KOREKSI *DNA SEQUENCING ERROR* DENGAN METODE SPECTRAL ALIGNMENT YANG DIMODIFIKASI

DNA SEQUENCING ERROR CORRECTION WITH MODIFIED SPECTRAL ALIGNMENT METHOD

Gerry Indramades Almi [1], Wisnu Ananta Kusuma [2*]

Departemen imu Komputer, FMIPA, Institut Pertanian Bogor, Bogor [1,2]
ananta@ipb.ac.id, Kampus IPB Dramaga, Jl. Meranti Wing 20 LV. V, Bogor 16680 [2*]

ABSTRACT

Graph based DNA sequence assembly is actually used to generate contigs form reads produced by second generation sequencer. However, graph based DNA sequence assembly is very sensitive against sequencing error. The existence of sequencing errors will increase the complexity of graph. Meanwhile, every process of sequencing always produce sequencing errors. This research aims to improve the performance of DNA sequencing error correction based on the spectral alignment by implementing a statistical approach. This approach generate a spectrum of solid tuple by choosing tuples that belong to the highest 75% of the tuple occurencies distribution. Reads containing sequencing errors are corrected using tuple references that belong to the solid tuple spectrum. Evaluation is conducted using Velvet, a DNA assembly software. The results show that our approach can reduce the complexity of graph produced by the previous approach.

Keywords: DNA sequencing error, spectral alignment, DNA sequence assembly, statistical approach

ABSTRAK

DNA *sequence assembly* berbasis graf biasanya digunakan untuk menghasilkan *contigs* dengan cara merangkai *reads* yang dihasilkan oleh mesin sekuensing generasi kedua. Namun demikian, DNA *sequence assembly* berbasis graf ini sensitif terhadap keberadaan *sequencing error*. Adanya *sequencing error* akan meningkatkan kompleksitas graf. Sementara itu, setiap proses sekuensing selalu menghasilkan *sequencing error*. Penelitian ini bertujuan untuk memperbaiki kinerja dari pengkoreksian DNA *sequencing error* berbasis *spectral alignment* dengan mengimplementasikan pendekatan statistika. Pendekatan ini menghasilkan spektrum dari *solid tuple* dengan memilih *tuple* yang termasuk ke dalam 75% tertinggi dari distribusi kemunculan *tuple*. *Reads* yang mengandung *sequencing error* dikoreksi dengan sekuens referensi yang termasuk di dalam spektrum *solid tuple* ini. Evaluasi dilakukan dengan menggunakan Velvet, sebuah perangkat lunak DNA assembly. Hasil evaluasi menunjukkan bahwa pendekatan ini dapat mereduksi kompleksitas graf yang dihasilkan oleh pendekatan sebelumnya.

Kata kunci: *DNA sequencing error, spectral alignment, DNA sequence assembly*, pendekatan statistika

PENDAHULUAN

Teknologi DNA *sequencing* saat ini telah mengalami banyak perkembangan, salah satunya teknologi DNA *sequencing* generasi kedua. Teknologi DNA *sequencing* generasi kedua dapat menghasilkan *reads* yang lebih banyak dan dengan biaya yang

lebih murah jika dibandingkan dengan teknologi Sanger *sequencing*, meskipun panjang reads yang dihasilkan jauh lebih pendek. Sebagai contoh, Illumina Genome Analyzer dapat menghasillkan 1.5 miliar *base-pairs* (bp) dengan panjang *read* 36, dalam satu kali eksekusi selama 60 jam [1].

Mayoritas teknologi DNA *assembly* yang sudah mapan saat ini dirancang agar dapat bekerja optimal ketika merakit *reads* hasil Sanger *sequencing*. Teknologi ini didesain untuk merangkai *reads-reads* berukuran panjang, tidak didesain untuk merangkai *reads* berukuran pendek [1]. Oleh karena itu diperkenalkanlah beberapa teknologi DNA *assembly* yang baru, salah satunya yang menggunakan pendekatan k-mer. Akan tetapi pendekatan ini tidak dapat merangkai bagian yang berulang (*repeat*) dengan akurat[1]. Untuk mengatasi masalah ini, dikembangkanlah metode DNA *sequence assembly* dengan pendekatan *Eulerian path* pada graf De-Bruijn [2].

