A Divide and Conquer Approach to Sequencing

Dan Boneh
dabo@cs.princeton.edu

Richard J. Lipton*
rjl@cs.princeton.edu

Department of Computer Science
Princeton University
Princeton, NJ 08544

Abstract

We suggest an approach to sequencing based on a "divide and conquer method". This approach eliminates the need for solving a hard NP-complete problem which arises when using traditional sequencing techniques. At the present time this algorithm has not been tried out in the lab. However, computer simulations using parts of the human genome provide some encouraging data.

1 Introduction

The problem of sequencing long DNA strands is one of the central problems facing biology today. Traditionally this is done in two steps. First, the long strand is broken into many overlapping short strands and each short strand is sequenced separately. Then the information is combined to obtain the entire long strand. This last step involves solving a hard NP-complete problem. This procedure is known as the *double digest* method.

In this paper we propose a new approach for sequencing based on a divide and conquer algorithm which is similar to quicksort, a widely used sorting algorithm [2]. We therefore refer to this algorithm as *quick-sequence*. This algorithm avoids the difficulty in recombining the information which arises when using the traditional approach. It should be pointed out that this algorithm has not been tested in the lab. It is still not clear whether it actually works in practice. However, even if the algorithm does not work at its present form, we hope that this paper will motivate researches to look into this line of research.

2 High Level Overview of the Algorithm

The algorithm will sequence a DNA strand S of length n. The value of n can be thought of as 10^6 though the algorithm works for any n. We assume that the strand S has a known sequence L of length 100 on one end and a known sequence R of length 100 on the other end. This can be achieved in various ways. Possibly the simplest way is to append some fixed strands L and R to the ends of the strand.

The strand S to be sequenced is given in a test tube containing many copies of S. The algorithm can be described as follows:

*Supported in part by NSF CCR–9304718.

1. Take some DNA strand from the test tube and use restriction enzymes to cut the strand into many pieces, each about 100 long. Randomly pick one of the pieces and sequence it. This piece will be called the pivot and we denote it by P.

2. For now we assume that the pivot P appears only once in the strand S. Thus, the strand S looks like $L - \cdots - P - \cdots - R$. The idea is to now break S around the pivot. To be precise, we would like to produce two solutions T_l and T_r.

   The solution T_l will contain copies of the strand $L - \cdots - P$.
   The solution T_r will contain copies of the strand $P - \cdots - R$.

   There are many ways of creating the solutions T_l and T_r. In the next section we give two potential methods. One method is based on PCR and the other method is based on restriction enzymes.

3. Recursively sequence the strands in T_l and in T_r. Note that the strands in T_l have L on one end and P on the other end. Hence, in T_l we can use L as the left marker and P as the right marker. A similar condition holds in T_r.

4. Once we sequence T_l and T_r we obtain two strings $S_l = L \cdots P$ and $S_r = P \cdots R$. It is now clear how to combine them to obtain the string S.

An example of the recursion tree is shown in Figure 1. Since it is possible to sequence 500 long DNA strands, the recursion can stop once the sequences are shorter than 500 base pairs. Since most base pairs in S are sequenced exactly once (the pivots are sequenced twice) the total amount of work involved is linear in the length of the original long strand S. For a strand of length $n = 10^6$ the recursion depth is likely to be less than 15.

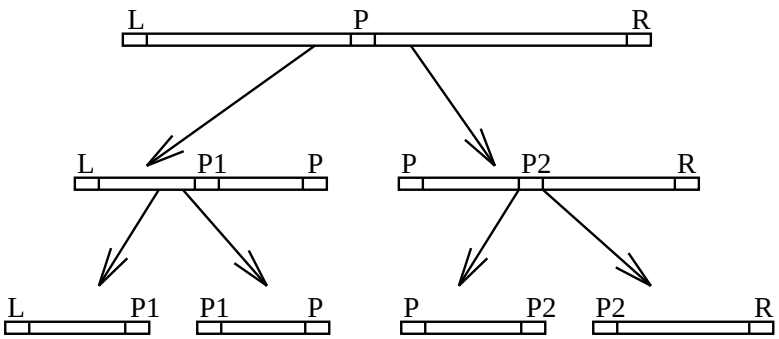

Figure 1: Recursion tree

Let $T(n)$ be the time needed to sequence an n long DNA strand. Then $T(n)$ satisfies the following recurrence:

