**CS5206**

**DNSRA – M3**

**FINAL REPORT**

**T01**

* **HOANG NGUYEN NHAT TAO**
* **BI RAN**

**DNSRA FINAL REPORT  
  
1. PROBLEM STATEMENT**

DNSRA is the abbreviation for De novo Short Read Assembly. The DNSRA problem can be stated as follows:

*Given a set of short reads, assemble them to form the original DNA sequence.*

The process of producing the set of reads includes: making many copies of the original DNA sequence, breaking them randomly into DNA fragments, selecting DNA fragments of a particular length and finally sequencing the start and end of each selected fragment to achieve the pair of reads. For simplicity, we only consider short reads of fixed length 30 and target errors caused by misreading bases on reads.

Since recovering the complete original DNA sequence is almost impossible in practice, the expected output is a set of contigs. A contig is a contiguous sequence of the targeting DNA sequence which is produced by combining multiple reads and should be longest possible.

**2. ALGORITHM OVERVIEW**

Our algorithm is an improved version of the original one used in Edena software. The algorithm consists of two main parts:

1. Error correction
2. Modified Edena

The error correction part is motivated by greedy algorithm. It first replaces all the ambiguous bases (base N) with a randomly selected base (A, T, G, C), and modify reads iteratively to reduce error.

Notice that the corrected file is still of fna format, it can be used directly as input of other assemblers which supports fna format.

The modified Edena part is slightly different from the original Edena. This part consists of eight subparts:

1. Read in data, ignore dulplicated reads.
2. Build overlap graph
3. Remove transitive edges
4. Non-strict clearing if non-strict mode is on
5. Build a compressed graph
6. Remove short ends from the compressed graph
7. Remove small bubbles from the compressed graph
8. Produce contigs from the compressed graph

More details about each part of the algorithm will be discussed in the next section.

**3. ALGORITHM DETAILS**

In the previous section, we have the overview of the algorithm. Let us now examine the details of each part of the algorithm:

**Error Correction**

replace all “N” bases with a random base from {A, T, G, C}.

**repeat** ITER\_NUMBER times

**for** MIN\_OVERLAP\_LEN **from** RANGE\_ST to RANGE\_ED **step** DELTA

**foreach** read *r* **in** read\_list

find the best extend on both left and right hand side of *r*

**if** both left and right overlaps are at least MIN\_OVERLAP\_LEN

add the read into new\_read\_list

**else**

**forall** r’ with exactly one different base from r

find the best left and right extend to r’

add the r’ with the largest sum of overlap length into new\_read list

read\_list = new\_read\_list

To find the best left and right extend, suffix and prefix array is used. Best left and right extend for read *r* refer to the reads with longest overlapping length. RANGE\_ST, RANGE\_ED and DELTA are adjustable parameters. By default, RANGE\_ST=14, RANGE\_ED=27, DELTA=2.

**Modified Edena**

* *Build overlap graph*

We first find the overlaps between each pair of reads with the MIN\_OVERLAP\_SIZE. MIN\_OVERLAP\_SIZE can also be customized as an argument of the program. We then construct the overlap graph. Each read is represented by a vertex. Two vertices are connected if two corresponding reads have overlap longer than the MIN\_OVERLAP\_SIZE. The connecting edge is labeled with overlap length. Trie is used to build the graph efficiently.

* *Remove transitive edges*

We simplify the overlap graph by removing transitive edges with some vertices. In particular, we remove reads for which another read with the same sequence overlaps the first read to a greater extent. For instance, suppose we have two paths: r1🡪r2🡪r3 and r1🡪r3. In this case, we remove the edge r1🡪r3. In the view of reads, one possible example of r1, r2 and r3 is:

r1: GTAATTGCCATCGTTGT  
r2: TAATTGCCATCGTTGTA  
r3: AATTGCCATCGTTGTAC

The overlap between r1 and r2 (TAATTGCATCGTTGTA) is larger than the overlap between r1 and r3 (AATTGCCATCGTTGT). Therefore, edge r1🡪r3 is removed from the graph.

By removing transitive edges, the total number of edges in the graph drops drastically, while the most important information is preserved.

* *Non-strict clearing*

Original edena allows users to choose non-strict mode. Under non-strict mode, when a branching node is reached while producing contigs, edena will choose the branch with longer overlap instead of stopping at that node. Under non-strict mode, edena may produce longer contigs at the risk of generating mismatches.

Our edena also supports non-strict mode. However, we have sub-modes under the “non-strict” mode: normal non-strict, very non-strict.