DNA *sequence assembly* dengan pendekatan graf ternyata mampu mengatasi data *reads* yang pendek dan banyak, serta dapat mengatasi masalah *repeat*. Masalahnya adalah DNA *sequence assembly* dengan pendekatan graf sangat sensitif terhadap *error*. Padahal setiap proses *sequencing* pasti menghasilkan *error*. Jika data yang dimasukan mengandung *error*, graf yang dihasilkan dapat menjadi lebih kompleks dari yang seharusnya. Sehingga *error-error* tersebut harus diperbaiki sebelum dirangkai dengan DNA *assembler* berbasis graf. Oleh karena itu, penelitian ini dilakukan untuk membangun sistem yang dapat mendeteksi dan mengoreksi *error* pada sekuens DNA.

Salah satu penelitian yang terkait pengkoreksian DNA *sequencing* dilakukan oleh Caesar *et al* [3] dengan menerapkan metode *spectral alignment*. Pada penelitian tersebut, Caesar *et al* [3] menggunakan nilai *threshold* untuk menentukan *solid tuple* yang menjadi bagian dari spectrum. Pada penelitian ini, penentuan *solid tuple* dilakukan dengan memperhatikan peringkat frekuensi kemunculan *tuple*. Diharapkan perangkat lunak yang dihasilkan dapat mendeteksi dan mengoreksi *error* dengan lebih baik dibandingkan perangkat lunak yang dihasilkan pada penelitian sebelumnya.

METODE PENELITIAN

Secara garis besar, penelitian ini dibagi menjadi empat tahap, yaitu akuisisi data sekuens DNA, implementasi metode *spectral alignment* yang dimodifikasi, koreksi data sekuens DNA, dan evaluasi hasil koreksi.

Akuisisi data sekuens DNA

Data sekuens DNA yang digunakan pada penelitian ini diunduh dari *website* NCBI. Data tersebut merupakan data hasil proses sekuensing bermacam-macam makhluk hidup. Dari data tersebut, dipilihlah tiga organisme seperti yang tercantum dalam Tabel 1. Data ke-3 organisme itu selanjutnya akan diproses dengan menggunakan perangkat lunak MetaSim [4], untuk disimulasikan dan menghasilkan fragmen-fragmen yang mengandung *error*. Fragmen-fragmen yang mengandung *error* itulah yang akan menjadi masukan perangkat lunak yang akan dikembangkan pada penelitian kali ini.

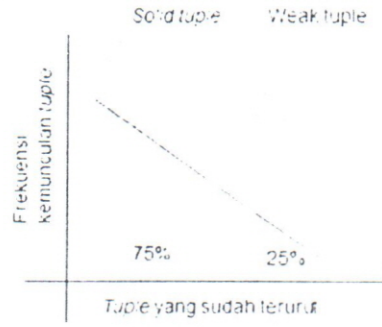
Tabel 1. Spesifikasi organisme yang digunakan pada penelitian

Organisme	Panjang sekuens Lengkap (bp)	Gi
Acetobacter pasteurianus IFO 3283-01 plasmid pAPA01-060	1815	258513334
Lactobacillus plantarun WCFS1 plasmid pWCFS101	1917	54307228
Staphylococcus aureus subsp. Aureus ED98 plasmid pAVY	1442	262225764

Implementasi Metode *Spectral Alignment* yang Dimodifikasi

Pada tahap ini, dibuat sebuah perangkat lunak yang mampu mengkoreksi *error* pada data sekuens DNA dengan menerapkan metode *spectral alignment* yang dimodifikasi. Penelitian ini menggunakan *library* SeqAn [5]. SeqAn dibangun menggunakan bahasa C++ berisi pustaka-pustaka yang mengimplementasikan algoritme dan struktur data yang efisien untuk mengelola dan memanipulasi data sekuens DNA.

