assignment2 report

学　　院：计算机科学与技术学院、软件学院

专　　业：软件工程（中外合作）

学生姓名：李肖

学　　号：202003340111

2022 年 11 月 26 日

1 算法解释

1.1 冒泡排序

冒泡排序的基本思想是：通过对待排序序列从前向后（从下标较小的元素开始），依次比较相邻元素的值，若发现逆序则交换，使值较大的元素逐渐从前移向后部，就象水底下的气泡一样逐渐向上冒。这样，每一趟会将最小的元素“浮”到顶端，最终达到完全有序。

1.2 归并排序

归并排序是建立在归并操作上的一种有效的排序算法，该算法是采用分治法将已有序的子序列合并，得到完全有序的序列；即先使每个子序列有序，再使子序列段间有序。

2 伪代码

2.1 冒泡排序

Algorithm 1 Bubble sort

Require: n， $a[1..n]$
Ensure: 排序后的序列 $a[1..n]$

```
1: for i ← 0; i < n; i++ do
2:     for j ← 0; j < n − i − 1; j++ do
3:         if A[j] > A[j+1] then
4:             swap (A[j], A[j+1])
5:         end if
6:     end for
7: end for
```

2.2 归并排序

Algorithm 2 Merge Sort

Require: n， $a[1..n]$
Ensure: 排序后的序列 $a[1..n]$

```
1: Function mergeSort(a[1..n])
2: if n > 1 then
3:     m ← ⌊n/2⌋
4:     a1 ← mergeSort(a[1..m])
5:     a2 ← mergeSort(a[m + 1..n])
6:     a ← merge(a1, a2)
7: end if
8: return a
9: END
```

3 理论时间分析

3.1 冒泡排序

$$f(n) = n^2 + n$$

3.2 归并排序

$$f(n) = n \log n + n$$

4 实验时间结果

```
n = 10 ^ 0
Bubble sort:

real   0m0.089s
user   0m0.001s
sys    0m0.001s

Merge sort:

real   0m0.091s
user   0m0.001s
sys    0m0.001s

----------------------------
n = 10 ^ 1
Bubble sort:

real   0m0.084s
user   0m0.001s
sys    0m0.001s

Merge sort:

real   0m0.092s
user   0m0.001s
sys    0m0.001s

----------------------------
n = 10 ^ 2
Bubble sort:

real   0m0.090s
user   0m0.001s
sys    0m0.001s

Merge sort:

real   0m0.091s
user   0m0.001s
sys    0m0.001s

----------------------------
n = 10 ^ 3
```

```
Bubble sort:

real   0m0.095s
user   0m0.008s
sys    0m0.001s

Merge sort:

real   0m0.094s
user   0m0.002s
sys    0m0.001s

----------------------------
n = 10 ^ 4
Bubble sort:

real   0m0.495s
user   0m0.411s
sys    0m0.002s

Merge sort:

real   0m0.090s
user   0m0.014s
sys    0m0.001s

----------------------------
n = 10 ^ 5
Bubble sort:

real   0m39.558s
user   0m39.078s
sys    0m0.115s

Merge sort:

real   0m0.670s
user   0m0.181s
sys    0m0.004s

----------------------------
```

5 回答问题

问题： At what input size do you consider the time required for initialization to be negligible in relation to the total running time of the algorithm?

回答： 初始化数据需要 n 次操作，在 n 比较小的时候 n^2 和 n^logn 和 n 的差距不大，需要考虑初始化数据的时间，但是当 n 很大的时候，初始化数据的时间可以忽略不计。

6 理论时间和实验时间的比较

可以看到在上面的时间中，归并排序的运行时间增长非常缓慢，而冒泡排序的运行时间在 n 成为 10^5 后增长非常快，这是因为归并排序的时间复杂度是 $n \log n$，而冒泡排序的时间复杂度是 n^2，所以归并排序的运行时间增长非常缓慢，而冒泡排序的运行时间增长非常快。

7 代码实现

7.1 冒泡排序

```cpp
#include <iostream>
#include <vector>

using std::vector;

void bubble_sort(vector<int> &arr) {
  for (int i = 0; i < arr.size(); ++i) {
    for (int j = 0; j < arr.size() - i - 1; ++j) {
      if (arr[j] > arr[j + 1]) {
        std::swap(arr[j], arr[j + 1]);
      }
    }
  }
}

int main() {
  int n;
  std::cin >> n;
  vector<int> arr(n, 0);
  for (int i = 0; i < n; ++i) {
    std::cin >> arr[i];
  }

  bubble_sort(arr);

  for (int i = 0; i < n; ++i) {
    std::cout << arr[i] << " ";
```

```
  }
  std::cout << std::endl;
}
```

7.2 归并排序

```
#include <iostream>
#include <vector>

using std::vector;

void merge_sort(vector<int> &arr) {
  if (arr.size() <= 1) {
    return;
  }

  int mid = arr.size() / 2;
  vector<int> left(arr.begin(), arr.begin() + mid);
  vector<int> right(arr.begin() + mid, arr.end());

  merge_sort(left);
  merge_sort(right);

  int i = 0, j = 0, k = 0;
  while (i < left.size() && j < right.size()) {
    if (left[i] < right[j]) {
      arr[k++] = left[i++];
    } else {
      arr[k++] = right[j++];
    }
  }

  while (i < left.size()) {
    arr[k++] = left[i++];
  }

  while (j < right.size()) {
    arr[k++] = right[j++];
  }
}

int main() {
  auto n = 0;
  std::cin >> n;
  vector<int> arr(n, 0);
```

```
  for (auto i = 0; i < n; ++i) {
    std::cin >> arr[i];
  }
  merge_sort(arr);

  for (auto i = 0; i < n; ++i) {
    std::cout << arr[i] << " ";
  }
  std::cout << std::endl;

  return 0;
}
```