In normal non-strict mode, at each branching node, we simply delete those branch edges with length less than some threshold. By default, the threshold is MIN\_OVERLAP\_LEN + 4.

We also remove “reverse branches” (multiple edges pointing to the same vertex) by doing the above step on the reverse graph of overlapping graph.

Very non-strict mode is almost the same as normal non-strict mode except that the threshold is chosen to be the length of the longest edge at that branch/reverse branch.

* *Build compressed graph*

After removing transitive edges, there will be a lot of chains in the overlap graph. A chain is defined as a maximal path with all intermediate nodes having indegree and outdegree of exactly one. Therefore, we build a new compressed graph by compressing each chain in the overlap graph into one edge. The length of edge is the length of the chain. Compressed graph further simplifies the overlap graph. It is possible to print out the graph into an image file using *dot* utility. Visualizing graph structure can help parameter selection in practice.

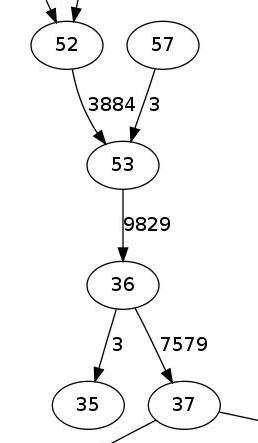
All the rest steps are done on this compressed graph.

* *Remove short ends from the graph*

In the original Edena, short branches are removed from the overlap graph.

In compressed graph, a vertex may have multiple paths branching from itself, and multiple edges pointing to it. Original edena algorithm only removes short branches going out of a node. However, we observe that some vertices with indegree more than 2 have many short incomming edges, which may be caused by errors. Thus our modifed edena algorithm not only removes short edges branching out, but also short incomming edges.

Generally, our edena removes edge r1🡪r2 if and only if the length of edge is smaller than the threshold (10 by default), and it has either indegree of r1 is 0, or outdegree of r2 is 0.



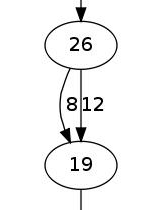
In the above graph, we remove the edge from 36 to 35, which is the short branching-out edge and then we also remove the edge from 57 to 53, which is a short incoming edge.

* *Remove small bubbles from the compressed graph*

Original edena removes bubbles from the graph. However, the definition of bubble is not clear because general directed graph may have complicated structures. In our implementation, we define bubbles as following:

*If two edges are going from r1 to r2 in compressed graph, then these two edges forms a bubble.*

Our edena always removes the shorter edge, keeps the longer edge, which is the same as the original edena algoritm.



For example, in the above graph, we remove the edge from 26 to 19 with weight 8.

* *Produce reads from the compressed graph*

Our method is very similar to original edena. Original edena looks for chains in overlap graph, which is the same as looking for edges in compressed graph. We also set a MIN\_CONTIG\_SIZE threshold to eliminate very short contigs since they are more likely to contain errors.

**foreach** edge **in** the compressed graph

**if** the length of edge is at least MIN\_CONTIG\_SIZE

**print** edge

MIN\_CONTIG\_SIZE is 100 by default, and adjustable.

Note that the compressed graph is compressed (replace chains by an edges) after each clearing operation.

**4. COMPLEXITY ANALYSIS**

Because of complicated structure of our implementation, we only give approximated upper bound of time and space complexity. Better complexity may be obtained by more efficient implementation or more detailed analysis.

**4.1 TIME COMPELXITY**

Notation:

N=the number of reads in the input file

L=the length of reads, always 30 in our dataset

**Error Correcting Program:**

Read in file O(NlogNL)

In each iteration: (assume G iterations)

building suffix/prefix array: O(LN)

for each read r: (N reads)

for all r’ with one base modification: (4L-1 different r’)

find best extend O(L+logN)

Thus time complexity for error correcting program is O(NlogNLG(NL+N(L2+LlogN)))

If L is considered as constatn(30 in our dataset), then the complexity becomes O(GN2log2N)

Usually, the number of reads requiring correction decreases as iteration number increases, thus the program does not run very slow in practice.

For our largest dataset reads504.fna, it takes 11-12 minutes on tembusu with default parameters.

