A 12-STEP SORTING NETWORK FOR 22 ELEMENTS

SHERENAZ W. AL-HAJ BADDAR
Department of Computer Science, Kent State University
Kent, Ohio 44240, USA

KENNETH E. BATCHER
Department of Computer Science, Kent State University
Kent, Ohio 44240, USA

Abstract— Sorting Networks are cost-effective multistage interconnection networks with sorting capabilities. Optimal sorting networks theoretically consume Θ(NlogN) comparisons. However, the fastest implementable sorting networks built so far consume $\Theta(N\log^2 N)$ comparisons, and generally, use the Merge-sorting strategy to sort the input. This implies that faster networks can be developed — here we show a network that sorts 22 elements in only 12 steps, outperforming the Merge-sorting based solution for this problem which needs at least 13 steps.

Keywords: Sorting Networks, Zero/One Principle, Partial Ordering, Parallel Sorting Algorithms.

1. Introduction

Parallel processors are fast and powerful computing systems that have been developed to help undertake computationally challenging problems. Deploying parallelism for solving a given problem implies splitting the problem into subtasks that get assigned to the computing components constituting a parallel system. These components usually communicate in order to accomplish their designated subtasks, which introduces the problem of connecting them efficiently. Several interconnection networks schemes have been built to help solve this problem, among which are multistage interconnection networks. These widely used networks deploy a significantly smaller number of switching elements to achieve a relatively more efficient inter-processor communication. Many multistage interconnection networks were developed including sorting networks.

Van Voorhis[1] defines a sorting network as a circuit with N inputs and N outputs such that for any set of inputs $\{I_1,I_2,\ldots,I_N\}$, the resulting output is the set $\{O_1,O_2,\ldots,O_N\}$. The output set must be a permutation of the input set $\{I_1,I_2,\ldots,I_N\}$. Moreover, for every two elements of the output set O_j and O_k, O_j must be less than or equal to O_k whenever $j \leq k$.

Sorting networks are constructed using stages (steps) of basic cells called Comparator Exchange(CE) modules. A CE is a two-element sorting circuit. It accepts two inputs via two input lines, compares them and outputs the larger element on its high output line, whereas the smaller is output on its low output line. It is assumed that two comparators with disjoint inputs can operate in parallel. A typical CE is depicted in Figure 1.

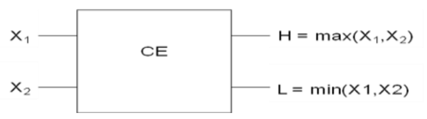


Fig. 1. A comparator exchange module[2].

An optimal N-element sorting network requires θ(NlogN) comparisons to sort N elements[3]. However, no practical solution for building such networks has been developed so far. On the other hand, several practical θ(Nlog^{2}N) techniques for building sorting networks exist among which is the Merge-sorting technique[2]. This technique has been generally deployed for building the best performing sorting networks known so far. Accordingly, the best known network for sorting 22 elements using the Merge-sorting technique consumes 13 steps. Nevertheless, a 12-step network for the same input size is illustrated here. This network shows that developing implementable networks that are closer to the optimal complexity of θ(NlogN) is possible.

Section 2 reviews some basic concepts necessary for discussing the network illustrated here. Section 3 depicts the 12 steps constituting the developed network using Sortnet, a software tool developed by Batcher to help build better sorting networks[4]. Finally, Section 4 concludes this report and highlights future work.

2. Background

This section illustrates some basic concepts necessary for discussing the developed sorting network. These concepts include: Knuth diagrams, Zero/One principle, partial ordering, Haase diagrams, and Sortnet.

2.1. *Knuth diagrams*

Knuth diagrams are pictorial representations of sorting networks that help distinguish the several steps constituting the investigated network[5]. In a typical Knuth diagram, each input element is represented by a horizontal line and each CE is represented by a vertical line connecting the two elements being compared. The elements being sorted are assumed to have numeric labels running from zero through (N-1). Moreover, the network's top most element is assigned the highest label and the bottom most element is assigned the smallest one. It is also assumed that the elements are fed into the left most end of the network and received sorted at the other end, with the maximum element being the top most element. Figure 2 illustrates a typical Knuth diagram for sorting four elements.

