CMPSC 250
Analysis of algorithms

Dr. Aravind Mohan
Allegheny College

February 14, 2018
Lecture 09 - 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 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

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

- First Pass: \( gap_1 = \text{round}(N/2) \)  \( N = 9 \)
- 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: $gap_2 = \text{round}(gap_1/2)$  \hspace{1cm} 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)

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

- Third Pass: $gap_3 = \text{round}(gap_2/2)$ \hspace{1cm} $gap_2 = 2$
- 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)

\[
\begin{array}{cccccccc}
11 & 14 & 18 & 19 & 23 & 29 & 30 & 37 & 40 \\
\end{array}
\]
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.
Questions