Solid Tuple dan Weak Tuple



Gambar 1 Ilustrasi penentuan *solid tuple*

Setelah data masukan dibaca, ditentukan *tuple* yang termasuk *solid tuple* atau *weak tuple*. Untuk menentukan jenis *tuple* tadi, didefinisikan hal berikut. Diberikan himpunan data sekuen hasil pembacaan yang disebut $R = \{r_1, r_2, r_3, \ldots, r_k\}$, dan $|r_i| = L$, dengan $r_i \in \{A,C,G,T\}^l$ untuk seluruh i pada selang $1 \leq i \leq k$. Simbol A, C, G dan T merupakan kode nuklotida dengan A untuk adenin, C untuk sitosin, G untuk guanin dan T untuk timin. Pada penelitian yang dilakukan oleh Caesar *et al* [3], parameter yang digunakan untuk menentukan *solid tuple* adalah, panjang dari *tuple* (*l*) dan *multiplicity* (*m*). *Multiplicity* adalah jumlah kemunculan sebuah *tuple* pada himpunan R. Pada penelitian kali ini, parameter yang digunakan tidak hanya panjang (*l*) dan *multiplicity* (*m*) saja, pendekatan statistik juga akan digunakan dengan mengambil 75% tertinggi dan distribusi kemunculan tuple, seperti yang digambarkan pada Gambar 1. Sebagai contoh, R = {AAGC, AAAG, AACT, AGGT, AACC}. Sebuah himpunan *l-tuple* dengan panjang 2, disebut juga 2-*tuple* berisi {AA, AG, AC, GT}. jika menerapkan parameter di atas, maka yang merupakan *solid tuple* adalah {AA, AG, AC} karena *tuple* tersebut termasuk ke dalam himpunan *tuple* dengan nilai *multiplicity* 75% tertinggi dari seluruh *tuple*.

Penyusunan Himpunan Spectrum

Spectrum yang dinotasikan dengan $T_l(R)$ adalah himpunan seluruh *l-tuple* yang *solid* saja dari himpunan sekuens DNA R. Jadi *spectrum* didapat dari hasil perumusan *l-tuple* yang merupakan *solid tuple* yang telah diperoleh pada tahap sebelumnya. *Spectrum* yang dihasilkan pada tahap ini akan digunakan pada tahap selanjutnya [3] .

Deteksi dan Koreksi Error

Setelah *spectrum* $T_l(R)$ selesai dibentuk, langkah selanjutnya adalah menentukan *reads* mana saja yang merupakan anggota T-String dan bukan anggota T-String. T-String merupakan himpunan *reads* yang seluruh *substring*-nya ada didalam $T_l(R)$, dan *reads* yang bukan merupakan anggota T-String akan masuk kedalam himpunan F, yakni himpunan yang seluruh anggotanya merupakan sekuens DNA yang mengandung *error*. Setelah T-String dan himpunan F terbentuk, setiap anggota F akan dikoreksi satu demi satu, dengan membandingkannya dengan setiap anggota T-String dan menggantinya dengan anggota T-String yang paling mirip. Metode yang digunakan untuk mengukur kemiripan dua sekuens tersebut adalah dengan menggunakan jarak Levenshtein atau sering juga disebut sebagai *edit distance* [6].

Evaluasi

Evaluasi dilakukan dengan memasukan empat *data set* ke dalam perangkat lunak DNA *sequence assembler*. Empat *data set* tersebut meliputi, *data set* hasil koreksi dengan perangkat lunak penelitian ini, *data set* hasil koreksi dengan perangkat lunak penelitian sebelumnya, *data set* tanpa *error* dan *data set* dengan *error* yang tidak dikoreksi. Pada penelitian kali ini, *sequence assembler* yang digunakan adalah Velvet. Velvet adalah perangkat lunak yang terdiri atas algoritme-algoritme untuk memanipulasi graf De Bruijn dalam rangka melakukan DNA *sequence assembly* [7]. Setelah diperoleh graf dari ke empat *data set* tersebut, jumlah node yang dihasilkan oleh masing-masing *data set* akan dibandingkan satu sama lain. Jumlah node merepresentasikan kompleksitas graf yang terbentuk. Semakin banyak *error* pada data, semakin kompleks graf yang terbentuk. *Data set* tanpa *error* akan memiliki jumlah *node* paling sedikit dibandingkan *data set* yang memiliki *error* dan *data set* hasil koreksi.

HASIL DAN PEMBAHASAN

Simulasi Data Sekuens DNA dengan MetaSim

Untuk *data set* yang menjadi input perangkat lunak penelitian kali ini, simulasi dilakukan pada setiap organisme sebanyak satu kali. Simulasi dilakukan dengan perangkat lunak MetaSim. Simulasi dilakukan untuk menghasilkan data yang mengandung *error*. *Model error* yang digunakan adalah *Sollexa* dengan panjang per fragmen adalah 36. Pada *model error* yang digunakan, jenis *error* yang ditimbulkan dibatasi hanya pada *error* subtitusi saja. Konfigurasi parameter secara lengkap dapat dilihat pada Tabel 2.

