Selection Sort

Contents

- Description.
- Sort Algorithm.
- Example.
- Analysis.
- Advantages of Sorting Algorithms.
- Disadvantages of Sorting Algorithms.

Description

- This sorting algorithm is an in-place comparison-based algorithm .
- in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end.
- Initially, the sorted part is empty and the unsorted part is the entire list.

Description Contd.

- The Selection Sort is a combination of searching and sorting.
- In Selection Sort, sorting is done after selecting a particular smallest element from an array and shifting it to a particular location in an array.
- During each pass, the unsorted element with the smallest value is moved to its proper position in the array.

Description Contd.

- The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array.
- This process continues moving unsorted array boundary by one element to the right
- each time selecting an item according to its ordering and placing it in the correct position in the sequence.

The Selection Sort Algorithm

- For each index position i
  - Find the smallest data value in the array from positions i through length - 1, where length is the number of data values stored.

- Exchange (swap) the smallest value with the value at position i.

Algorithm

- **Step 1**– Set MIN to location 0
- **Step 2** – Search the minimum element in the list
- **Step 3** – Swap with value at location MIN
- **Step 4** – Increment MIN to point to next element
- **Step 5** – Repeat until list is sorted

Example

- Unsorted list:

5	2	1	4	3

- **1st iteration:**
  - Smallest = 5
  - $2 < 5$, smallest = 2
  - $1 < 2$, smallest = 1
  - $4 > 1$, smallest = 1
  - $3 > 1$, smallest = 1

Swap 5 and 1

1	2	5	4	3

Example contd.

- 2nd iteration:
  - Smallest = 2
  - 2 < 5, smallest =2
  - 2 < 4, smallest =2
  - 2 < 3, smallest =2
- No swap

1	2	5	4	3

Example contd.

- **3rd iteration:**
  - Smallest = 5
  - 4 < 5, smallest = 4
  - 3 < 4, smallest = 3

- **Swap 5 and 3**

1	**2**	**3**	**4**	**5**

Example contd.

- **4th iteration:**
  - Smallest = 4
  - 4 < 5, smallest = 4
- **No swap**

1	2	3	4	5

- **Finally, the sorted list is**

1	2	3	4	5

Selection Sort Example

Sorted Unsorted

23 78 45 8 32 56 Original List

8 78 45 23 32 56 ⟵ *After pass 1*

8 23 45 78 32 56 ⟵ *After pass 2*

8 23 32 78 45 56 ⟵ *After pass 3*

8 23 32 45 78 56 ⟵ *After pass 4*

8 23 32 45 56 78 ⟵ *After pass 5*

Selection sort

Example:
small-to-large sort

Selection Sort: Algorithm

```
void selection_sort (int arr[ ], int n)
{
        int i, j, smallest, temp;
        for (i = 0; i < n – 1 ; i++)
        {
                smallest = i;
            for (j = i + 1; j < n  ; j++)
                {
                if( arr[ j ] < arr[ smallest ];
                        smallest = j;
                }
        temp = arr[smallest];
        arr[smallest] = arr[i];
        arr[i] = temp;

        }
}
/* NOTE: Inner for loop, finds the next smallest value and the outer for loop places that
value into its proper location */
```

Selection Sort: Analysis

Complexity:

- Best case performance $O(n^2)$
- Average case performance $O(n^2)$
- Worst case performance $O(n^2)$

Selection Sort

swap

i/p = (77, 55, 12, 15, 20, 40, 5)
(1, 2, 3, 4, 5, 6, 7)

Pass ① $i=1$
min = ~~1~~ ~~2~~ ~~3~~ 7 Swap (a[i], a[min])

} n-1 comparison
only 1 swapping

(5, 55, 12, 15, 20, 40, 77)
(1, 2, 3, 4, 5, 6, 7)

Pass ② $i=2$
min = ~~2~~ 3 Swap (a[i], a[min])

(5, 12, 55, 15, 20, 40, 77)
(1, 2, 3, 4, 5, 6, 7)

n-2 comparison.
1 swap.

Swap
⇓
a = ~~10~~ 20, b = ~~20~~ 10
⇓
t = a
a = b
b = t
with temp

t = 10

CPU time is less.
(no operation)
~~more~~ ~~less~~ space

Swap
⇓
a = ~~10~~ ~~20~~ 20, b = ~~20~~ 10
⇓
a = a + b
b = a - b
a = a - b
without temp

CPU time is more

~~more~~ less space

Pass ③ $i=3$
min = ~~3~~ 4 Swap (a[i], a[min])

(5, 12, 15, 55, 20, 40, 77)
(1, 2, 3, 4, 5, 6, 7)

n-3 comparison
1 swap

Pass ④ $i=4$
min = ~~4~~ 5 Swap (a[i], a[min])

(5, 12, 15, 20, 55, 40, 77)
(1, 2, 3, 4, 5, 6, 7)

n-4 comparison.

Pass (5) $i = 5$

$min = \cancel{5}\ 6$ Swap $(a[i], a[min])$

5	12	15	20	40	55	77
1	2	3	4	5	6	7

$n-5$ comparison

1 swap

Pass (6) $i = 6$

$min = \cancel{6}\ 6$

5 12 15 20 40 55 (77)

no - comparison ! only one element.

$\therefore$ Total comparison

$$= 1 + 2 + 3 + \cdots + n-1$$

$$= \frac{(n-1)n}{2} = O(n^2)$$

Note

① Every Pass of Selection Sort will take maximum n comparison

② Total number of Pass in Selection Sort is $n-1$.

③ Time Complexity of Selection Sort is $\Theta(n^2)$ { Best Case, Avg Case, worst case all are same }

it does not bother whether element are sorted or not.

Date_____

(4) Total # of Swaps in Selection Sort is $n-1$.

{ 1 ~~pass~~ swap for each pass
and total pass $= n-1$

i.e in every pass, we are doing exactly 1 swap.

(5) It is a in-place Sorting algorithm.

→ no extra space. { temp is a variable
it will take $O(1)$ constant time }

Straight Selection Sort

Max^m element will go to last place accordingly.

Advantages & Disadvantages

Advantages

- The main advantage of the selection sort is that it performs well on a small list.
- no additional temporary storage is required.

Disadvantages

- The primary disadvantage of the selection sort is its poor efficiency when dealing with a huge list of items.
- the selection sort requires n-squared number of steps for sorting n elements.

Thank
you
DesiComments.com