38 | 24 | 41 | 30 | 31 | 12 |
Let us try another example

Unsorted array

| 12 | 19 | 13 | 30 | 35 | 18 | 24 | 22 | 14 |

Class activity: Solve it with your team, post your solution in the slack page.

Points awarded for participation credits.
Lecture 09 (2) - Quick Sort
Characteristics of Quick Sort

- sort almost in "place", i.e., does not require an additional array
- very practical, average sort performance $O(N \times \log_2(N))$, with small constant factors.
- worst case running time is $O(N^2)$
Some background about Quick Sort

- Invented by Tony Hoare in 1960
Quick Sort - the Principle

To understand quick-sort, let’s look at a high-level description of the algorithm.

A divide-and-conquer algorithm.

- **Divide:** partition array into 2 subarrays such that elements in the lower part ≤ elements in the higher part.
- **Conquer:** recursively sort the 2 subarrays.
- **Combine:** trivial since sorting is done in place.
function \textsc{Partition}($A$, $p$, $r$)

\begin{align*}
x & \leftarrow A[r] \\
i & \leftarrow p - 1 \\
j & \leftarrow r + 1
\end{align*}

\textbf{while} true \textbf{do}

\begin{align*}
\text{repeat} & \quad j \leftarrow j - 1 \\
\text{until} & \quad A[j] \leq x \\
\text{repeat} & \quad i \leftarrow i + 1 \\
\text{until} & \quad A[i] \geq x \\
\text{if} & \quad i \leq j \text{ then} \\
& \quad \text{exchange } A[i] \text{ and } A[j] \\
\text{else} & \\
& \quad \text{return } j
\end{align*}

\textbf{end if}

\textbf{end while}
Questions