算分作业 8

24.05.06

8.1

(1) 设两张图的邻接矩阵分别为 (a_{ij}) 和 (b_{ij}). 对每个 $1 \le i, j \le n$, 检查是否有 $a_{ij} \ne b_{ij}$, 若都不等则两张图互为补图, 若有相等则不是补图. 显然这是 $\Theta(n^2)$ 的.

(2) 对于互补的两张图, 其互补性取决于所有 a_{ij} 和 b_{ij}, 它们必须都被读取, 否则可以改变未被读取的元素, 使之不互补. 因此, 若要证明两张图互补, 必须读取邻接矩阵的所有元素, 即任何算法都是 $\Omega(n^2)$ 的, 由于 (1) 已经给出了 $\Theta(n^2)$ 的算法, 这个问题的复杂度就是 $\Theta(n^2)$.

8.2

(1) 设 $P_n(x) = a_n$, 且 $P_m(x) = P_{m+1}(x)x + a_m, 0 \le m < n$. 显然 $P(x) = P_0(x)$. 由于单次递推是 $O(1)$ 的, 从 $P_n(x)$ 开始递推得到 $P_0(x)$ 是 $\Theta(n)$ 的.

(2) 多项式的每个系数都影响在非零点处的值, 因此都必须被读取, 这已经需要 $\Theta(n)$ 的时间, 因此任何算法都是 $\Omega(n)$ 的.

8.4

记 $m = n/k$.

(1) $O(2m) + O(3m) + \cdots + O(km) = O(k^2 m) = O(kn)$.

(2) 将数表两两分组, 分别组内归并, 轮空的不变. 继续下去, 直到只剩一个数表. 注意到单次归并时每个元素的代价是 $O(1)$ 的, 而每个元素至多经历 $\log k$ 次归并, 因此这个算法是 $O(n \log k)$ 的.

(3) 任何基于比较的算法都天然对应一颗决策树, 不同叶节点对应不同输入. 由于输入互不相同, 这棵树为二叉树. 输入共有 $\frac{n!}{(m!)^k}$ 种, 因此树深至少是 $\Theta(\log \frac{n!}{(m!)^k}) = \Theta(\log n! - k \log m!) = \Theta(n \log n - km \log m) = \Theta(n \log k)$, 即任何基于比较的算法都是 $\Omega(n \log k)$ 的.

8.7

(1) 伪代码如下. 对 `A[1..n]` 调用 `Solve` 即得答案.

```
Solve:
Input: A[l..r]
Output: i
i = floor((l + r) / 2)
if A[i] == i: return i
else if A[i] < i: return Solve(A[i+1..r])
else: return Solve(A[l..i-1])
```

每经过 $O(1)$ 的计算可使输入规模减少一半, 因此这个算法是 $O(\log n)$ 的.

(2) 这里只考虑基于比较的算法. 任何这类算法都天然对应一颗决策树, 不同叶节点对应不同输入, 且为三叉树. 由于我们只关注使 `A[i] == i` 的 `i`, 叶节点共有 n 个, 因此树深至少是 $\Theta(\log_3 n) = \Theta(\log n)$, 即任何这类算法都是 $\Omega(\log n)$ 的.

8.10

使用计算选中位数的复杂度下界时相同的方法即可.

设第 k 小的数为 t, 按照算法对元素的比较顺序进行赋值. 若比较的元素 x, y 都未被赋值, 对其赋值使 $x < t < y$; 若一者已被赋值, 则将另一者赋值为 t 另一侧的任意值; 若都被赋值, 不动. 继续下去, 直到已有 $k-1$ 个元素得到小于 t 的值, 或 $n-k$ 个元素得到大于 t 的值, 此时将剩余元素赋为 t 另一侧的任意值, 并结束构造.

在构造的该输入上, 算法至少比较 $n-1+\min\{k-1, n-k\} = n + \min\{k-1, n-k+1\} - 2$ 次.