Matching Nuts and Bolts Faster*

Noga Alon[1] Phillip G. Bradford[2] Rudolf Fleischer[2]

[1] Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. E-mail: noga@math.tau.ac.il.

[2] Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: {bradford,rudolf}@mpi-sb.mpg.de.

Abstract. The problem of matching nuts and bolts is the following : Given a collection of n nuts of distinct sizes and n bolts such that there is a one-to-one correspondence between the nuts and the bolts, find for each nut its corresponding bolt. We can *only* compare nuts to bolts. That is we can neither compare nuts to nuts, nor bolts to bolts. This humble restriction on the comparisons appears to make this problem very hard to solve. In fact, the best explicit deterministic algorithm to date is due to Alon *et al.* [2] and takes $\Theta(n \log^4 n)$ time. In this paper, we give a simpler $O(n \log^2 n)$ time algorithm. The existence of an $O(n \log n)$ time algorithm has been proved recently [6, 9].

1 Introduction

In [14], page 293, Rawlins posed the following interesting problem :

> *We wish to sort a bag of n nuts and n bolts by size in the dark. We can compare the sizes of a nut and a bolt by attempting to screw one into the other. This operation tells us that either the nut is bigger than the bolt; the bolt is bigger than the nut; or they are the same size (and so fit together). Because it is dark we are not allowed to compare nuts directly or bolts directly.*
>
> *How many fitting operations do we need to sort the nuts and bolts in the worst case?*

As a mathematician (instead of a carpenter) you would probably prefer to see the problem stated as follows ([2]) :

---

* A preliminary version of the paper was presented at ISAAC'95 [5].

* The first author was supported in part by a USA Israeli BSF grant. The second and the third author were partially supported by the EU ESPRIT LTR Project No. 20244 (ALCOM-IT).

Given two sets $B = \{b_1, \ldots, b_n\}$ *and* $S = \{s_1, \ldots, s_n\}$, *where* B *is a set of* n *distinct real numbers (representing the sizes of the bolts) and* S *is a permutation of* B, *we wish to find efficiently the unique permutation* $\sigma \in S_n$ *so that* $b_i = s_{\sigma(i)}$ *for all* i, *based on queries of the form* compare b_i and s_j. *The answer to each such query is either* $b_i > s_j$ *or* $b_i = s_j$ *or* $b_i < s_j$.

The obvious information theoretic lower bound shows that at least $\Omega(n \log n)$ comparisons are needed to solve the problem, even for a randomized algorithm. In fact, there is a simple randomized algorithm which achieves an expected running time of $O(n \log n)$, namely Quicksort : Pick a random nut, find its matching bolt, and then split the problem into two subproblems which can be solved recursively, one consisting of the nuts and bolts smaller than the matched pair and one consisting of the larger ones. The standard analysis of randomized Quicksort gives the expected running time as stated above (see for example [7]).

Unfortunately, it is much harder to find an efficient deterministic algorithm. The first $o(n^2)$ time algorithm, also based on Quicksort, was given by Alon *et al.* [2]. To find a good pivot element which splits the problem into two subproblems of nearly the same size, they run $\log n$ iterations of a procedure which eliminates half of the nuts in each iteration while maintaining at least one good pivot; since there is only one nut left in the end, this one must be a good pivot. This procedure uses the edges of a highly efficient expander of degree $\Theta(\log^2 n)$ to define its comparisons. Therefore, finding a good pivot takes $\Theta(n \log^3 n)$ time, and the entire Quicksort takes $\Theta(n \log^4 n)$ time.

Recently, Bradford [6] and Komlós, Ma and Szemerédy [9] discovered independently how to solve the nuts and bolts problem in $O(n \log n)$ time. The former paper does an $O(n)$ selection of the nuts and bolts, whereas the latter one is based on a modification of the famous AKS sorting network [1]. Unfortunately, both papers only prove the existence of an optimal algorithm by probabilistic arguments with no hint how it could actually be constructed. Komlós *et al.* [9] also mention (without giving any details) that they have found a fairly simple $O(n(\log \log n)^2)$ time algorithm for selecting a good pivot, and hence an $O(n \log n (\log \log n)^2)$ time algorithm for sorting nuts and bolts.

