CMPSC250
Lecture 7: Shell Sort

Prof. John Wenskovitch
02/05/2016
Last Time

- Sort structure and “rules”
- Selection sort
- Insertion sort
- Comparing sorting methods
Shell Sort

```c
int h = 1;
while (h < N/3) {
    h = 3*h + 1;
} //while
while (h >= 1) {
    for (int i = h; i < N; i++) {
        for (int j = i;
             j >= h && less(a[j], a[j-h]);
             j -= h) {
            exch(a, j, j-h);
        } //for
    } //for
    h = h/3;
} //while
```
Shell Sort

- Values for h: 1, 4, 13, 40, 121, 364, 1093, ...
- In Insertion Sort, if the item with the smallest key is at the right end of the array, you need $N - 1$ exchanges to move it in place.
  - **Solution**: If two array items are far apart and definitely need to be swapped, why not just do one exchange instead?
  - Shell Sort improves over Insertion Sort by creating interleaved partially sorted arrays that can later be efficiently sorted by Insertion Sort.
Shell Sort Visual
Shell Sort Visual

input

40-sorted

13-sorted

4-sorted

result
Shell Sort Evaluation

• Sequence of decreasing values \((3k + 1)\).
  – Starts at the smallest increment \(\geq \lfloor n/3 \rfloor\) and decreasing to 1. This is called an increment sequence. (how do we pick a sequence to use?)

• When an h-sorted array is k-sorted, it remains h-sorted.

• Worst case for Shell Sort is \(O(n^{3/2})\).

• Average case... depends on the increment sequence selected.

• Practical improvement depends on array size.
Picking a Sequence

• No provably best sequence has been found.
  – Depends on the number of increments, arithmetical interactions among the increments (common divisors), etc.
  – *(could a best sequence exist?)*

• What we pick should be easy to compute and use.
  – Simple sequences can perform almost as well as more sophisticated sequences.
What Have We Learned?

• The simple increment sequence modification brought our run time down from $O(n^2)$ to $O(n^{3/2})$.
  – Finding things like this is a primary goal for many algorithm design problems.

• Shell Sort pros:
  – An acceptable run time for moderately large arrays. (Very good algorithms may only run twice as fast except for very large arrays.)
  – Doesn’t require much code.
  – No extra space required.
Space Evaluation

- How much extra space did Selection Sort require?
  - One temp variable for `exch()`

- How much extra space did Insertion Sort require?
  - One temp variable for `exch()`

- How much extra space did Shell Sort require?
  - One temp variable for `exch()`
  - One integer for `h`

- Did any of this extra space change complexity classes?
Any Questions?