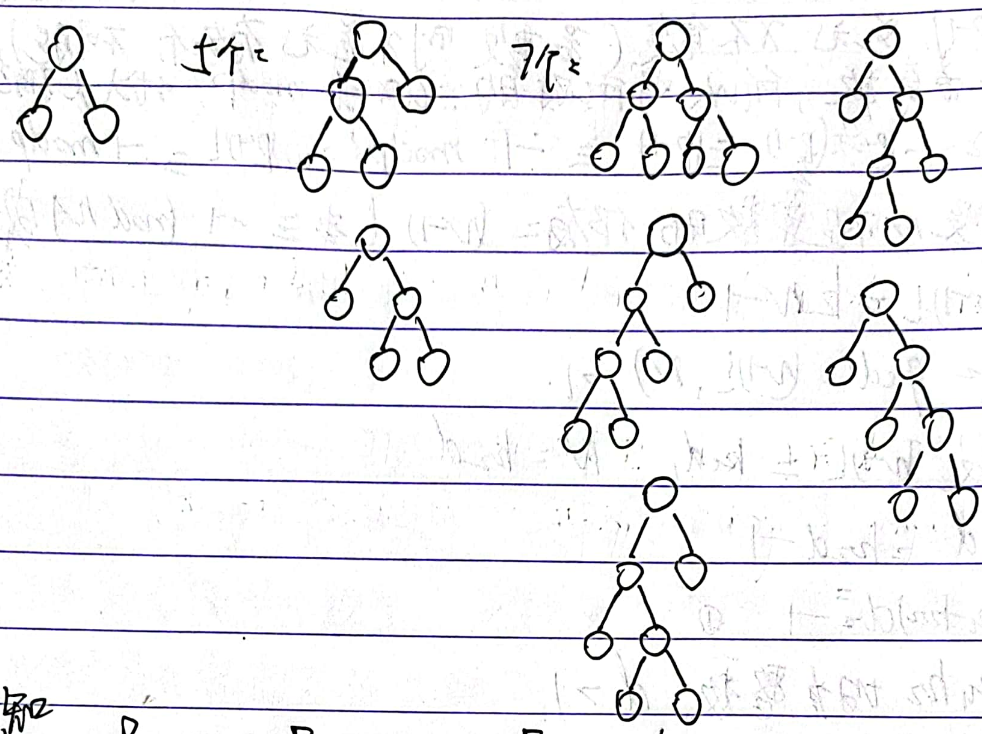
作业2：

2.13：

解：

(a) ① 3个：　　5个：　　7个：



② 由上图得知 $B_3=1$，$B_5=2$，$B_7=5$

③：对于一个满二叉树，其必定有一个根节点($B_1=1$)，而之后每一层节点数均为2^k，为偶数，故节点总数不可能为偶数

(b)：对于B_n，其代表有n个节点的满二叉树，
∴对除去根节点之外节点(n-1)，
只需保证两端数量和为(n-1)且均为满树即可

∴ $B_n = \sum_{i=1,\,i为奇数}^{n-2} B_i B_{n-1-i}$

(c)：由(b)得知 $B_n=\sum_{i=1,\,i为奇数}^{n-2} B_i B_{n-1-i}$，

∵ $B_1=1$，∴ $B_1 = 2^{\Omega(1)}$

设 $B_k = 2^{\Omega(k)}$ 成立

则 $B_{k+2} = \sum_{i=1,\,i\le k+1}^{} B_i B_{n-1-i} \geq \left(\frac{k-1}{2}+1\right)\cdot 2^{k-1} = (k+1)\,2^{k-2}$

$= \frac{k+1}{4}\, 2^k \geq 2^k$ ($k\geq 3$，且k为奇)

∴ B_{k+2}

则 $B_{2k+2} = \sum_{i=1, \text{i为奇}}^{k} B_i B_{2k+1-i} \geq \sum_{i为奇} (\frac{k+1}{2}+1) 2^{k+1} = (k+1) 2^{k+1}$