**Modified Edena:**

Read in file O(NlogNL)

Build Trie O(NL2)

Build overlap graph O(NL2) assuming each node has indegree/outdegree L

Remove transitive edges O(NL)

Non-strict clearning O(NL)

Build compressed graph O(NL)

Clear short edges O(NL)

Clear bubbles O(NL)

Produce contigs O(NL)

The total complexity is about O(NlogNL+NL2). If consider L as constant, then the complexity is O(NlogN). In practice, running on the largest dataset reads504.fna(after error correction) takes 2 minutes on tembusu. Note that the running time in practice heavily depends on the input dataset. If the input dataset contains a lot of errors, then the running time grows drastically because overlap graph and compressed graph tends to have more complicated structure, and clearing the graph is more time consumming.

**4.2 SPACE COMPELXITY**

**Error Correcting Program:**

Our error correcting program is very space efficient. The only nontrivial space cost is prefix/suffix array, which is only O(NL).

**Modified Edena:**

The space complexity is harder to analysis for our edena because the size of compressed graph, overlap graph and trie dependes on the error rate of dataset. An approximated space complexity is O(NL), which is the size of overlap graph before clearing.

For our largest dataset reads504.fna(after error correction), our edena requires about 1.5G memory.

**5. RESULTS AND OBSERVATIONS**

To save pages, we only compare large datasets(reads3xx, reads4xx, reads5xx).

All dataset with errors has been corrected using error correction program.

**5.1. ERROR CORRECTION RESULT**

We will compare the number of reads occurred in the original genome (**in**), the number of reads not occurred in the original genome (**not in**) and the **error rate(not in/(in+not in))** before and after the error correction(default parameters). The error correction program uses default parameters.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Before** |  |  | **After** |  |
| **file name** | **in** | **not in** | **error rate** | **in** | **not in** | **error rate** |
| 302 | 82195 | 50869 | 0.38229 | 94087 | 39 | 0.000414 |
| 303 | 82556 | 50755 | 0.380726 | 94408 | 47 | 0.000498 |
| 304 | 106681 | 101756 | 0.488186 | 112494 | 82 | 0.000728 |
| 305 | 35490 | 35617 | 0.500893 | 56608 | 265 | 0.00466 |
| 306 | 35497 | 35633 | 0.500956 | 56457 | 312 | 0.005496 |
| 402 | 435256 | 266979 | 0.380185 | 497905 | 1027 | 0.002058 |
| 403 | 435710 | 266888 | 0.379859 | 498617 | 1030 | 0.002061 |
| 404 | 562850 | 532765 | 0.48627 | 593021 | 2303 | 0.003868 |
| 405 | 187499 | 186734 | 0.498978 | 297478 | 3045 | 0.010132 |
| 406 | 187420 | 186701 | 0.499039 | 297383 | 3110 | 0.01035 |
| 502 | 1090303 | 672561 | 0.381516 | 1246156 | 2422 | 0.00194 |
| 503 | 1091278 | 672730 | 0.381364 | 1246983 | 2269 | 0.001816 |
| 504 | 1405732 | 1344141 | 0.488801 | 1480347 | 5037 | 0.003391 |
| 505 | 472036 | 470799 | 0.499344 | 747399 | 7198 | 0.009539 |
| 506 | 471189 | 471049 | 0.499926 | 746395 | 7436 | 0.009864 |

It can be observed from the table that the error correction program has better performance for smaller datasets.

**5.2. EDENA RESULT**

**Modified Edena(default parameters)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **file number** | **M** | **N** | **B** | **L** | **Score** |
| 301 | 116555 | 117080 | 117080 | 17633 | 5.903743 |
| 302 | 116551 | 117080 | 117080 | 17633 | 5.903675 |
| 303 | 116556 | 117080 | 117080 | 17633 | 5.90376 |
| 304 | 116555 | 117080 | 117080 | 11916 | 5.338365 |
| 305 | 114516 | 117080 | 117080 | 608 | 1.956201 |
| 306 | 110982 | 117080 | 117080 | 684 | 1.895832 |
| 401 | 615952 | 615980 | 615980 | 597914 | 8.60082 |
| 402 | 615975 | 615980 | 615980 | 89399 | 5.859287 |
| 403 | 615980 | 615980 | 615980 | 137162 | 6.476854 |
| 404 | 615980 | 615980 | 615980 | 102102 | 6.050984 |
| 405 | 602833 | 615980 | 615980 | 661 | 1.957314 |
| 406 | 599699 | 615980 | 615980 | 709 | 1.947138 |
| 501 | 1533307 | 1553927 | 1553927 | 14280 | 1.973461 |
| 502 | 1533327 | 1553927 | 1553927 | 14280 | 1.973487 |
| 503 | 1533367 | 1553927 | 1553927 | 14288 | 1.973538 |
| 504 | 1533369 | 1553927 | 1553927 | 14280 | 1.973541 |
| 505 | 1510354 | 1553927 | 1553927 | 658 | 1.943919 |
| 506 | 1502952 | 1553927 | 1553927 | 646 | 1.934392 |

