排序网络

华东师大二附中 符文杰

过去，我们学习了许多关于串行计算机的排序算法（如堆排序、快速排序），这种类似的计算机每次只能执行一个操作。而今天，我所要介绍的排序算法是基于计算上的一种排序网络模型的基础之上的。在这种网络模型中可以同时执行多个比较操作。

首先，我们要熟悉几个概念。

排序网络是总能对其输入进行排序的***比较网络***。

一个***比较网络***仅由***比较器***和***线路***构成。

比较器是具有两个输入 x 和 y 以及两个输出 x'和 y'的一个装置且执行下列函数：

x'=min(x,y)

y'=max(x,y)

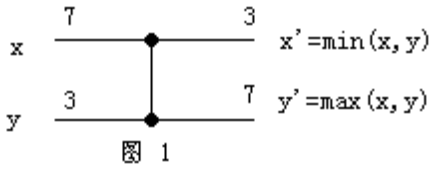


图 1

我们通常把比较器画为一竖垂直线，如图 1 所示。输入在左面，输出在右面，较小的输入值在输出端的上部，较大的输入值在输出端的下部。因此，我们可以认为比较器对其两个输入进行了排序。

我们假定每个比较操作占用的时间为 O(1)。换句话说，我们假定自出现输入值 x 和 y 到产生输出值 x'和 y'之间的时间为常数。

线路把一个值从一处传输到另一处。它可以把一个比较器的输出端与另一个比较器的输入端相连，在其他情况下它要么是网络的输入线，要么是网络的输出线。

比较网络就是一个由线路互相联接着的比较器的集合，我们把具有 n 个输入的比较网络画成一个由 n 条水平线组成的图，比较器则垂直地

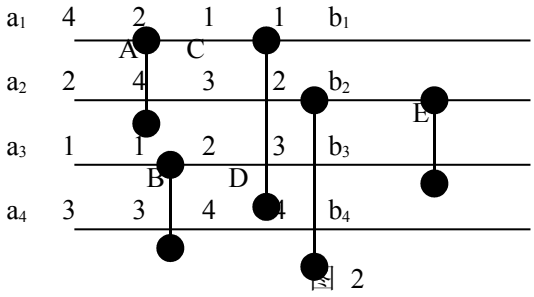


图 2

与两条水平线相连接。每个比较器的输入端要么与网络的 n 条输入线路 $a_1,a_2,\ldots\ldots,a_n$ 中的一条相连，要么与另一个比较器的输出端相连接。类似地，每个比较器的输出端要么与网络的 n 条输出线路 $b_1,b_2,\ldots\ldots,b_n$ 中的一条相连，要么与另一个比较器的输入端相连接。互相连接的比较器主要应满足如下要求：其互相连接所成的图中必须没有回路。

只有当同时有两个输入时，比较器才能产生输出值。

在每个比较器均运行单位时间的假设下，我们可以对比较网络的“运行时间”作出定义，这就是从输入线路接收到其值的时刻到所有输出线路收到其值所花费的时间。

排序网络是指对每个输入序列其输出序列均为单调递增（即 $b_1\le b_2\le\ldots\le b_n$）的一种***比较网络***。当然并非每个比较网络都是排序网络，不过图 2 中的比较网络是排序网络。这个不难证明。

比较网络和过程的相似之处在于它指定如何进行比较，其不同之处在于其实际规模决定于输入和输出的数目。因此，我们实际是在描述比较网络的家族。我们的目标就是寻找一个关于有效排序网络的家族排序程序***SORTER***。在家族 SORTER 中具有 n 个输入和 n 个输出的排序网络定义为***SORTER[n]***。

在寻找这样的 SORTER 之前，我们首先应该明确如何去判断一个比较网络

是否是排序网络。根据排序网络的定义，排序网络是对每个输入序列其输出序列均为单调递增的比较网络。根据这一点，如果我们要判断一个有 n 个输入和 n 个输出的比较网络是排序网络，就必须测试 n！种输入序列。这个工作量是非常大的，是否有简单一些的判断方法呢？

经过研究，我们发现了***0-1 原则***。0-1 原则认为：如果对于属于集合{0，1}中的每个输入值（输入序列中的每个数是 0 或 1），排序网络都能正确运行，则对任意的输入值，它也能正确运行。这样就把原来的问题变地简单一些了。