$$T(n) = T(k) + T(n-k) + c$$

for some integer k between 0 and n where c is the time it takes to separate a strand into its left and right components. The solution to this recursion can be shown to be $T(n) = cn$. Hence, the time it takes to sequence a strand is proportional to the length of the strand.

In step 2 of the algorithm we made the assumption that the pivot P appears only once in the strand S. We now show that this condition can be relaxed at the expense of increasing the running

time of the algorithm. Suppose the strand S contains two copies of the pivot P, i.e. it looks like

$$L - \cdots - P - \cdots - P - \cdots - R$$

As before we generate two solutions T_l and T_r. The strands in T_l will extend all the way from the left to the right most occurrence of the pivot P in S. Similarly, the strands in T_r will extend all the way from the right to the left most occurrence of the pivot. Formally:

The solution T_l will contain the strands: $L - \cdots - P - \cdots - P$.
The solution T_r will contain the strands: $P - \cdots - P - \cdots - R$.

In the next section we suggest some procedures for separating the solution into T_l and T_r. These procedures have the property that when the pivot in not unique we obtain the required solutions T_l and T_r described above.

We see that when the pivot P appears more than once the strands in T_l and T_r overlap. This implies that some base-pairs will be sequenced more than once: the base pairs in the overlap between the left and right recursive steps will be sequenced at least twice. This phenomenon is shown in Figure 2. Note however that recombining the left and right strings (step 4 of the algorithm) is still easy.

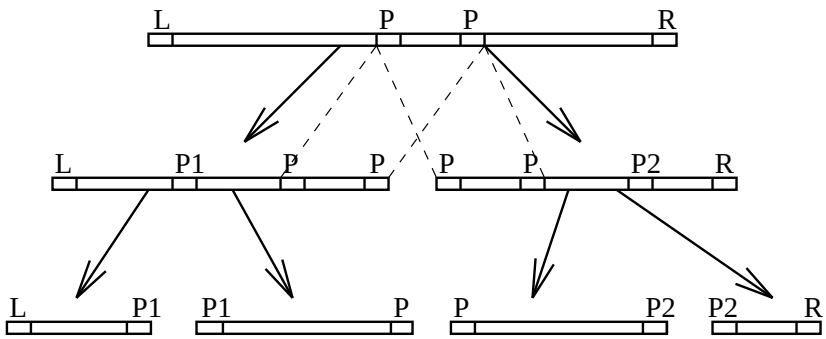

Figure 2: Recursion tree using a bad pivot P.

There are several ways to improve the choice of the pivot P. First of all, one could use longer pivots. Hopefully, the longer the pivot is the less likely it is to appear twice in the strand S. Another approach is to test the quality of the chosen pivot. Let A_l, A_r be the length of strands in T_l, T_r respectively. If $A_l + A_r$ is much larger than the length of S then clearly the chosen pivot performs poorly. Presumably, testing if $A_l + A_r$ is larger than S can be done using gel electrophoresis. Once it has been determined that the chosen pivot is bad a new one can be picked. Note that this test need only be done in the initial stages of the recursion since those are the crucial steps. A bad pivot in the first step would slow down the algorithm considerably. As was observed before, the higher level of the recursion tree contain very few nodes. Hence, it makes sense to spend the time and make sure that a good pivot has been chosen in those few steps. In addition, it always makes sense to run standard statistical tests to make sure that the pivot is close to being a random string. The reason is that if the pivot was a truly random string of sufficient length then it would be unlikely to occur more than once in the DNA strand being sequenced.