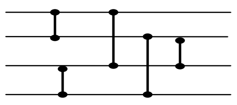


Fig. 2. A Knuth diagram for a four-element sorting network[5].

2.2. *The Zero/One principle*

The Zero/One principle plays a vital role in building and verifying sorting networks. It states that a given N-element sorting network sorts N inputs correctly if it sorts all the 2^N binary strings of length N[5].

A zero/one case is a sequence of length N where the value of each entry in the sequence is either zero or one. If a given zero/one case, A, of length N has j zeros then it has N-j ones. When a comparator element compares two entries in A, it might swap the values of the compared entries. However, the number of zero and one entries in A remains the same. Hence, a series of comparator elements sorts case A iff it rearranges A’s entries such that the first j locations hold zeros and the next N-j locations hold ones. The number of zeros(ones) in a given binary sequence of N bits ranges between zero and N which implies that there are N+1 zero/one sorted cases. Hence, an N-element sorting network, that rearranges any of the 2^N input permutations into one of the N+1 possible sorted zero/one cases, can sort any arbitrary sequence of N elements.

2.3. *Partial ordering and Haase diagrams*

A total order relation is imposed on the set of elements sorted by a typical sorting network. This total ordering is achieved by the end of the sorting process. However, only a partial order relation exists on this set at any arbitrary step in the sorting process prior to the last step, and such a set is called a **p**artial **o**rder **set** (poset)[6]. Having a partial ordering relation, denoted by R, on a set of elements implies that there may exist a pair of elements x and y, such that there is at least one zero/one case where x = zero and y = one, and at least another zero/one case, where x = one and y = zero.

Haase diagrams are used in order to visualize the progress of a sorting network[6]. In a conventional Haase diagram, elements are represented by vertices and relations among them are represented by edges. An edge running from vertex x to vertex y exists iff there does not exist a zero/one case in which x = zero and y = one. It is noticed that the relative positioning of an edge’s two endpoints implies its direction. More precisely, if vertex x appears above vertex y, then the direction of the edge running between both is assumed to be from x to y.

Figure 3 depicts the progress of the sorting network illustrated in Fig. 2 using Haase diagrams. In the first step, two partial ordering relations are formed and such diagram is called a multi-segment poset. In the second step, the two segments are combined into a one-segment poset, or simply a poset. Finally, step three transforms this poset into a totally ordered chain of elements.

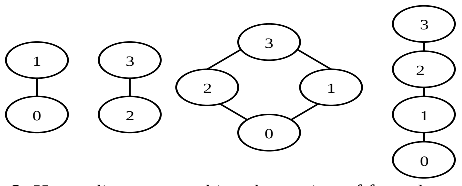


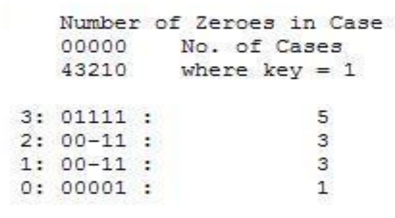
Fig. 3. Haase diagrams tracking the sorting of four elements.

2.5. Using Sortnet

Sortnet is a program developed by K. E. Batcher to help synthesize and analyze sorting networks[4]. A subset of Sortnet commands, that will help in analyzing the sorting network described here, is illustrated in this subsection.

2.5.1 *Shmoo charts*

A Shmoo chart is a two-dimensional diagram where each column shows all zero/one cases with the same number of zeros and each row shows an element[4]. Figure 4 illustrates the Shmoo chart generated after step two of the four-element network depicted in Fig. 2.



```
   Number of Zeroes in Case
   00000    No. of Cases
   43210    where key = 1

3: 01111 :            5
2: 00-11 :            3
1: 00-11 :            3
0: 00001 :            1
```

Fig. 4. The Shmoo chart of the four-element network after step two.

