算法第四次作业

10205101530-赵晗瑜

8.1-3 线性复杂度

8.1-3 线性复杂度

Show that there is no comparison sort whose running time is linear for at least half of the $n!$ inputs of length n. What about a fraction of $1/n$ of the inputs of length n? What about a fraction $1/2^n$?

对于给定的 m 种输入，设决策树树高为 h，由于叶节点数目不多于 2^h

则有：$m \le 2^h$，则 $h \geq \lg m$

① 当 $m = \frac{n!}{2}$ 时，$h \geq \lg \frac{n!}{2} = \Omega(n \lg n)$

② 当 $m = \frac{n!}{n}$ 时，$h \geq \lg \frac{n!}{n} = \lg (n-1)! = \Omega(n \lg n)$

③ 当 $m = \frac{n!}{2^n}$ 时，$h \geq \lg \frac{n!}{2^n} = \lg(n!) - \lg 2^n = \lg(n!) - n\lg 2$

因此三种情况都不能在线性时间内达到

8.3-4 基数排序

8.3-4

Show how to sort n integers in the range 0 to $n^3 - 1$ in $O(n)$ time.

因为$range \in [0..n^3-1]$,要排序的数字较大，k!=O(n)，因此不能直接使用计数排序。

可以做一次转化：把这n个数看做n进制的数，那么只需要3位即可对[0..n^3]的数进行排序

```
LINEAR_SORT(A):
    // convert the array into base n so that it has most 3 digits
    let B = CONVERT_TO_BASE_N(A)
    for i = 1 to 3
        // use counting sort to sort the ith digit of the array
        COUNTING_SORT(A, i)
```

可知时间复杂度为O(n)。

8.4-2 桶排序

8.4-2

Explain why the worst-case running time for bucket sort is $\Theta(n^2)$. What simple change to the algorithm preserves its linear average-case running time and makes its worst-case running time $O(n \lg n)$?

[译]

解释为什么桶排序在最坏情况下运行时间是$\theta(n^2)$？我们该如何修改算法，使其在保持平均情况为线性时间代价的同时，最坏情况下时间代价为O(nlgn)?

[ans]

因为在最坏情况下，输入并不呈均匀分布，而是集中在一个桶中，因此插入排序的时间复杂度为 $O(n^2)$, 总的时间复杂度为$O(n^2)$。

可以通过改变排序算法来改善最坏情况的复杂度，使得桶排序在最坏情况下运行时间是 $nlg(n)$。如，我们在相同的桶中采用归并排序或快速排序来对其进行排序。