3 Implementation of step 2

We now turn our attention to the implementation of step 2 of the algorithm. Given a solution containing many copies of the strand S and a pivot P, we wish to generate the two solution T_l and T_r described above.

3.1 First Implementation: PCR based approach

We begin by discussing how the procedure works when the pivot P appears only once in the strand S. To produce T_l and T_r we use PCR [3] as follows: melt the double strands in the solution to single strands. Use P and it's Watson-Crick complement as primers and apply a polymerase enzyme. When the reaction is over melt the strands into single strands again. Since a polymerase enzyme only elongates from the 5' end to the 3' end the resulting strands in the solution are now:

$$5' - L - \cdots - P - \cdots - R - 3'$$
$$3' - \hat{L} - \cdots - \hat{P} - \cdots - \hat{R} - 5'$$
$$3' - \hat{P} - \cdots - \hat{R} - 5'$$
$$5' - L - \cdots - P - 3'$$

Where for a string x the string $\hat{x}$ is the Watson-Crick complement of x. We can now run through the same process again using the primers $\hat{L}_0$ and R. This will transform the single strands into double strands. Using gel electrophoresis we can get rid of the long strands. This will result in a solution containing the double strands $L - \cdots - P$ and $P - \cdots - R$. Finally, to produce T_l (resp. T_r) extract all strands containing L (resp. R). This type of extraction is a widely used biological operation.

A useful feature of this method is that whenever PCR is used the primers are already prepared in advance. The primers are simply the pivots used in previous recursion steps. Hence, to run the algorithm there is no need to use special machinery to generate specific primers.

There is an important caveat. In the first few steps of the recursion, PCR has to be performed on very long strands, e.g. of length 10^6. It is not clear whether the polymerase enzyme can work with such long strands. Potentially, the enzyme is capable of replicating long strands, though is takes a long time to do so. If this were the case then the algorithm can still be useful. This is true since the number of long replications is very small. They only occur at the initial steps of the recursion. Hence, waiting a long time for long replications to complete is not likely to effect the running time of the algorithm. On the other hand, if the polymerase enzyme can not be made to replicate long strands for some fundamental reason then the algorithm is flawed(unless other methods for doing this replication can be found).

We know of successes in doing PCR involving DNA strands containing 42,000 base pairs [1]. This implies that using current technology our technique can be made to sequence DNA strands of length 84,000. It is important to point out that standard PCR consists of many cycles. In each cycle the number of DNA strands roughly doubles. However, for our purposes we only need to run through *one* cycle of the PCR process. It is therefore quite possible that one can run through one cycle of the PCR process with much longer strands.

We now discuss how this procedure works when the pivot P is not unique. After adding P and $\hat{P}$ as primers and allowing a polymerase enzyme to act on the solution we obtain strands of the form:

$$5' - L - \cdots - P - \cdots - P - \cdots - P - \cdots - R - 3'$$
$$3' - \hat{L} - \cdots - \hat{P} - \cdots - \hat{P} - \cdots - \hat{P} - \cdots - \hat{R} - 5'$$

$$5' - L - \cdots - P - \cdots - P - \cdots - P - 3'$$
$$3' - \hat{P} - \cdots - \hat{P} - \cdots - \hat{P} - \cdots - \hat{R} - 5'$$

Note that to obtain these strands we must use a DNA ligase. The ligase must be used to connect the chain anchored at one primer to the chain anchored at the next primer. It is unclear whether a DNA ligase can be used in conjunction with a polymerase enzyme. However, it is our understanding that the ligase chain reaction [4] does exactly that.