In this paper, we propose a simple and fast algorithm for finding a good pivot. First, we connect the set of nuts with the set of bolts via some expander graph of constant degree. We then choose greedily a maximal set of nuts which are connected to a smaller bolt and a larger bolt. On these nuts we play a simple knockout tournament where in each round half of the nuts are eliminated, guaranteeing that the winner of the tournament is a good pivot. Since we can play each round of the tournament in $O(n)$ time, we can find a good pivot in $O(n \log n)$ time. Therefore, we can solve the nuts and bolts matching problem in $O(n \log^2 n)$ time.

Alon *et al.* [2] mention two potential applications of this problem: the first is local sorting of nodes in a given graph [8], and the second is selection of read only memory with a little read/write memory [13].

In the next section, we describe the Quicksort algorithm more formally and recall some facts about expanders. In Section 3, we show how we can efficiently find a good pivot. We conclude with some remarks in Section 4.

2 Basic Definitions

Let $S = \{s_1, \ldots, s_n\}$ be a set of nuts of different sizes and $B = \{b_1, \ldots, b_n\}$ be a set of corresponding bolts. For a nut $s \in S$ define $rank(s)$ as $|\{t \in B \mid s \geq t\}|$. The rank of a bolt is defined similarly. For a constant $c < \frac{1}{2}$, s is called a *c-approximate median* if $cn \leq rank(s) \leq (1-c)n$. Similarly, define the *relative rank* of s with respect to a subset $T \subseteq B$ as $rank_T(s) := \dfrac{|\{t \in T \mid s \geq t\}|}{|T|}$.

The algorithm for matching nuts and bolts works as follows.

(1) Find a c-approximate median s of the n given nuts (we will determine c later).

(2) Find the bolt b corresponding to s.

(3) Compare all nuts to b and all bolts to s. This gives two piles of nuts (and bolts as well), one with the nuts (bolts) smaller than s and one with the nuts (bolts) bigger than s.

(4) Run the algorithm recursively on the two piles of the smaller nuts and bolts and the two piles of the bigger nuts and bolts.

In the next section, we show how we can find a c-approximate median in $O(n \log n)$ time, where c is a small constant. Then our main result follows immediately.

Theorem 1. *We can match n nuts with their corresponding bolts in $O(n \log^2 n)$ time.*

Proof. The correctness of the algorithm above follows immediately from the correctness of Quicksort. For the running time observe that each subproblem has size at most $(1-c)n$, hence the depth of the recursion is only $O(\log n)$, and in each level of the recursion we spend at most $O(n \log n)$ time to compute the c-approximate median and $O(n)$ time to split the problem into the two subproblems. □

We now recall some facts about expanders (see for example [10] if you want to learn more about expanders). An undirected graph G is called an *(n, d, λ)-graph* if it is a d-regular graph on n nodes, and the absolute value of each of the eigenvalues of its adjacency matrix, besides the largest, is at most λ.

Call a sequence of integers *dense* if for every $\epsilon > 0$ there exists some $m_0 = m_0(\epsilon)$ so that for every $m > m_0$ the sequence contains a member between m and $(1+\epsilon)m$.

Proposition 2 ([11, Th. 2.3],[12]). *Let p be a prime congruent to 1 modulo 4 and $d = p + 1$. Then there is a dense sequence of integers such that we can explicitly construct in time $O(d \cdot n)$ an $(n, d, 2\sqrt{d-1})$-graph for every member n of the sequence.* □

Such graphs are called *Ramanujan graphs* in [11]. The construction of the graphs is quite simple, just the proofs are involved. For more details on such graphs see for example [4].

Proposition 3 ([4, Cor. 2.5, page 122]). *Let $G = (V, E)$ be an (n, d, λ)-graph. Then for every two sets of vertices B and C of G, where $|B| = bn$ and $|C| = cn$, we have $|(\#\text{edges between } B \text{ and } C) - cbdn| \le \lambda \cdot \sqrt{bc} \cdot n$.* □

Theorem 4. *We can construct in $O(n)$ time a bipartite graph $G = (X \cup Y, E)$, $|X| = |Y| = n$, with the property that any subset of X of size $\frac{n}{6}$ is connected to at least $\frac{7}{8}n$ nodes in Y.*

Proof. Let $p = 197$ (a prime congruent to 1 modulo 4), $d = p + 1 = 198$ and $\lambda = 2\sqrt{d-1} = 2\sqrt{197}$. Let $\epsilon = \frac{1}{100}$ and n' be a member of the dense sequence of integers mentioned in Prop. 2 such that $n \le n' \le (1+\epsilon)n$. Let $G' = (V, E')$ be the (n', d, λ)-graph constructed in $O(n)$ time in Prop. 2. Let X and Y be arbitrary subsets of V of size n each. Define $G = (X \cup Y, E)$ by joining $x \in X$ with $y \in Y$ iff $x = y$ or $(x, y) \in E'$.