Untuk *data set* yang akan menjadi *input* perangkat lunak penelitian sebelumnya, seluruh parameternya sama dengan parameter yang ada pada Tabel 2. Sedangkan untuk *data set* tanpa *error*, parameter yang digunakan juga sama seperti parameter pada Tabel 2, namun *Error model* yang digunakan adalah *Exact*, yang berarti tanpa *error*.

Tabel 2 Konfigurasi parameter untuk membangkitkan data sekuens DNA dengan MetaSim

Organisme	*Number of reads or mate pairs*	*Error model*	*DNA clone size distribution type*	*Mean*	*Second parameter*
Acetobacter pasteurianus	2000	*Empirical – Sollexa*	*Normal*	36	0
Lactobacillus plantarun	2000	*Empirical – Sollexa*	*Normal*	36	0
Staphylococcus aureus	1500	*Empirical - Sollexa*	*Normal*	36	0

Perangkat Lunak Pendeteksi dan Pengkoreksi *Error*

Perangkat lunak menerima *input* berupa fail FASTA (.fna) yang berisi fragmen-fragmen yang mengandung *error* hasil simulasi dengan MetaSim. Setelah isi fail dibaca dan disimpan ke dalam memori, sistem akan membangkitkan permutasi tanpa perulangan dari nukleotida A, C, G dan T. Panjang permutasi (*l*) yang digunakan adalah lima, sama seperti penelitian yang dilakukan Caesar *et al* [3]. Selanjutnya, kumpulan permutasi yang dihasilkan akan disebut sebagai *pool*.

Setelah *pool* terbentuk, selanjutnya akan dibentuk himpunan *spectrum*. Himpunan *spectrum* berisikan *tuple-tuple* yang termasuk ke dalam *solid tuple*. *Solid tuple* sendiri merupakan *tuple* yang memiliki panjang *l* dan termasuk 75% tertinggi dalam hal nilai *multiplicity*-nya.

```
Masukan: Himpunan pool, himpunan reads
Keluaran: Himpunan spectrum

set struct : tuple, number : Tuples
set spectrum
for i := 1...length pool
      set number = 0
      for j := 1...length reads
            if (pool[i] equal reads[j])
                  number++
            end if
      end for
      append pool[i] and number to Tuples[i]
end for
sort_tuple_by_number Tuples
limit = length Tuples ^0.75
for k :=1...limit
      append tuple.Tuples[k] to spectrum[k]
end for
return spectrum
```

Gambar 2 *Pseudocode* pembentukan *spectrum*

Selanjutnya akan dibentuk himpunan T-String dengan merujuk pada himpunan spektrum. Himpunan T-String adalah himpunan *reads* yang seluruh *substring*-nya terdapat pada himpunan spektrum, sehingga dapat juga dikatakan sebagai himpunan *reads* yang tidak mengandung *error*. *Reads-reads* yang tidak termasuk kedalam himpunan T-String akan masuk kedalam himpunan F, yakni himpunan *reads* yang mengandung *error*.

Setelah himpunan T-String dan F terbentuk, anggota himpunan F dikoreksi satu demi satu dengan membandingkan setiap anggotanya dengan setiap anggota himpunan T-String. Setiap anggota F akan dihitung nilai kedekatannya dengan setiap anggota T-String. Nilai kedekatan dihitung dengan menggunakan jarak Levensthein [6]. Setelah seluruh nilai kedekatan anggota T-String diketahui, dipilih anggota T-String dengan nilai tertinggi untuk menggantikan anggota himpunan F yang mengandung *error*.

Hasil keluaran perangkat lunak ini adalah sebuah fail FASTA (.fna) berisi fragmen-fragmen DNA hasil koreksi yang akan digunakan pada tahap evaluasi.

Perbandingan hasil koreksi perangkat lunak

Jumlah anggota *spectrum* yang digunakan pada penelitian ini lebih sedikit dibandingkan yang digunakan oleh Caesar *et al* [3], seperti yang ditunjukan pada Tabel 3. Karena jumlah anggota *spectrum* yang digunakan lebih sedikit, maka jumlah himpunan T-String yang dihasilkan menjadi lebih sedikit dan membuat jumlah *reads error* yang terdeteksi semakin banyak.