Once we obtain these strands we can follow the same procedure to produce the solution T_l and T_r. The solution T_l will now contain strands that look like

$$L - \cdots - P - \cdots - P - \cdots - P$$

where the right most P corresponds to the occurrence of P in S which is the furthest away from L. The strands in T_r have a similar form. Hence, there is some overlap between the strands in T_l and those in T_r. The overlap is between the left most and right most occurrences of the pivot P in S.

3.2 Second Implementation: Restriction Enzymes

The second implementation we propose does not require the use of PCR in the first few steps. To produce the solutions T_r and T_l we use some restriction enzyme. Let E be the recognition site for this enzyme. The enzyme will cut positions in the strand containing the string E. We require that the recognition site E for this enzyme be sufficiently rare in the strand S being sequenced, e.g. it should only appear at most 20 times in the strand.

The idea is to use a small amount of the enzyme in the solution. Thus only a small fraction of the recognition sites E will be cut. To create the solution T_l we can now extract all strands that contain both the left marker L and the pivot P. A strand will be extracted only if the enzyme did not cut at any of the recognition sites E that lie between L and P. The extracted strands look like:

$$L - \cdots - P - \cdots - U_1$$
$$L - \cdots - P - \cdots - U_1\ E - \cdots - U_2$$
$$L - \cdots - P - \cdots - U_1\ E - \cdots - U_2\ E - \cdots - U_3$$
$$L - \cdots - P - \cdots - U_1\ E - \cdots - U_2\ E - \cdots - U_3\ E - \cdots - R$$

where U_1, U_2, U_3 are unknown pieces to the right of the pivot. We can now extract the shortest strands in this solution using gel electrophoresis. The set of shortest strands will be used as T_l. The solution T_r is constructed in a similar manner.

Note that this method will not work if the pivot P appears more than once on the strand S since the section of the strand between the first occurrence and last occurrence of P will be lost. There are ways of making this method work even when P is not unique. We do not discuss these methods here.

4 Experimental Results

To test the algorithm we ran computer simulations on a 100,000 long sequence taken from the human genome. We used pivots of length 100. Without testing the quality of the chosen pivot the overhead was roughly 5, i.e. each base pair was sequenced 5 times. By testing the quality of the chosen pivot in the initial steps of the algorithm the overhead dropped to 1.5. This means that by choosing good pivots, each base pair was sequenced one and a half times on average.

5 Summary

In this paper we propose a different approach to DNA sequencing. Our algorithm can sequence large DNA strands with little overhead. The algorithm is easily implemented and could potentially be entirely automated. Furthermore, it can be easily parallelized. Parallelism is important when sequencing large DNA strands since the number of short fragments that have to be sequenced is large.

Since our approach has not been tested in the lab we can not guarantee that it actually works. There are two biological issues that are still open. First, it is not clear whether a polymerase enzyme can be applied to very long DNA strands, e.g. of length 10^6. Second, we do not know whether a DNA ligase can be used in conjunction with a polymerase enzyme. Both steps seem crucial for this algorithm to work.

It should also be pointed out that it is not clear whether the procedure we propose is less time consuming than the traditional approach to sequencing. We hope this paper will motivate researchers to test the algorithm and compare it to other approaches.

Finally, our computer simulations are encouraging. They show that it is possible to sequence DNA with very little overhead while still avoiding the need to solve an NP-complete problem.

Acknowledgments

The authors wish to thank Dr. Mike Lipkowitz for many helpful comments on this manuscript.

References

[1] W. Barnes, "PCR amplification of up to 35-kb DNA with high fidelity and high yield from λ bacteriophage templates", Proc. Natl. Acad. Sci., Vol. 91, March, 1994, pp. 2216–2220.

[2] T. Corman, C. Leiserson, R. Rivest, "Introduction to Algorithms", McGraw Hill, 1991.

[3] H. Erlich, "PCR Technology: Principled and Applications for DNA Amplification", Stockton Press.

[4] F. Barany, "The ligase chain reaction in a PCR world", PCR-methods-Appl., 1991, pp. 5–16.