$= 2^{k+1+\log_2 (k+1)}$ ∵ $k \geq 3$ ∴ $\log_2(k+1) \geq 2$ ∴ $B_{2k+2} \geq 2^{k+2}$

∴ $B_{2k+2} = 2^{\Omega(k+2)}$ ∴ 归纳证得 $B_n = 2^{\Omega(n)}$

2.4.4：

(a)：由归并排序中2-3节中的 merge 过程可知，其时间复杂度为 $O(n)$，其中 n 为合并后数组总长度。

∴ 采用此方法的时间复杂度为：

$T(n) = O(2n) + O(3n) + O(4n) + \cdots O((k-1)n + n)$

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

~~(b)：对于给定的 k 个数组进行二分先合并前 $\lfloor \frac{n}{2} \rfloor$ 个，再合并后 $\frac{n}{2}$~~

(b)：先合并前 $\lfloor \frac{k}{2} \rfloor$ 个数组，再合并后 $\lceil \frac{k}{2} \rceil$ 个数组，之后将两部分结果合并，（递归合并每一部分）

分析：(b)中算法时间复杂度为：~~$O(kn \log kn)$，故优于 $O(k^2 n)$~~

$T(k) = 2 \cdot T(\lfloor \frac{k}{2} \rfloor) + O(kn)$

∴ 由 Master Theorem 有：$T(k) = O(nk \log kn)$ （∵ $a=b=2$，$\log_b a = 1$

∴ 优于 (a) 中的 $O(n \cdot k^2)$ ∴ 更有效

2.4.22：

解：假设 m、n 中有序，均为升序，时要求的时间复杂度为 $O(\lg m + n)$

故一定不能先合并 m、n 再找，需分别对 m、n 进行二分，设分别对 m、n 进行二分，设其左右边界指针分别为：l_m, r_m, l_n, r_n

①：若 $k=1$，则比较 $m[l_m]$ 与 $n[l_n]$，较小者即为结果（递归终止条件）

设递归函数为：$F(m, n, l_m, r_m, l_n, r_n, k)$

②：$k>1$：找 $mid_m = \lfloor \frac{l_m + r_m}{2} \rfloor$，$mid_n = \lfloor \frac{l_n + r_n}{2} \rfloor$，只四分以下情况：

i. 若 $m[mid_m] \leq n[mid_n]$：

s、若 $midm < k$，则调用 $F(m, n, midm, rm, ln, rn, k-\frac{midm}{2})$，

(即此时 m的前 midm个元素均在第k个前面，故二分右半部分)

ss、若 $midm > k$，则调用 $F(m, n, lm, midm-1, ln, rn, k)$

ii、若 $m[midm] > n[midn]$：

s、若 $midn < k$，调用 $F(m, n, lm, rm, midn, rn, k-midn)$；

ss、若 $midn > k$，调用 $F(m, n, lm, rm, ln, midn-1, k)$

一直递归到 $k=1$ 比较后结束。

由于每次二分均在 m, n 上各自进行，m, n 各自划分时间为 $O(\log m)$，$O(\log n)$，

故总时间复杂度为 $O(\log mn)$，即为 $O(\log m + \log n)$

2-28：1中：

取 $v = (v_1 \ldots v_{2^{k-1}} | v_{2^{k-1}+1} \ldots v_{2^k})^T = (v_1 | v_2)^T$

$$\therefore H_k v = \begin{pmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} H_{k-1}(v_1+v_2) \\ H_{k-1}(v_1-v_2) \end{pmatrix}$$

其中 v_1, v_2，均为长度为 2^{k-1} 的列向量

∴ 上式实际上就是一种对于运算长度的二分法。

设总时间为 $T(n)$.

∵ 对每个结果需要额外一次 +/- 法 且 +、- 为 $O(1)$

故合并的额外时间为 $O(n)$

$\therefore T(n) = 2 \cdot T(\frac{n}{2}) + O(n)$

$\therefore a = b = 2, d = 1 \quad \therefore \log_b a = d$

∴ 由 Master Theorem 有：$T(n) = O(n \log n)$

扫描全能王 创建