The columns of a typical Shmoo chart are ordered according to the number of zeros in the set of zero/one cases with the case of all zeros at the left of the chart and the one with all ones at the right of it. A typical dash in a Shmoo chart that is located in row r and column c implies that element r is equal to zero in at least one zero/one case with c zeros and is also equal to one in at least one different zero/one case with the same number of zeros[4]. A Shmoo chart becomes dash-free when all elements get sorted, i.e. all entries in the Shmoo chart are either zeroes or ones.

2.5.2 *The SHOW.BESTCE command*

This command displays the CEs that affect the most zero/one cases and remove the most dashes along with the number of the earliest step in which they can be deployed[4]. The CEs suggested by the BESTCE command are sorted according to the number of treated cases, and then the number of dashes removed in non-increasing order. It is then up to the network designer to pick the CEs she/he thinks are the best and add them to the CE-list.

3. A 12-Step Solution for Sorting 22 Elements

This section describes a 12-step solution for sorting 22 elements that outperforms the merge-based, 13-step network which used to be the best known solution for this problem. Figure 5 illustrates the developed network.

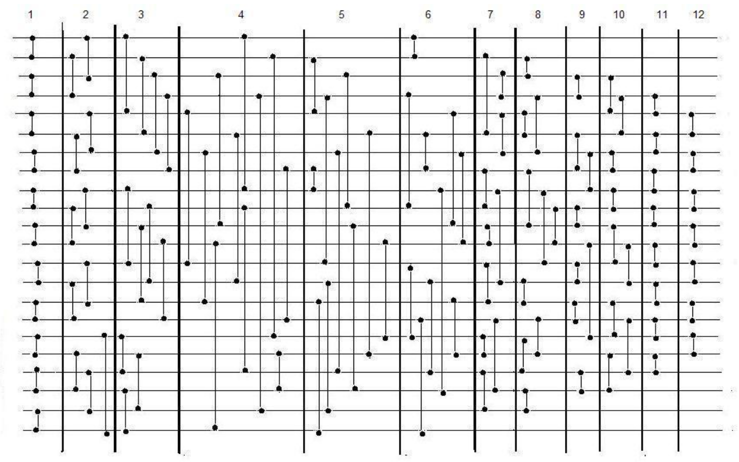


Fig. 5. The Knuth diagram of the 12-step network for sorting 22 elements.

Figure 6 depicts the comparison elements that compose the network illustarated in Figure 5.

/* STEP 1 */ 0:1 2:3 4:5 6:7 8:9 10:11 12:13 14 :15 16:17 18:19 20: 21 /* STEP 2 */ 2 :4 1 :3 0:5 6:8 7:9 10 :12 11: 13 14: 16 15: 17 18 :20 19: 21 /* STEP 3 */ 6 :10 7: 11 8: 12 9 :13 14 :18 15: 19 16: 20 17: 21 3 : 5 1 : 4 0 : 2 /* STEP 4 */ 9 :17 7 :15 11: 19 8: 16 3 :12 0: 10 1: 18 5 :20 13: 21 6 :14 2: 4	/* STEP 5 */ 0 : 7 17 :20 3: 15 9: 18 2: 11 4: 16 5: 10 1 : 8 12: 19 13 :14 /* STEP 6 */ 20: 21 0 : 6 3 : 8 12: 18 2: 13 14 : 16 5 : 9 10 :15 4 :7 11: 17 /* STEP 7 */ 16: 20 18: 19 15: 17 12 :14 10: 11 7: 9 8 :13 4: 5 1 : 3 2 : 6 /* STEP 8 */ 19: 20 16: 17 15: 18 11: 14 9: 13 10: 12 7 : 8 3 : 5 4: 6 1 : 2	/* STEP 9 */ 18: 19 14 :16 13: 15 11: 12 8 : 9 5: 10 6 : 7 2 : 3 /* STEP 10 */ 17: 19 16: 18 14: 15 12: 13 9 :11 8: 10 5 : 7 3 : 6 2 : 4 /* STEP 11 */ 17: 18 15: 16 13: 14 11 :12 9 :10 7 : 8 5 : 6 3 : 4 /* STEP 12 */ 16: 17 14 :15 12 :13 10: 11 8 : 9 6 : 7 4 : 5