Let B be any subset of X of size $\frac{n}{6}$ and C any subset of Y of size $\frac{n}{8}$. Then $|B| = \frac{n}{6} \ge \frac{100}{606} \cdot n'$ and $|C| = \frac{n}{8} \ge \frac{100}{808} \cdot n'$ and hence by Prop. 3 ($\#$ edges between B and C) $\ge \lceil \frac{100}{606} \cdot \frac{100}{808} \cdot 198 \cdot n - 2 \cdot \sqrt{197} \cdot \sqrt{\frac{100}{606} \cdot \frac{100}{808}} \cdot n \rceil \ge \lceil 4.043n - 4.012n \rceil > 0$. Hence G has the desired property. □

3 Finding a c-Approximate Median

Our algorithm to find a c-approximate median is based on a knockout tournament played on some subset of the nuts. We start with a subset $S_1 \subset S$ of the nuts where each nut $s \in S_1$ has a set $T_1(s)$ of two bolts associated with it, one smaller than s and the other one larger than s. The sets $T_1(s)$ are pairwise disjoint. We describe later how to construct efficiently such a set S_1 of sufficient size.

We then play $\lceil \log |S_1| \rceil$ rounds of the tournament, where in each round half of the nuts survive for the next one. Intuitively, we take any two nuts together with their sets of associated bolts, determine which nut splits the union of both sets of bolts less equally, eliminate that nut, and give both sets of bolts to the surviving nut. Unfortunately, pairing the nuts arbitrarily does not quite work, i.e., the winner of the tournament would not necessarily be a c-approximate median, but pairing only nuts with small relative rank (or nuts with large relative rank, respectively) is sufficient to yield the desired result.

In general, let S_i be the set of nuts before we start round i. For each nut $s \in S_i$ let $T_i(s)$ be the set of bolts associated with s and let $r_i(s) := rank_{T_i(s)}(s)$ be the relative rank of s with respect to its set of bolts $T_i(s)$. Let $S_i^{\text{high}} := \{s \in S_i \mid r_i(s) \geq \frac{1}{2}\}$ be the nuts in S_i of high relative rank and $S_i^{\text{low}} := \{s \in S_i \mid r_i(s) < \frac{1}{2}\}$ be the set of nuts in S_i of small relative rank.

We play the *knockout tournament* as follows.

$i := 1$;
while $|S_i| > 2$ **do**

(1) Pair the nuts of S_i^{high} arbitrarily. If $|S_i^{\text{high}}|$ is odd then we eliminate the single nut which did not get a partner.

(2) Let (s_1, s_2) be a pair of nuts from S_i^{high}. Compute the relative ranks of s_1 and s_2 with respect to $T_{i+1} := T_i(s_1) \cup T_i(s_2)$. Note that it is sufficient to compare s_1 with all bolts in $T_i(s_2)$ and s_2 with all bolts in $T_i(s_1)$, because $rank_{T_{i+1}}(s_j) = \frac{1}{2}(rank_{T_i(s_j)}(s_j) + rank_{T_i(s_{3-j})}(s_j))$, for $j = 1, 2$ (this follows from Observation 5 (c)).
Whichever nut s has relative rank closer to $\frac{1}{2}$ survives in S_{i+1} and is associated with $T_{i+1}(s) := T_{i+1}$.

(3) Repeat steps (1) and (2) with S_i^{low} instead of S_i^{high}.

od

Let l be the value of i after the **while**-loop terminates, i.e., $|S_l| \leq 2$. We claim that if S_1 was sufficiently large then every nut in S_l is a c-approximate median, where c is a small constant (see Lemma 6). But first we make a few simple observations.

Observation 5. *Assume we play the tournament starting with some set S_1 of nuts. Then*

(a) $\lceil \log |S_1| \rceil - 1 \leq l \leq \lceil \log |S_1| \rceil$.

(b) $S_l \neq \emptyset$.

(c) $|T_i(s)| = 2^i$ *for* $i = 1, \ldots, l$ *and all* $s \in S_i$. *In particular,* $|T_l(s)| \geq \frac{|S_1|}{2}$ *for all* $s \in S_l$.