下面让我们来证明 0-1 原则的正确性。我们先要证明一个引理。

引理 1：如果比较网络把输入序列 $a=<a_1,a_2,\ldots\ldots,a_n>$转化为输出序列 $b=<b_1,b_2,\ldots\ldots,b_n>$，则对任意单调递增函数 f（不一定严格单调递增），该网络把输入序列 $f(a)=<f(a_1), f(a_2),\ldots\ldots,f(a_n)>$转化为输出序列 $f(b)=<f(b_1),f(b_2),\ldots\ldots,f(b_n)>$。

证明：我们可以采用数学归纳法来证。对比较网络中比较器的个数 m 进行归纳。

证明 m=1 时引理 1 成立。（即只有一个比较器的情况）

设这个比较器连接 ai 和 aj。

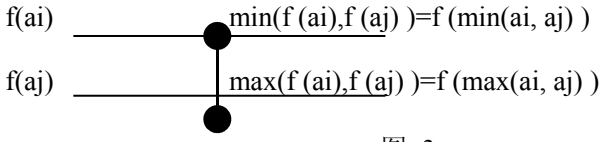


图 3

如果输入值为 ai 和 aj，那么该比较器上端输出为 min(ai,aj)，下端输出为 max(ai,aj)。如果我们把 f(ai)和 f(aj)作为该比较器的输入，由图 3 所示，则该比较器上端输出为 min(f(ai),f(aj))，下端输出为 max(f(ai),f(aj))。由于 f 是单调递增函数，若 ai≤aj，则 f(ai)≤f(aj)。因此我们有下列等式：

min(f(ai),f(aj))=f(min(ai,aj))
max(f(ai),f(aj))=f(max(ai,aj))

所以当我们把 f(ai)和 f(aj)作为比较器的输入时，它所得出的值就是 f(min(ai,aj))和 f(max(ai,aj))。所以在 m=1 时，引理 1 成立。

当比较器的个数 m<k(k≥2)时，引理 1 均成立。考虑 m=k 时的情况。

我们可以把具有 k 个比较器的比较网络拆成两个比较网络。前 k-1 个比较器构成一个比较网络 S1，最后一个比较器单独构成一个比较网络 S2。S1 和 S2 中比较器的个数均小于 k 个。原来的比较器就可以看作是 S1 和 S2 的串联，比较器 S1 的输出序列就是比较器 S2 的输入序列。

不妨设输入序列 a=<a_1,a_2,……,a_n>经过比较器 S1 后，成为序列 c=< c_1,c_2, ……,c_n>，再把它作为输入序列经过比较器 S2 后，成为最终的输出序列 b=< b_1,b_2,……,b_n >。

那么由归纳假设知：如果把 f(a)=<f(a_1), f(a_2),……,f(a_n)>作为比较器 S1 的输入序列，输出即为 f(c)=<f(c_1), f(c_2),……,f(c_n)>；再把 f(c)=<f(c_1), f(c_2), ……,f(c_n)>作为比较器 S2 的输入序列，得到最终的输出序列为 f(b)=<f(b_1),f(b_2),……,f(b_n) >。

因此，引理 1 在 m=k 时亦成立。

综上所述，无论比较网络中有多少个比较器，引理 1 均成立。
证毕。

有了引理 1 作为基础，我们就可以证明 0-1 原则了。

0-1 原则：如果一个具有 n 个输入的比较网络能够对所有可能存在的 2^n 个 0 和 1 组成的序列进行正确的排序，则对所有任意数组成的序列，该比较网络也可

能对其正确排序。

证明：我们用反证法来证明。假设这个比较网络能对所有 0-1 序列进行排序，但存在一个由任意数组成的序列，网络不能对该序列正确地排序。这就是说，存在

$$f(x)=\begin{cases}0 & \text{如果} x \le ai \\ 1 & \text{如果} x > ai\end{cases}$$

一个输入序列，其中两个元素 ai 和 aj 满足 ai<aj，但在输出序列中 aj 被排在 ai 之前。我们定义一个单调递增函数 f 为：

因为当 a=<a_1,a_2,……,a_n>作为输入序列时，其输出序列中 aj 位于 ai 之前，所以由引理 1 可知当序列 f(a)=<f(a_1), f(a_2),……,f(a_n)>作为输入序列时，其输出序列中 f(aj)必位于 f(ai)之前。但由于 f(aj)=1，f(ai)=0。这样我们就推出网络不能正确地对 0-1 序列<f(a_1), f(a_2),……,f(a_n)>进行排序，与假设矛盾。