Fig. 6 The CE-list of the 22-element network

3.1. The first three steps

Several ways exist for creating a 22-element one-segment poset. To mention few, consider building a 2-segment poset that consists of a 10-element segment together with a 12-element segment. Alternatively, consider a 2-segment poset whose two segments consist of 11 elements. The later poset can be formed by building two posets each of which contains 12 elements, and then throwing one element away form each poset. Nevertheless, experimentation showed that building a 22-element one-segment poset out of one 6-element segment together with two 8-element segments helped in developing a faster solution for this problem. These three segments are illustrated in Figure 7 and the set of CEs that creates them is depicted in Figure 6.

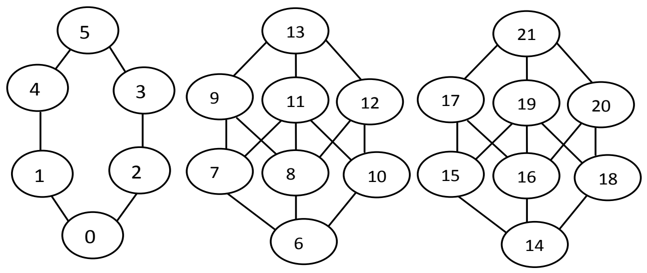
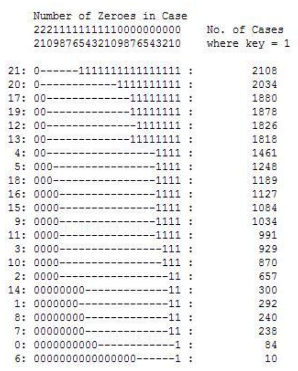


Fig. 7. The three-segment poset obtained after applying the first three steps.

3.2. The fourth and fifth steps

The CEs in step four will transform the three-segment poset generated from the first three steps into a one-segment poset. Obviously, many ways exist to pair the 22 elements being sorted so that a one-segment poset is formed. However, one criterion that is crucial for building a faster sorting network at this stage is to preserve as much order as possible. Thus, consider the CEs from step 4 and notice the pairs (9:17), (7:15), (11:19), (8:16), (13:21), and (6:14). These pairs compare corresponding elements in the two 8-element segments obtained after the first three steps, and obviously, preserve order. Now, consider the pair (3:12), these two elements are similar in the sense that each one of them is covered by one element and covers 2 elements in their corresponding segments. A similar analysis applies to the pair (1:18). This justifies including these two pairs of comparison in step 4. Re-examining Figure 6 shows that the elements that need yet to be compared are (5, 4, 2, 0) from the 6-element segment, 10 from the first 8-element segment, and 20 from the second 8-element segment. Experimentation showed that the following pairing of these elements resulted in finding a faster solution: (5:20), (0,10), and (2,4). The Shmoo chart that results from applying these 11 CEs is illustrated in Figure 8(a).

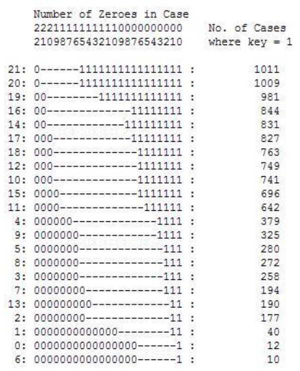
Due to the fact that posets become messy at such a stage, a different strategy needs to be deployed in order to help pick better CEs. Thus, the SHOW.BESTCE command, provided by Sortnet, was utilized to help compose step 5 of this network. The SHOW.BESTCE command applied at this stage returns multiple rows of CEs, experimentation showed that selecting the CEs that belong to the lower rows not only removed more dashes from the Shmoo chart, but also helped in eliminating more cases. These CEs are depicted in Figure 6 and the corresponding Shmoo chart is illustrated in Figure 8(b).



```
     Number of Zeroes in Case
     22211111111110000000000   No. of Cases
     21098765432109876543210   where key = 1

21:  0------1111111111111111 :        2108
20:  0------------1111111111 :        2034
17:  00-------------11111111 :        1880
19:  00-------------11111111 :        1878
12:  00--------------1111111 :        1826
13:  00-------------11111111 :        1818
 4:  00-----------------1111 :        1461
 5:  000----------------1111 :        1248
18:  000----------------1111 :        1189
16:  0000---------------1111 :        1127
15:  0000---------------1111 :        1084
 9:  0000---------------1111 :        1034
11:  0000---------------1111 :         991
 3:  0000----------------111 :         929
10:  0000----------------111 :         870
 2:  0000-----------------11 :         657
14:  00000000-------------11 :         300
 1:  0000000--------------11 :         292
 8:  00000000-------------11 :         240
 7:  00000000-------------11 :         238
 0:  0000000000------------1 :          84
 6:  0000000000000000------1 :          10
```