(d) Each round needs $O(|S_1|)$ *time.*

Proof.

(a) In each round, we eliminate half of the nuts which could be paired, and at most two unpaired nuts. Hence $|S_{i+1}| \geq \frac{|S_i|-2}{2}$. We stop if at most two nuts remain. It is now easy to show by induction on $|S_1|$ that l must be at least $\log(|S_1| + 2) - 1$. This proves the first inequality.
The second inequality follows directly from $|S_{i+1}| \leq \frac{|S_i|}{2}$.

(b) We never eliminate all nuts.

(c) By induction on i.

(d) Observe that in each round, every bolt is involved in at most one comparison (in step (2)). Since we start with $2|S_1|$ bolts in the first round, we do at most $2|S_1|$ comparisons in each round. Furthermore, pairing the nuts, computing the relative ranks, and merging the two sets of bolts does not increase the asymptotic complexity. □

Lemma 6. *Let $S_1 \subseteq S$ be of size βn. Suppose each nut $s \in S_1$ lies between the two bolts in $T_1(s) = \{b_{\text{low}}(s), b_{\text{high}}(s)\}$, i.e., $b_{\text{low}}(s) < s < b_{\text{high}}(s)$. If we play the knockout tournament on S_1 then any nut s in the final set S_l is a $\frac{\beta}{8}$-approximate median.*

Proof. Before the first round, we have $r_1(s) = \frac{1}{2}$ for all $s \in S_1$, and hence $\frac{1}{4} \leq r_2(s) \leq \frac{3}{4}$ for all $s \in S_2$. We now prove by induction that this inequality holds after each round.

So assume we know that $\frac{1}{4} \leq r_i(s) \leq \frac{3}{4}$ for all $s \in S_i$. Let (s_1, s_2) be a pair from S_i^{high} and w.l.o.g. $s_1 < s_2$. Let T_{i+1} be the set $T_i(s_1) \cup T_i(s_2)$. Since s_1 is larger than half of the bolts in $T_i(s_1)$, it must be larger than a quarter of the bolts in T_{i+1}. On the other hand, it is smaller than a quarter of the bolts in $T_i(s_1)$ and smaller than a quarter of the bolts in $T_i(s_2)$ (because it is smaller than s_2); hence it is smaller than a quarter of the bolts in T_{i+1}. Therefore, the inequality holds for s_1, and we only eliminate s_1 if the relative rank of s_2 with respect to T_{i+1} is even closer to $\frac{1}{2}$.

Now consider any arbitrary nut s in the final set S_l. Since $T_l(s)$ contains at least $\frac{\beta n}{2}$ bolts by Observation 5 (c), we conclude from the inequality above that s is larger than $\frac{\beta n}{8}$ bolts and smaller than another $\frac{\beta n}{8}$ bolts. Hence s is a $\frac{\beta}{8}$-approximate median. □

Now we show that we can construct in linear time a sufficiently large set S_1 with the property needed in Lemma 6.

Lemma 7. *We can construct in time $O(n)$ a set $S_1 \subseteq S$ of size at least $\frac{n}{12}$ and pairwise disjoint sets $T_1(s) = \{b_{\text{low}}(s), b_{\text{high}}(s)\} \subseteq T$ for all $s \in S_1$, such that $b_{\text{low}}(s) < s < b_{\text{high}}(s)$ for all $s \in S_1$.*

Proof. Connect the bolts T with the nuts S using the bipartite graph $G = (T \cup S, E)$ from Theorem 4 which can be constructed in $O(n)$ time. Let S_1 be the empty set. Then we choose arbitrary triples (t_1, s, t_2) where (t_1, s) and (t_2, s) are edges of G and $t_1 < s < t_2$, whenever there are such triples. In this case, we add s to S_1 and set $b_{\text{low}}(s) := t_1$ and $b_{\text{high}}(s) := t_2$; we also remove s from S and t_1, t_2 from T. We stop if no more such triples can be found.