Tabel 3 Perbandingan hasil eksekusi perangkat lunak

Organisme	Metode	Jumlah *reads*	Jumlah anggota *spectrum*	Jumlah anggota T-String	Jumlah *reads* yang terdeteksi sebagai *error*
Acetobacter pasteurianus	Penelitian ini	2000	768	563	1437
	Caesar*et al* [3]	2000	895	1606	394
Lactobacillus plantarum	Penelitian ini	2000	768	531	1469
	Caesar *et al* [3]	2000	909	1573	427
Staphylococcus aureus	Penelitian ini	1500	768	709	791
	Caesar *et al* [3]	1500	811	1015	485

Waktu eksekusi yang dibutuhkan perangkat lunak penelitian ini lebih lama dibandingkan perangkat lunak Caesar *et al* [3], seperti yang ditunjukan pada Tabel 4. Hal ini dikarenakan jumlah *error* yang harus dikoreksi semakin banyak, sehingga waktu eksekusi menjadi lebih lama.

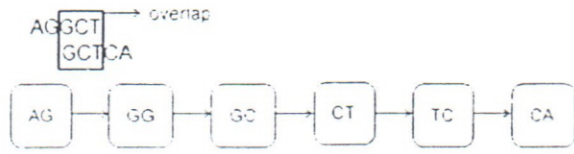
Tabel 4 Perbandingan waktu eksekusi perangkat lunak

Organisme	Metode	Waktu eksekusi (ms)
Acetobacter pasteurianus	Penelitian ini	524301
	Caesar *et al* [3]	461769
Lactobacillus plantarum	Penelitian ini	512051
	Caesar *et al* [3]	483340
Staphylococcus aureus	Penelitian ini	370102
	Caesar *et al* [3]	344651

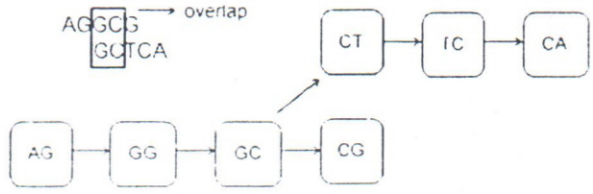
Evaluasi hasil koreksi perangkat lunak

Pada proses DNA *sequencing* dengan pendekatan graf, keberadaan *error* pada *sequence* DNA dapat membuat graf yang terbentuk bertambah kompleks. *Error* dapat menciptakan cabang baru pada graf yang seharusnya tidak ada, sehingga graf yang terbentuk lebih kompleks. Sehingga, kompleksitas sebuah graf dapat diukur dari jumlah *node* dan *edge* yang dihasilkan [3]. Sebagai contoh, diberikan dua sekuens DNA untuk

dibentuk graf, yaitu AGGCT dan GCTCA. Jika kedua fragmen tersebut dibentuk graf maka jumlah *node* dan *edge* yang terbentuk hanya enam dan lima, seperti pada Gambar 3. Akan tetapi jika terdapat satu saja kesalahan pembacaan, jumlah *node* dan *edge* akan bertambah satu seperti pada Gambar 4.



Gambar 3 Hasil konstruksi graf dengan fragmen tanpa *error*



Gambar 4 Hasil konstruksi graf dengan fragmen yang mengandung *error*

Evaluasi hasil koreksi dengan Velvet

Evaluasi dilakukan dengan menggunakan empat *data set*. Setiap *data set* terdiri dari tiga organisme, sehingga jumlah keseluruhan data yang digunakan ada dua belas. Empat *data set* terdiri dari, data yang mengandung *error*, data yang tidak mengandung *error*, data yang mengandung *error* dan telah dikoreksi dengan perangkat lunak penelitian kali ini dan data yang mengandung *error* dan telah dikoreksi dengan perangkat lunak penelitian sebelumnya. Data-data tersebut dievaluasi dengan perangkat lunak DNA *sequence assembler* yaitu Velvet. Keluaran dari velvet adalah graf De Bruijn yang merupakan hasil *assembly* dari data masukan.

Hasil keluaran Velvet disimpan kedalam dua fail, yaitu "PreGraph" dan "LastGraph". Fail "PreGraph" berisi daftar *node* dari graf yang dihasilkan dari data masukan. Sedangkan fail "LastGraph" berisi daftar *node* dari graf yang dihasilkan dari data masukan dan dilakukan koreksi data oleh velvet itu sendiri. Sehingga fail yang perlu kita perhatikan adalah "PreGraph".