Fig. 8(a). The Shmoo chart after step four.



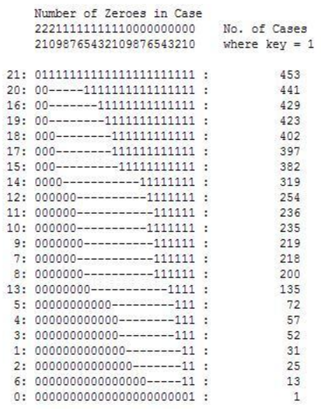
```
     Number of Zeroes in Case
     22211111111110000000000   No. of Cases
     21098765432109876543210   where key = 1

21:  0------1111111111111111 :        1011
20:  0------1111111111111111 :        1009
19:  00---------111111111111 :         981
16:  00-------------11111111 :         844
14:  00------------111111111 :         831
17:  000------------11111111 :         827
18:  000-------------1111111 :         763
12:  000-------------1111111 :         749
10:  000-------------1111111 :         741
15:  0000------------1111111 :         696
11:  0000-------------111111 :         642
 4:  000000-------------1111 :         379
 9:  0000000------------1111 :         325
 5:  0000000-------------111 :         280
 8:  0000000-------------111 :         272
 3:  0000000-------------111 :         258
 7:  00000000------------111 :         194
13:  000000000------------11 :         190
 2:  00000000-------------11 :         177
 1:  0000000000000--------11 :          40
 0:  0000000000000000------1 :          12
 6:  0000000000000000------1 :          10
```

Fig. 8(b). The Shmoo chart after step five.

3.4. The sixth and seventh steps

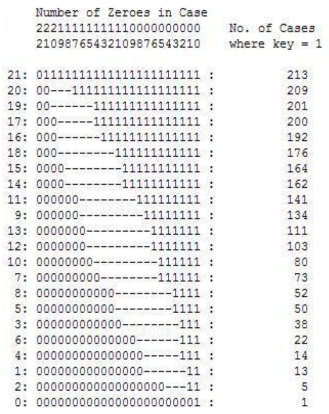
The inspection of the Shmoo chart in Figure8(b) suggests deploying (0:6) and (20:21) in the CE list of step 6. The rest of CEs that constitute this step were selected via using the SHOW.BESTCE, with the same previous criterion for CE selection. Figure9(a) illustrates the Shmoo chart obtained from applying the CEs in step 6 as illustrated in Figure 6.
Experimentation showed that using the Shmoo chart that resulted from step 6 to figure out the CEs that compose step 7 helped in finding a faster network for sorting 22 elements. Hence, each two adjacent elements that appear in Figure 9(a) were compared resulting in the Shmoo chart depicted in Figure 9(b).



```
     Number of Zeroes in Case
     22211111111110000000000   No. of Cases
     21098765432109876543210   where key = 1

21:  01111111111111111111111 :         453
20:  00-----1111111111111111 :         441
16:  00-------11111111111111 :         429
19:  00--------1111111111111 :         423
18:  000--------111111111111 :         402
17:  000--------111111111111 :         397
15:  000---------11111111111 :         382
14:  0000-----------11111111 :         319
12:  000000----------1111111 :         254
11:  000000-----------111111 :         236
10:  000000----------1111111 :         235
 9:  0000000----------111111 :         219
 7:  000000-----------111111 :         218
 8:  0000000----------111111 :         200
13:  00000000-----------1111 :         135
 5:  00000000000---------111 :          72
 4:  000000000000--------111 :          57
 3:  000000000000--------111 :          52
 1:  0000000000000--------11 :          31
 2:  00000000000000-------11 :          25
 6:  0000000000000000-----11 :          13
 0:  00000000000000000000001 :           1
```

Fig. 9(a). The Shmoo chart after step six.