We claim that this greedy procedure always yields a set S_1 of size at least $\frac{n}{12}$. This can be seen as follows. Let T_{low} be the $\frac{n}{3}$ bolts in T of smallest rank, T_{high}

be the $\frac{n}{3}$ bolts in T of highest rank, and $S_{\rm med}$ be the $\frac{n}{3}$ nuts in S of medium rank. If the greedy procedure finds less than $\frac{n}{12}$ triples $t_1 < s < t_2$, then more than $\frac{n}{6}$ bolts in $T_{\rm low}$, $\frac{n}{6}$ bolts in $T_{\rm high}$, and more than $\frac{n}{4}$ nuts in $S_{\rm med}$ have not been chosen. Since both of these sets of $\frac{n}{6}$ bolts are each connected to at least $\frac{7}{8}n$ nuts, at least one of the unchosen nuts in $S_{\rm med}$ must be connected to an unchosen bolt in $T_{\rm low}$ and an unchosen bolt in $T_{\rm high}$, a contradiction. □

We note that the proof of the Lemma above can be refined to give better constants, but that would not have a considerable impact on the total complexity of the algorithm.

Theorem 8. *In $O(n \log n)$ time, we can compute a $\frac{1}{96}$-approximate median nut s of S.*

Proof. We can find a starting set S_1 of at least $\frac{n}{12}$ nuts for the tournament in $O(n)$ time (Lemma 7). The tournament then takes $O(|S_1| \log |S_1|) = O(n \log n)$ time (Observation 5 (a),(d)) and returns a $\frac{n}{8 \cdot 12}$-approximate median (Lemma 6). □

4 Conclusions

We have presented an $O(n \log^2 n)$ time deterministic algorithm for matching nuts and bolts. This improves the previous $O(n \log^4 n)$-time solution of this problem, given by Alon *et al.* [2], by a factor of $\log^2 n$. As already mentioned in [2], the methods described in this (and their) paper seem not to be sufficient to reduce the complexity below $O(n \log^2 n)$.

On the other hand, [6] and [9] have proved the existence of a deterministic $O(n \log n)$ algorithm, the first paper based on $O(n)$ selection, the second one based on the AKS sorting network. It would be nice to find an explicit optimal algorithm.

References

1. M. Ajtai, J. Komlós, and E. Szemerédy. *An $O(n \log n)$ sorting network.* Proceedings of the 15^{th} ACM Symposium on the Theory of Computing (STOC'83), 1983, pp. 1–9.
2. N. Alon, M. Blum, A. Fiat, S. Kannan, M. Naor, and R. Ostrovsky. *Matching nuts and bolts.* Proceedings of the 5^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94), 1994, pp. 690–696.
3. N. Alon, Z. Galil, and V.D. Milman. *Better expanders and superconcentrators.* Journal of Algorithms **8** (1987), pp. 337–347.
4. N. Alon, and J. Spencer. *The Probabilistic Method.* John Wiley and Sons Inc., New York, 1992.

5. P.G. Bradford, and R. Fleischer. *Matching nuts and bolts faster.* Proceedings of the 6^{th} International Symposium on Algorithms and Computation (ISAAC'95), 1995, pp. 402–408.

6. P.G. Bradford. *Matching nuts and bolts optimally.* Technical Report MPI-I-95-1-025, Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany, September 1995.

7. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

8. W. Goddard, C. Kenyon, V. King, and L. Schulman. *Optimal randomized algorithms for local sorting and set-maxima.* SIAM Journal on Computing **22** (1993), pp. 272–283.

9. J. Komlós, Y. Ma, and E. Szemerédi. *Matching nuts and bolts in $O(n \log n)$ time.* Proceedings of the 7^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96), 1996, pp. 232–241.

10. A. Lubotzky. *Discrete Groups, Expanding Graphs and Invariant Measures.* Birkhäuser Verlag, 1994.

11. A. Lubotzky, R. Phillips, and P. Sarnak. *Explicit expanders and the Ramanujan conjectures.* Proceedings of the 18^{th} ACM Symposium on the Theory of Computing (STOC'86), 1986, pp. 240–246.
See also: A. Lubotzky, R. Phillips, and P. Sarnak. *Ramanujan graphs.* Combinatorica **8** (1988), pp. 261–277.

12. G.A. Margulis. *Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators.* Problemy Peredachi Informatsii **24** (1988), pp. 51–60 (in Russian).
English translation in Problems of Information Transmission **24** (1988), pp. 39–46.

13. J.I. Munro, and M. Paterson. *Selection and sorting with limited storage.* Theoretical Computer Science **12** (1980), pp. 315–323.

14. G.J.E. Rawlins. *Compared to What ? An Introduction to the Analysis of Algorithms.* Computer Science Press, 1992.