0-1 原则得证。

当我们构造排序网络和其它比较网络时，0-1 原则使得我们可以把注意力集中于对仅由 0 和 1 组成的输入序列进行相应操作。一旦我们构造好排序网络并证明它能对所有的 0-1 序列进行排序，我们就可以运用 0-1 原则说明它能对任意值的序列进行正确的排序。

0-1 原则虽然可以帮助我们更容易地判断一个比较网络是否是排序网络，但是我们必须构造出这样的一类排序网络。

对于一个有 n 个输入和 n 个输出的***比较网络 SORTER[n]***，如果想要一次性地构造出来，难度会比较大。所以，我们可以考虑用采用**二分法**的思想。先把 n 个元素分成上下两半，分别用 ***SORTER[n/2]*** 对它们进行排序，再用一个

MERGER[n]的比较网络把它们合并成一个递增序列。我们可以用同样的方法来解决 ***SORTER[n/2]***。这样，问题的关键又变成了如何构造一类比较网络 ***MERGER[n]***，能够把两个含有 n/2 个元素的递增序列合并成一个递增序列。

如果我们把一个有序序列由小到大、另一个有序序列从大到小接在一起，就构成了一个***双调序列***。所谓双调序列是指序列要么先单调递增然后再单调递减，要么先单调递减然后又单调递增。例如序列<1,4,6,8,3,2>和<9,8,3,2,4,6>都是双调的。双调的 0-1 序列的结构比较简单。其形式为 $0^i1^j0^k$ 或 $1^i0^j1^k$，其中 i,j,k≥0。双调序列的一些特性可以帮助我们构造出***双调排序网络***。

双调排序程序由 $\log_2 n$ 个阶段组成，其中每一个阶段称为一个***半清洁器***。每个半清洁器是一个深度为 1 的比较网络，其中输入线 I 与输入线 I+n/2 进行比较，I=1,2,…,n/2（这里假设 n 为偶数）。图 4 即为一个具有 8 个输入和 8 个输出的半清洁器。

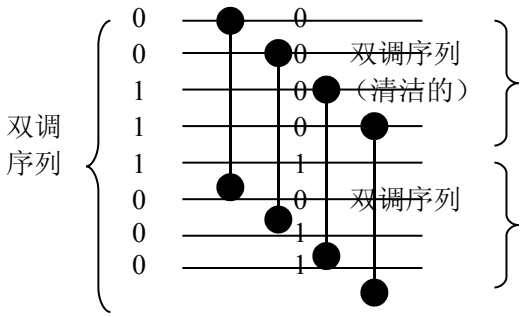


图 4

当由 0 和 1 组成的双调序列用作半清洁器的输入时，半清洁器产生的输出序列满足如下条件：

较小的值位于输出的上半部，较大的值位于输出的下半部。

两部分序列仍是双调的。

两部分序列中至少有一个是清洁的——全由 0 或 1 组成。（它的名称也是由此而来）

下面就来证明这些性质。

定理 1：如果一个半清洁器的输入是一个由 0 和 1 组成的双调序列，则其输出

满足如下性质：输出的上半部分和下半部分都是双调的，上半部分输出的每一个元素至少与下半部分输出的每一个元素一样小，并且两部分中至少有一个部分是清洁的。

证明：不妨设输入形如：00…011…100…0（输入为 11…100…011…1 的情形与上述情形是对称的）。根据序列的中点 n/2 落在序列中连续的“0”段或连续“1”