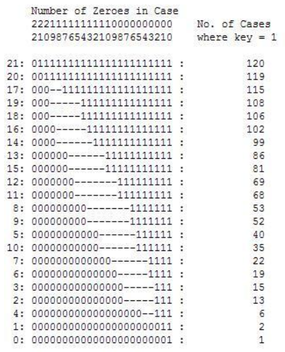
```
     Number of Zeroes in Case
     22211111111110000000000   No. of Cases
     21098765432109876543210   where key = 1

21:  01111111111111111111111 :         213
20:  00---111111111111111111 :         209
19:  00------111111111111111 :         201
17:  000-----111111111111111 :         200
16:  000------11111111111111 :         192
18:  000--------111111111111 :         176
15:  0000--------11111111111 :         164
14:  0000--------11111111111 :         162
11:  000000--------111111111 :         141
 9:  000000---------11111111 :         134
13:  0000000---------1111111 :         111
12:  0000000---------1111111 :         103
10:  00000000---------111111 :          80
 7:  000000000--------111111 :          73
 8:  00000000000--------1111 :          52
 5:  00000000000--------1111 :          50
 3:  000000000000--------111 :          38
 6:  00000000000000------111 :          22
 4:  000000000000000-----111 :          14
 1:  000000000000000------11 :          13
 2:  000000000000000000---11 :           5
 0:  00000000000000000000001 :           1
```

Fig. 9(b). The Shmoo chart after step seven.

3.5. The eighth and ninth steps

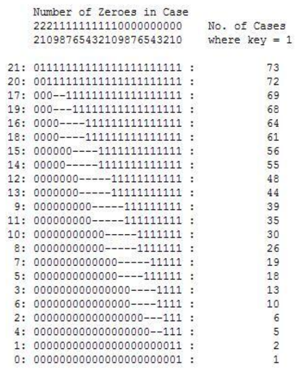
Experimentation showed again that using the Shmoo chart for building step 8 helps in building a faster network. Thus, the Shmoo chart obtained from step 7 and depicted in Figure 9(b) was used to compose step 8 as illustrated in Figure 6, and the resulting Shmoo chart is depicted in Figure 10(a).
Inspecting the Shmoo chart obtained from step 8 shows a grouping of elements, i. e. pairs of adjacent elements that have an almost equal number of dashes can be identified. These pairs compose the CE-list of step 9 as shown in Figure 6 and the resulting Shmoo chart is illustrated in Figure 10(b).



```
    Number of Zeroes in Case
    22211111111110000000000   No. of Cases
    21098765432109876543210   where key = 1

21: 01111111111111111111111 :          120
20: 00111111111111111111111 :          119
17: 000--111111111111111111 :          115
19: 000-----111111111111111 :          108
18: 000-----111111111111111 :          106
16: 0000-----11111111111111 :          102
14: 0000------1111111111111 :           99
13: 000000------11111111111 :           86
15: 000000------11111111111 :           81
12: 0000000-------111111111 :           69
11: 0000000-------111111111 :           68
 8: 000000000-------1111111 :           53
 9: 000000000-------1111111 :           52
 5: 00000000000------111111 :           40
10: 00000000000------111111 :           35
 7: 0000000000000------1111 :           22
 6: 00000000000000-----1111 :           19
 3: 000000000000000-----111 :           15
 2: 000000000000000-----111 :           13
 4: 000000000000000000--111 :            6
 1: 00000000000000000000011 :            2
 0: 00000000000000000000001 :            1
```

Fig. 10(a). The Shmoo chart after step eight.