**Modified Edena With Normal Non-Strict Mode**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **file number** | **M** | **N** | **B** | **L** | **Score** |
| 301 | 116582 | 117080 | 117080 | 17640 | 5.904777 |
| 302 | 116558 | 117080 | 117080 | 17643 | 5.904612 |
| 303 | 116587 | 117080 | 117080 | 31331 | 6.733603 |
| 304 | 116586 | 117080 | 117080 | 17640 | 5.904845 |
| 305 | 114087 | 117080 | 117080 | 572 | 1.948873 |
| 306 | 110610 | 117080 | 117080 | 588 | 1.889477 |
| 401 | 615980 | 615980 | 615980 | 597914 | 8.600911 |
| 402 | 615913 | 615980 | 615980 | 98305 | 5.996092 |
| 403 | 615980 | 615980 | 615980 | 54985 | 5.158083 |
| 404 | 615980 | 615980 | 615980 | 613670 | 8.638436 |
| 405 | 600150 | 615980 | 615980 | 562 | 1.948602 |
| 406 | 597034 | 615980 | 615980 | 601 | 1.938485 |
| 501 | 1535013 | 1553927 | 1553927 | 26488 | 2.745077 |
| 502 | 1534918 | 1553927 | 1553927 | 16716 | 2.080845 |
| 503 | 1534808 | 1553927 | 1553927 | 20256 | 2.357823 |
| 504 | 1534982 | 1553927 | 1553927 | 26488 | 2.745037 |
| 505 | 1500165 | 1553927 | 1553927 | 556 | 1.930805 |
| 506 | 1495416 | 1553927 | 1553927 | 541 | 1.924693 |

**Modified Edena With Very Non-Strict Mode**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **file number** | **M** | **N** | **B** | **L** | **Score** |
| 301 | 22484 | 117080 | 117080 | 0 | 0.384079 |
| 302 | 116641 | 117080 | 117080 | 67767 | 7.847517 |
| 303 | 116592 | 117080 | 117080 | 84337 | 8.162263 |
| 304 | 116593 | 117080 | 117080 | 84337 | 8.16228 |
| 305 | 112690 | 117080 | 117080 | 548 | 1.925009 |
| 306 | 110821 | 117080 | 117080 | 668 | 1.893082 |
| 401 | 615980 | 615980 | 615980 | 597914 | 8.600911 |
| 402 | 615975 | 615980 | 615980 | 155251 | 6.655559 |
| 403 | 615980 | 615980 | 615980 | 82819 | 5.749006 |
| 404 | 615980 | 615980 | 615980 | 329012 | 7.739113 |
| 405 | 601579 | 615980 | 615980 | 610 | 1.953242 |
| 406 | 598183 | 615980 | 615980 | 655 | 1.942216 |
| 501 | 1465343 | 1553927 | 1553927 | 28401 | 2.75601 |
| 502 | 1535531 | 1553927 | 1553927 | 22966 | 2.539904 |
| 503 | 1482334 | 1553927 | 1553927 | 20110 | 2.27985 |
| 504 | 1535596 | 1553927 | 1553927 | 28135 | 2.832854 |
| 505 | 1505967 | 1553927 | 1553927 | 598 | 1.938273 |
| 506 | 1499272 | 1553927 | 1553927 | 588 | 1.929656 |