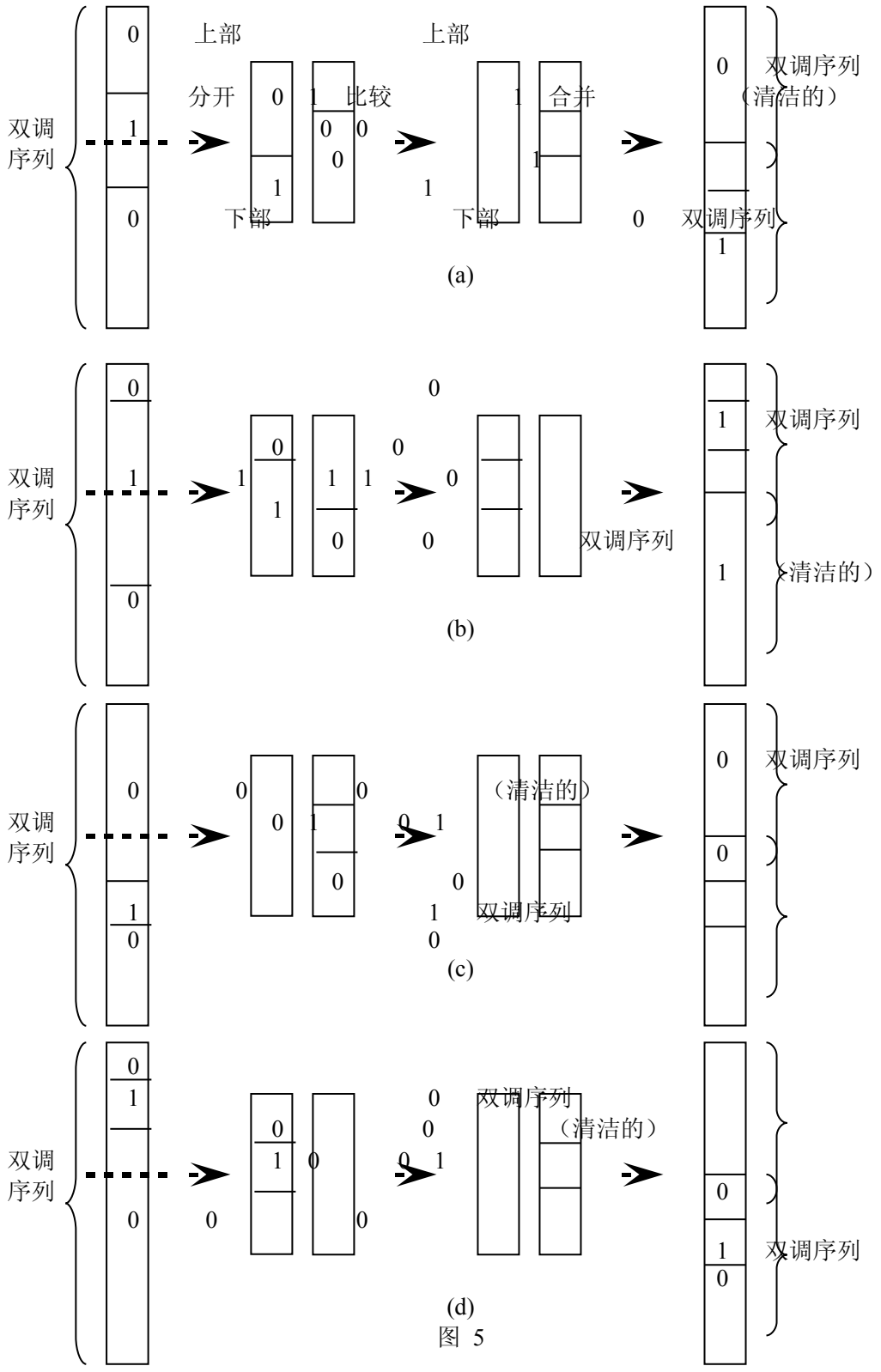


图 5

段可分为三种情况，并且这些情况中有一种（中点处于连续的“1”段的情形）又可继续分为两种情况。图 5 分别说明了这四种情形。在每一种情形下，引理都成立。

通过递归地连接半清洁器，我们可以建立一个双调排序网络。***双调排序网络[n]***的第一阶段由***半清洁器[n]***组成，由定理 1 可知***半清洁器[n]***产生两个规模缩小一半的双调序列且满足上半部分的每个元素不比下半部分的任一个元素大。因此，我们可以运用两个***双调排序网络[n/2]***分别对两部分递归地进行排序，以此完成整个排序工作，如图 6 所示。