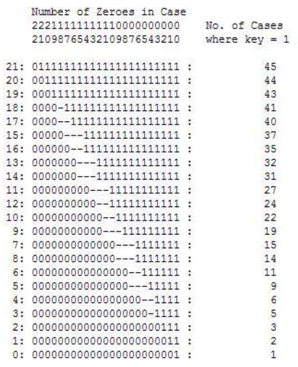
```
    Number of Zeroes in Case
    22211111111110000000000   No. of Cases
    21098765432109876543210   where key = 1

21: 01111111111111111111111 :           73
20: 00111111111111111111111 :           72
17: 000--111111111111111111 :           69
19: 000---11111111111111111 :           68
16: 0000----111111111111111 :           64
18: 0000----111111111111111 :           61
15: 000000----1111111111111 :           56
14: 00000-----1111111111111 :           55
12: 0000000-----11111111111 :           48
13: 0000000-----11111111111 :           44
 9: 000000000-----111111111 :           39
11: 000000000-----111111111 :           35
10: 00000000000-----1111111 :           30
 8: 00000000000-----1111111 :           26
 7: 0000000000000-----11111 :           19
 5: 0000000000000----111111 :           18
 3: 000000000000000----1111 :           13
 6: 000000000000000----1111 :           10
 2: 00000000000000000---111 :            6
 4: 000000000000000000--111 :            5
 1: 00000000000000000000011 :            2
 0: 00000000000000000000001 :            1
```

Fig. 10(b). The Shmoo chart after step nine.

3.6. The tenth and eleventh steps

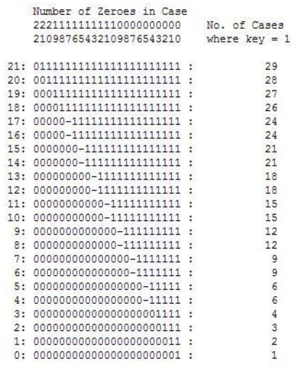
It is obvious that the Shmoo chart illustrated in Figure 10(b) shows grouping patterns of the elements too. Thus, each two elements within a group are compared composing step 10 as illustrated in Figure 6 and resulting in the Shmoo chart depicted in Figure 11(a). The Shmoo chart obtained from step 10 maintains the grouping pattern with some distortions within some groups. It also shows that the number of dashes within groups in the middle is smaller than the number of dashes within groups located at the two ends of the Shmoo chart. Again, each two adjacent elements are compared and the resulting Shmoo chart is illustrated in Figure 11(b).



```
    Number of Zeroes in Case
    22211111111110000000000   No. of Cases
    21098765432109876543210   where key = 1

21: 01111111111111111111111 :           45
20: 00111111111111111111111 :           44
19: 00011111111111111111111 :           43
18: 0000-111111111111111111 :           41
17: 0000--11111111111111111 :           40
15: 00000---111111111111111 :           37
16: 000000--111111111111111 :           35
13: 0000000---1111111111111 :           32
14: 0000000---1111111111111 :           31
11: 000000000---11111111111 :           27
12: 0000000000--11111111111 :           24
10: 00000000000--1111111111 :           22
 9: 00000000000---111111111 :           19
 7: 0000000000000---1111111 :           15
 8: 0000000000000---1111111 :           14
 6: 000000000000000--111111 :           11
 5: 000000000000000---11111 :            9
 4: 00000000000000000--1111 :            6
 3: 000000000000000000-1111 :            5
 2: 00000000000000000000111 :            3
 1: 00000000000000000000011 :            2
 0: 00000000000000000000001 :            1
```

Fig. 11(a). The Shmoo chart after step ten.



