Basic Sorting

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 4, 2017

Overview I

1. Intro
2. Selection sort
   - Analysis
   - Summary
3. Bubble Sort
   - Analysis
   - Summary
4. Insertion Sort
   - Summary

Intro

- **Time Complexity:**
  - worst/average/best cases
  - Order andruntime complexity function
  - Number of comparisons vs number of swaps
- **Space Complexity:**
  - Is it in place, this requires just a constant amount of memory.
- **Basis:**
  - Comparison (selection, insertion, bubble, merge, quick and heap sort).
  - Grouping (counting, bucket and radix).
- **Stability:**
  - Algorithm stable if the ordering of equal items in the original array is maintained in the sorted array.

Selection sort

1. Find the minimum value in the list.
2. Swap it with the value in the first position.
3. Repret the steps above for the for remainder of the list (starting in the second position).

```
for i = 1 to n-1:
  pos = i
  for j=i+1 to n:
    if T[j] < T[pos]
      pos = j
    endif
  endfor
  temp = T[pos]
  T[pos] = T[i]
  T[i] = temp
endfor
```

Analysis

- Fundamental operation: `T[j] < T[pos]`.
- Worst case: All cases the same.
- Complexity function:

$$
\begin{aligned}
t(n) &= \sum_{i=1}^{n-1}(\sum_{j=i+1}^{n}(1)) \\
&= \sum_{i=1}^{n-1}(n-(i+1)+1) \\
&= \sum_{i=1}^{n-1}(n-i) \\
&= n(n-1) - \sum_{i=1}^{n-1}(i) \\
&= n(n-1) - \frac{n(n-1)}{2} \\
&= \frac{n(n-1)}{2} \\
&= \frac{n^2}{2} - \frac{n}{2} \\
\Theta&(n^2)
\end{aligned} \tag{1}
$$

Summary

- **Basis:** Comparison
- **Time Complexity:**
  - Number of comparisons is $\Theta(n^2)$
  - Number of swaps is $\Theta(n)$.
- **Space Complexity:** Constant (in place).
- **Stability:** Not Stable

Bubble sort

1. Scan the array, swapping any out of order neighbouring elements.
2. Once the largest element is in the last position, the procedure is repeated to find the next largest, and so on.

```
for i=n to 1:
  for j=1 to i-1:
    if T[j] > T[j+1]
      SWAP(T[j],T[j+1]
    endif
  endfor
endfor
```

Analysis

- Fundamental operation: `T[j] > T[j+1]`.
- Worst case: All cases are the same.
- Complexity function:

$$
\begin{aligned}
t(n) &= \sum_{i=1}^{n}(\sum_{j=1}^{i-1}(1)) \\
&= \sum_{i=1}^{n}(i-1) \\
&= \sum_{i=1}^{n}(i) - \sum_{i=1}^{n}(1) \\
&= \sum_{i=1}^{n}(i-n) \\
&= \frac{n(n-1)}{2} - n \\
&= \frac{n^2}{2} + \frac{n}{2} - n \\
&= \frac{n^2}{1} - \frac{n}{2} \\
\Theta(n^2)
\end{aligned}
\qquad (2)
$$

Summary

- **Basis:** Comparison
- **Time Complexity:**
  - Number of comparisons is $\Theta(n^2)$.
  - Number of swaps is $\Theta(n^2)$.
- **Space Complexity:** Constant (in place sorting)
- **Stability:** Stable

Insertion Sort

1. A list size 1 is already sorted.
2. Insert the element in position 2 into the already sorted list.
3. From position 3 to n repreat the steps above

```
for i=2 to n:
  temp = T[i]
  j = i-1
  while j > 0 and temp < T[j]
    T[j+1] = T[j]
    j = j-1
  endwhile
  T[j+1] = temp
endfor
```

Summary

- **Basis:** Comparison
- **Time Complexity:**
  - Worst Case and Average Case:
    - Number of Comparisons: $O(n^2)$
    - Number of swaps: $O(n^2)$
  - Best case:
    - O(n).
- **Space Complexity:** Constant (in place)
- **Stability:** Stable.

The End