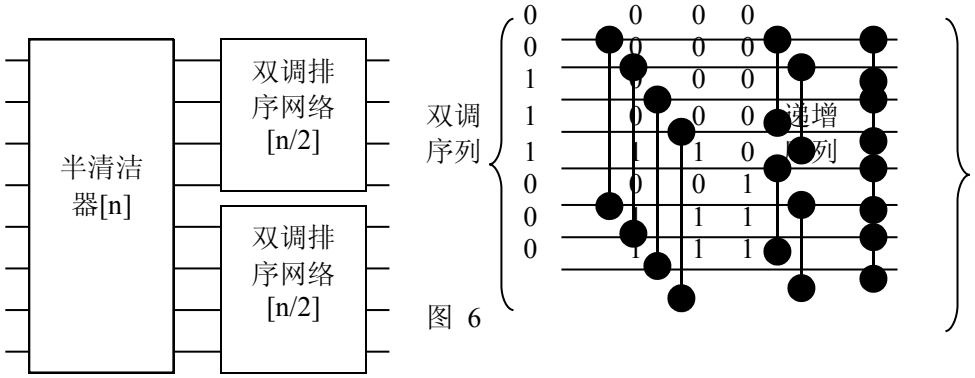


图 6

很明显，双调排序网络的深度为 $\log_2 n$。

我们只要把含 n 个元素的双调排序网络的第一个半清洁器修改一下就可以得到***合并网络 MERGER[n]***。由于输入的上半部和下半部都是单调递增的，所以我们把比较网络的下半部分颠倒一下，输入就成了一个双调序列。添上半清洁器，

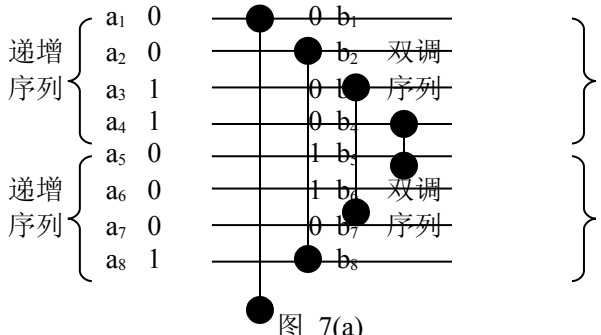


图 7(a)

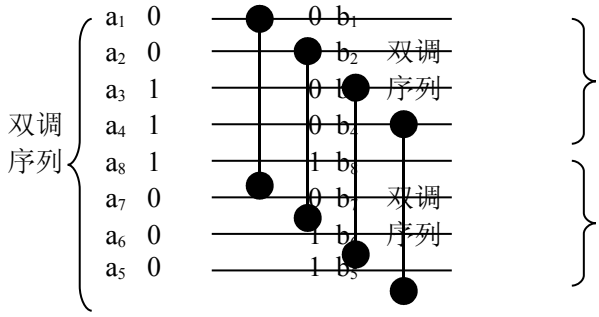
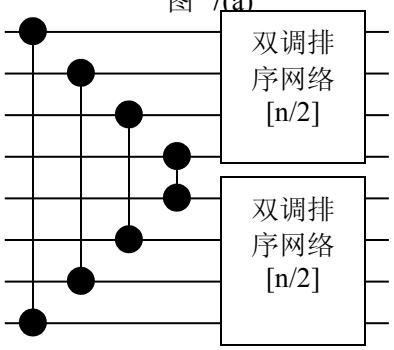


图 7(b)



再颠倒回去，半清洁器就变成了把输入 a_i 和 a_{n-i+1} 比较。这时，输出也被颠倒了。但是，一个双调序列颠倒了以后还是一个双调序列。所以，这种修改过的半清洁器仍然满足**定理 1** 中的性质。