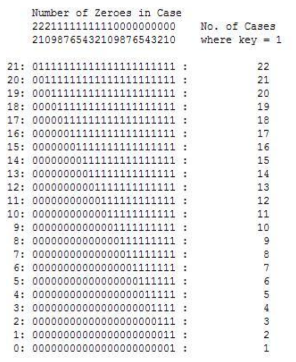
```
    Number of Zeroes in Case
    22211111111110000000000   No. of Cases
    21098765432109876543210   where key = 1

21: 01111111111111111111111 :           29
20: 00111111111111111111111 :           28
19: 00011111111111111111111 :           27
18: 00001111111111111111111 :           26
17: 00000-11111111111111111 :           24
16: 00000-11111111111111111 :           24
15: 0000000-111111111111111 :           21
14: 0000000-111111111111111 :           21
13: 000000000-1111111111111 :           18
12: 000000000-1111111111111 :           18
11: 00000000000-11111111111 :           15
10: 00000000000-11111111111 :           15
 9: 0000000000000-111111111 :           12
 8: 0000000000000-111111111 :           12
 7: 000000000000000-1111111 :            9
 6: 000000000000000-1111111 :            9
 5: 00000000000000000-11111 :            6
 4: 00000000000000000-11111 :            6
 3: 00000000000000000001111 :            4
 2: 00000000000000000000111 :            3
 1: 00000000000000000000011 :            2
 0: 00000000000000000000001 :            1
```

Fig. 11(b). The Shmoo chart after step 11.

3.7. The twelfth steps

It is quite apparent that the Shmoo chart illustrated in Figure 11(b) shows a perfect grouping pattern of the elements, with two dashes per group. Hence, the seven remaining CEs are applied to step 12 and the resulting Shmoo chart shows that the 22 elements have become perfectly sorted after twelve steps as depicted in Figure 12.



```
    Number of Zeroes in Case
    22211111111110000000000    No. of Cases
    21098765432109876543210    where key = 1

21: 01111111111111111111111 :            22
20: 00111111111111111111111 :            21
19: 00011111111111111111111 :            20
18: 00001111111111111111111 :            19
17: 00000111111111111111111 :            18
16: 00000011111111111111111 :            17
15: 00000001111111111111111 :            16
14: 00000000111111111111111 :            15
13: 00000000011111111111111 :            14
12: 00000000001111111111111 :            13
11: 00000000000111111111111 :            12
10: 00000000000011111111111 :            11
 9: 00000000000001111111111 :            10
 8: 00000000000000111111111 :             9
 7: 00000000000000011111111 :             8
 6: 00000000000000001111111 :             7
 5: 00000000000000000111111 :             6
 4: 00000000000000000011111 :             5
 3: 00000000000000000001111 :             4
 2: 00000000000000000000111 :             3
 1: 00000000000000000000011 :             2
 0: 00000000000000000000001 :             1
```

Fig. 12. The Shmoo chart after step 12.

4. Conclusion and Future Work

Sorting networks are cost-effective multistage interconnection networks with sorting capabilities. The fastest practical sorting networks built so far generally deploy the Merge-sorting technique developed by Batcher and consume $\theta(N\log^2 N)$ comparisons. Nevertheless, optimal sorting networks theoretically consume $\theta(N\log N)$ comparisons. Consequently, sorting networks that outperform the fastest practical sorting networks can be built. Here we describe a 22-element sorting network that is one step less than the fastest known network for the same input size. This network was designed using Sortnet, a software package developed by Batcher to help build better sorting networks.

The 22-element sorting network illustarted here enables improving the best known networks with input sizes that are multiples of 22. Nevertheless, such a network requires further analysis so that it can be generalized for larger input sizes that are not multiples of 22.

References

1. Van Voorhis, D. C., "Efficient Sorting Networks", Stanford University, Ph.D. thesis(Stanford University, 1972).
2. K. E. Batcher, Sorting Networks and their Applications, Spring *Joint Computer Conference, AFIPS Proc.* 32 (1968) 307-314.
3. M. Ajtai, J. Komlos, and E. Szemerdi, Sorting in nlogn Steps, *Combinatorica* 3 (1983) 1-19.

4. Batcher K. E. and S. Al-Haj Baddar, "Sortnet: A Program for Building Sorting Networks", Technical Report No. TR-KSU-CS-2008-01, Department of Computer Science, Kent State University, February 2008.
5. Knuth, D. E., *The Art of Computer Programming: Volume 3: Sorting and Searching* (Addison-Wesley Longman, USA, 2nd Edition, 1998, Chapter 5, pp. 225-228).
6. Rosen, K. H., *Discrete Mathematics and its Applications* (McGraw-Hill Companies, USA, 5th Edition, 2003) 520-525.