Pada setiap menjalankan Velvet, kita diminta untuk memasukan nilai *hash length* atau *k*. Parameter *hash length* atau *k* adalah panjang k-mers yang dimasukkan kedalam *hash table*. Nilai *k* harus ganjil, lebih kecil dari MAXKMERHASH yaitu 31 bp, dan lebih kecil dari panjang per fragmen data. Nilai *k* yang digunakan pada penelitian kali ini adalah 17. Hasil proses DNA *sequence assembly* menggunakan Velvet dapat dilihat pada Tabel 4.

Tabel 5 Hasil DNA sequence assembly dengan Velvet

Organisme	Jumlah node			
	Tanpa koreksi	Caesar *et al* [3]	Hasil penelitian ini	Tanpa *error*
Acetobacter pasteurianus	129	105	45	8
Lactobacillus plantarum	109	72	44	7
Staphylococcus aureus	83	31	19	4

Pada Tabel 5 diatas, data set dengan label "Tanpa koreksi" berisi data sekuens dengan error, data set dengan label "Caesar et al [3]" berisi data sekuens yang dikoreksi dengan perangkat lunak dari penelitian sebelumnya, data set "Hasil penelitian ini" berisi data sekuens yang dikoreksi dengan perangkat lunak penelitian kali ini dan data set "Tanpa error" berisi data sekuens tanpa error. Data set yang dikoreksi dengan perangkat lunak penelitan kali ini menghasilkan graf yang lebih sederhana jika dibandingkan dengan penelitian sebelumnya, dilihat dari jumlah node yang lebih sedikit.

Hal ini dapat dicapai karena parameter penentuan solid tuple yang digunakan pada penelitian kali ini, lebih ketat dibandingkan dengan penelitian sebelumnya. Pada penelitian kali ini, jumlah solid tuple digunakan hanya 75% dari keseluruhan tuple. Sedangkan pada penelitian sebelumnya, jumlah solid tuple yang digunakan berkisar antara 82-91% dari keseluruhan tuple. Karena solid tuple yang digunakan lebih sedikit, maka jumlah anggota spectrum juga akan berkurang, yang akan berakibat berkurangnya jumlah anggota himpunan T-String. Jika jumlah anggota T-String lebih sedikit, maka jumlah fragmen yang dideteksi sebagai error akan semakin banyak, dan akhirnya hasil koreksi menjadi lebih maksimal.

KESIMPULAN DAN PROSPEK

Kesimpulan

Penelitian ini telah mampu menghasilkan perangkat lunak yang dapat mendeteksi dan mengoreksi *error* yang ada pada sekuens DNA. Kinerja keseluruhan yang dicapai lebih baik dibandingkan kinerja perangkat lunak penelitian sebelumnya, terlihat dengan lebih sederhananya graf yang dihasilkan dan lebih banyaknya jumlah error yang terdeteksi. Namun demikian, waktu eksekusi yang diperlukan oleh perangkat lunak ini lebih lama jika dibandingkan perangkat lunak yang dihasilkan oleh Caesar *et al* [3].

Prospek

Untuk penelitian selanjutnya, dapat digunakan pendekatan statistika yang lain dalam penentuan *solid tuple*, sehingga hasil koreksi yang didapat menjadi lebih baik. GPU *paralel processing* juga dapat diterapkan untuk mempersingkat waktu eksekusi yang dibutuhkan.

DAFTAR PUSTAKA

[1] Shi H, Schmidt B, Liu W, Wittig WM. 2009. Accelerating error correction in high-throughput short-read DNA sequencing data with CUDA. Di dalam: IEEE International Symposium on Parallel and Distributed Processing. hlm 1-8.

[2] Pevzner PA, Tang H, Waterman MS. 2001. An Euler path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. 98:9478-9753.
[3] Caesar N, Kusuma WA, Wijaya, SH. 2013. DNA sequencing error correction using spectral alignment Di dalam: International Conference on Advanced Computer Science and Information Systems (ICACSIS). hlm 279-284.
[4] Richter DC, Ott F, Auch AF, Schmid R, Huson DH. 2008. MetaSim A sequencing simulator for genomics and metagenomics. PLoS ONE 3(10).
[5] Döring A, Weese D, Rausch T, Reinert K. 2008. SeqAn an efficient, generic C++ library for sequence analysis. BMC Bioinformatics 9(1):11.
[6] Levenshtein VI. 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10:707.
[7] Zerbino DR, Birney W. 2008. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18(5):821-829.