图 8 为 ***MERGER[n]***的一般构造方法以及 n=8 时的具体情况。

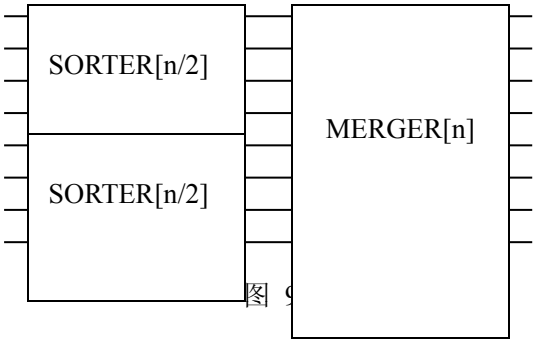


图 9

SORTER[n]的构造方法即如图 9 所示。

我们以一种递归的形式构造出了排序网络 ***SORTER[n]***，递归的边界是 **n=2**。

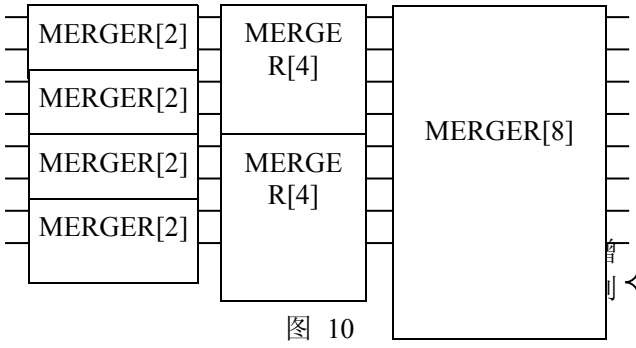


图 10

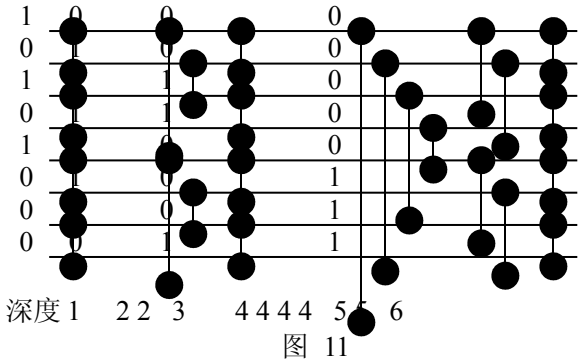


图 11

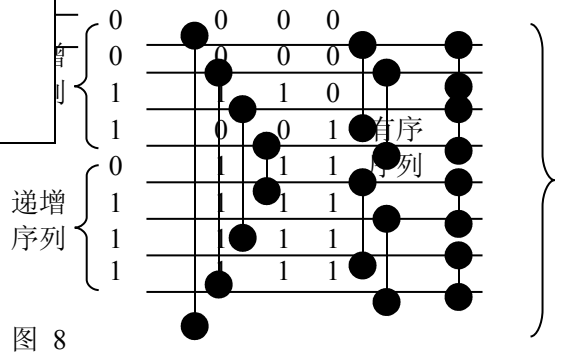


图 8

图 10 和图 11 详细地给出了 n=8 的排序网络。

在排序网络 ***SORTER[n]***中，数据要通过 $\log_2 n$ 个阶段。其中第 K 个($1 \le K \le \log_2 n$)阶段包含 $n/2^K$ 个 ***MERGER[2^K]***，它们并行地把每对由 2^{K-1} 个元素组成的递增序列进行合并以产生长度为 2^K 的递增序列。它的深度为 K。所以，***SORTER[n]*** 总的深度为 $O(\lg^2 n)$，我们能够在 $O(\lg^2 n)$ 的时间内并行地对 n 个数进行排序。

在解决排序网络这个问题的过程中，我们多次把问题化简、转化。我们先是利用 ***0-1 原则***大大地缩小了输入序列的范围。接着，我们又采用二分法的思想，把原序列分成两个规模缩小一半的序列，先分别对它们排序，再把它们合并成一个递增序列。于是，问题就转化为构造一类***合并网络***。然后，我们发现***半清洁器***可以把一个***双调序列***变为两个规模缩小一半的双调序列且使得上半部分的输出值都不大于下半部分的输出值。这是第二次使用了二分法的思想。通过递归地调用半清洁器，***双调排序网络***应运而生，合并网络得以解决。最后，我们通过递归地调用合并网络，构造出了一类***排序网络***。

在解题的过程中，**缩小范围**、**二分法**、**递归调用已有结果**等思想给予了我们很大的帮助。巧妙地使用这些技巧往往可以事半功倍。