**Original Edena(default parameters)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **file number** | **M** | **N** | **B** | **L** | **Score** |
| 301 | 116085 | 117080 | 117080 | 9157 | 4.950383 |
| 302 | 116078 | 117080 | 117080 | 9157 | 4.950264 |
| 303 | 116082 | 117080 | 117080 | 9377 | 4.984584 |
| 304 | 116080 | 117080 | 117080 | 9157 | 4.950298 |
| 305 | 112655 | 117080 | 117080 | 584 | 1.924411 |
| 306 | 108745 | 117080 | 117080 | 623 | 1.857619 |
| 401 | 615824 | 615980 | 615980 | 35828 | 4.539626 |
| 402 | 615929 | 615980 | 615980 | 28571 | 4.213431 |
| 403 | 615928 | 615980 | 615980 | 35829 | 4.540004 |
| 404 | 615932 | 615980 | 615980 | 34460 | 4.483811 |
| 405 | 597480 | 615980 | 615980 | 648 | 1.939933 |
| 406 | 592830 | 615980 | 615980 | 685 | 1.924835 |
| 501 | 1518182 | 1553927 | 1553927 | 9325 | 1.953994 |
| 502 | 1518342 | 1553927 | 1553927 | 9486 | 1.9542 |
| 503 | 1518280 | 1553927 | 1553927 | 9693 | 1.95412 |
| 504 | 1518409 | 1553927 | 1553927 | 9464 | 1.954286 |
| 505 | 1484707 | 1553927 | 1553927 | 637 | 1.91091 |
| 506 | 1474473 | 1553927 | 1553927 | 619 | 1.897738 |

**Original Edena With Non-Strict Mode**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **file number** | **M** | **N** | **B** | **L** | **Score** |
| 301 | 116391 | 117080 | 117080 | 22247 | 6.236275 |
| 302 | 116454 | 117080 | 117080 | 22250 | 6.237545 |
| 303 | 116494 | 117080 | 117080 | 22250 | 6.238229 |
| 304 | 116471 | 117080 | 117080 | 22250 | 6.237836 |
| 305 | 112170 | 117080 | 117080 | 601 | 1.916126 |
| 306 | 105872 | 117080 | 117080 | 647 | 1.808541 |
| 401 | 615871 | 615980 | 615980 | 391841 | 7.990887 |
| 402 | 615939 | 615980 | 615980 | 391581 | 7.99015 |
| 403 | 615950 | 615980 | 615980 | 142038 | 6.527152 |
| 404 | 615949 | 615980 | 615980 | 401040 | 8.024618 |
| 405 | 597475 | 615980 | 615980 | 648 | 1.939917 |
| 406 | 593062 | 615980 | 615980 | 692 | 1.925588 |
| 501 | 1521969 | 1553927 | 1553927 | 22965 | 2.522386 |
| 502 | 1446163 | 1553927 | 1553927 | 21270 | 2.314202 |
| 503 | 1516433 | 1553927 | 1553927 | 23252 | 2.533179 |
| 504 | 1521252 | 1553927 | 1553927 | 23488 | 2.55395 |
| 505 | 1471590 | 1553927 | 1553927 | 643 | 1.894027 |
| 506 | 1470622 | 1553927 | 1553927 | 634 | 1.892781 |

* The running time of the error correction program is a bit long. It takes approximately 11-12 minutes with seven iterations to perform corrections on the largest dataset (reads504.fna). The range and increasing step of MIN\_OVERLAP\_LEN can be adjusted to obtain better performance for specific datasets.
* The modified Edena algorithm is quite fast. It takes only 2 minutes to process the largest dataset (reads504.fna). The original Edena is about twice slower than our modified Edena. Note that it will take much longer time to process the original erroneous dataset. Erroneous dataset results in much more complicated graph, which is very time-consuming to clear short edges and bubbles.
* In terms of memory usage, the algorithm requires about 1.5GB memory to run on the largest data set. It costs about 5 times more memory than the original Edena. Better implementation may reduce the memory usage.
* There are several parameters in the algorithm which can be adjusted to obtain better performance: MIN\_OVERLAP\_LEN, MIN\_CONTIG\_SIZE, MIN\_DEAD\_END\_LEN.
* Lower MIN\_OVERLAP\_LEN appears to be better for those datasets with low coverage and high error rate. MIN\_OVERLAP\_LEN = 20 is good for “reads\*05.fna” and “reads\*06.fna” datasets according to our experiment.
* The performance of the modified Edena(default parameters) algorithm is better than the original Edena(default parameters) algorithm in most cases.
* The very non-strict mode of modified Edena is unstable. It produces longer contigs for some dataset, while it gets very low score for a few datasets.
* The normal non-strict mode of modifed Edena is more stable. Some results are as good as original Edena(non-strict mode), while the others are not as good.

**6. CONCLUSION**

This algorithm is an modified version of the original algorithm in the Edena software. There are many improvements such as error correction part, compressed graph construction, etc. Actual experiments and results show that the performance of the algorithm is slightly better than the original Edena in most cases and its running time is faster despite higher memory usage.

**References**

David Hernandez, P. F. (2007, September 28). *De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer*. Retrieved March 5, 2008, from Genome Research: http://genome.cshlp.org/content/18/5/802.full