ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC

Đề tài: Sorting Algorithms

Môn học: DSA

Sinh viên thực hiện:

TpG (23127244)

Giáo viên hướng dẫn:

BHT

Ngày 25 tháng 6 năm 2024

Mục lục

1 Algorithms Presentation

1.1 Selection sort

1.1.1 Ideas

The meaning of an ascending sorted array is that the smallest element is placed first, the second smallest element is placed second, ..., the largest element is placed last.

For the first position in the array, we find the smallest element and swap it with the first element. For the second position, find the second smallest and swap it with the second element in the array, and so on.

1.1.2 Descriptions

Define the $0th$ smallest element as the smallest element, the $1st$ as the second smallest,...

We browse and sort each element of n-element array (from 0 to $n-1$ respectively). When browsing to position i, we have the elements in the segment $[0, i-1]$ already sorted, the remaining elements are not sorted.

At this point, we need to find the ith smallest element to swap with element i. We have $i-1$ smallest element that is before i, so the ith smallest element of the array will be in the segment $[i, n-1]$ which is also the smallest element in the segment $[i, n-1]$.

Algorithm 1 Selection sort

```
1:  function selectionSort(array, n)
2:      for i ∈ [0, n − 2] do
3:          min_id ← i
4:
5:          for j ∈ [i + 1, n − 1] do
6:              if array[j] < array[min_id] then
7:                  mid_id ← j
8:              end if
9:          end for
10:
11:         temp ← array[i]
12:         array[i] ← array[min_id]
13:         array[min_id] ← temp
14:     end for
15: end function
```

1.1.3 Time complexity

The problem size is: n - the number of elements of the array.
Choose key operation (comparison) of the pseudocode is: $array[j] < array[min\_id]$.
Define $f(n)$ is the number of key time units. With $i = 0, 1, ..., n-2$, there are $n-1, n-2, ..., 1$ time unit respectively. So the total number of time units is

$$f(n) = (n-1) + (n-2) + ... + 1 = \frac{n(n-1)}{2} \approx \frac{1}{2}n^2$$

Time complexity: $O(n^2)$.

1.1.4 Space complexity

Since selection sort is an in-place sorting algorithm, it does not require additional storage.
Space complexity: $O(n)$.

1.2 Insertion sort

1.2.1 Ideas

Suppose we have a deck of cards in our hand, each card has an integer written on it. To sort the deck, we examine the cards one by one from left to right.
When checking the ith card, it means that the first $i-1$ cards have been sorted. So we just need to insert the ith card into the appropriate position in the first $i-1$ cards, then the first i cards will be sorted. Same with positions $i+1, i+2, ...$

1.2.2 Descriptions

To sort the array using insertion sort, we simulate checking and inserting cards from the above idea. When browsing to the ith element, we need to insert that element into the appropriate position in the first $i-1$ element. However, inserting elements is different from inserting cards. To place an element at position j, we need to shift elements $j, j+1, ...$ one position to the right before assigning the value to element j.
In turn checking and inserting elements from left to right, we will get a sorted array.

Algorithm 2 Insertion sort

```
1: function insertionSort(array, n)
2:     for i ∈ [1, n − 1] do
3:         key ← array[i]
4:         j ← i − 1
5:
6:         while j ≥ 0 and key < array[i] do
7:             array[j + 1] ← array[j]
8:             j ← j − 1
9:         end while
10:
11:        array[j + 1] ← key
12:    end for
13: end function
```

1.2.3 Time complexity

The problem size is: n - the number of elements of the array.

Choose key operation (comparision) of the pseudocode is: $key < array[i]$.

Define $f(n)$ is the number of key time units.

- **The worst case:** each element in the array needs to be inserted at the beginning. Specifically, the original array is sorted descending while we need to sort the array ascending.

  With $i = 1, 2, ..., n-1$, there are $1, 2, ..., n-1$ time unit respectively. So the total number of time units is

  $$f(n) = 1 + 2 + ... + (n-1) = \frac{n(n-1)}{2} \approx \frac{1}{2}n^2$$

- **The best case:** each element in the array does not need to be inserted into a different location. Specifically, the initial array is sorted.

  Each element need 1 time unit. So the total number of time units is

  $$f(n) = \underbrace{1 + 1 + ... + 1}_{n-1 \text{ times}} = n - 1$$

Time complexity (worst case): $O(n^2)$.

Time complexity (best case): $O(n)$.

1.2.4 Space complexity

Since insertion sort is an in-place sorting algorithm, it does not require additional storage.

Space complexity: $O(n)$.

Tài liệu

A Phụ lục

- Template này **không phải** là template chính thức của Khoa Công nghệ thông tin - Trường Đại học Khoa học Tự nhiên.
- Các hình ảnh, bảng biểu, thuật toán trong template chỉ mang tính chất ví dụ.
- Nhóm tác giả phân phối **miễn phí** template này trên GitHub và trên Overleaf với Giấy phép GNU General Public License v3.0. Nhóm tác giả không chịu trách nhiệm với các bản phân phối không nằm trong hai kênh phân phối chính thức nêu trên.