Bioinformatics

Dungeom: A short DNA sequence de Bruijn graph-based assembler

Alexandros Karagiannopoulos[1]

[1]Department of Biology, Box 118, 221 00, Lund University, Sweden

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Genome assembly has been quite a challenge in genomic research due to the large amount of sequenced data that needs to be processed. De Bruijn graph-based algorithms, which extract the assembled sequence through the construction of a de Bruijn graph of the initial reads, are widely considered as the most efficient way of approaching the genome assembly process. Dungeom is based on the same principal and can reassemble a short DNA sequence after splitting it into smaller fragments (kmers).

1 INTRODUCTION

Even though NGS technologies revolutionized genomic research, the limitations of the sequencing instruments allowed for the creation of several downstream problems, such as sequence mapping, assembly, analysis and even data storage. The first programs used in genome assembly used to merge the generated sequence reads with an overlap-consensus-layout method, which was based on identifying and merging the reads with overlapping ends. This approach, though, could not handle effectively the long repetitive genomic regions found, especially, in eukaryotic genomes. This problem was merely solved with the introduction of de Bruijn graph-based algorithms and, as a result, many software packages that are typically used nowadays to assemble the underlying genomes and transcriptomes (Velvet, Spades, Trinity, etc.,) use de Bruijn graphs to reconstruct genomes from NGS libraries.

Although running those packages do not require any knowledge of de Bruijn graphs, such understanding is necessary to improve the quality of work in terms of minimizing errors in assembly and memory optimization. Thus, the design of a de Bruijn graph-based DNA sequence assembler DNA seemed like a reasonable step to that direction.

Dungeom, unlike similar software tools, is able to reassemble only a single DNA sequence of a maximum length of 1000bp and not numerous sequence reads, but uses the same two step workflow. In the first step, the DNA sequence is broken into small pieces (k-mers) and a de Bruijn graph is constructed from those short pieces. In the next step, the sequence is derived from the de Bruijn graph.

2 METHODS

Dungeom is written in Python (3.5.2) with Atom (1.19.2). Random DNA sequences for testing the program's efficacy were generated by RANDNA, a random DNA sequence generator [1] and the De Bruijn graph-based algorithm was inspired by Björn Canbäck's presentation on the subject.

3 FEATURES

Dungeon is a de Bruijn graph-based assembler that gets a short DNA sequence (max. 1000 bp) and the number of kmer size as inputs and outputs the de Bruijn graph of the sequence (in pdf format) and the reassembled sequence, reporting if it is identical or no to the input one. The sequence is first broken into small pieces (kmers) and the de Bruijn graph is constructed from the generated kmers. Then all kmers are shuffled in order to assure the randomness of the assembly except the first one, which will be the starting point of the reassembly. The process of the assembly is based on finding the Eulerian path from the graph, that is the trail in the de Bruijn graph which visits every edge exactly once. After the Eulerian walk is executed, the reassembled sequence is generated.

Table 1. Minimum kmer size which leads to successful reassembly of random DNA sequences of different sizes and GC content.

Size (bp)	GC content		
	0.4	0.6	0.8
100	6	7	7
200	7	8	8
500	8	8	11
1000	9	9	11

4 DISCUSSION

Dungeom is a de Bruijn graph-based assembler of short sequences. Although not a proper genome assembler, it can provide a useful insight into how the most widely used assemblers work. Consequently, it might be helpful to new bioinformaticians who would like a better conceptual understanding of the de Bruijn graph-based algorithm, as the Dungeom's script is quite simple and easy to follow.

ACKNOWLEDGEMENTS

Dungeom's name is a tribute to the room where the author received all his programming background, the Lund University's Bioinformatic Dungeon. The author would like to thank Björn Canbäck for the theoretical background he has provided on the subject, as well as for his useful contribution on the Dungeom's script.

REFERENCES

[1] Pica F. and Princip to G**.** (2006) RANDNA: a random DNA sequence generator. *In Silico Biol.*, **6**, 253-258.

[2] en.wikipedia.org/wiki/Velvet_assembler

[3] homolog.us/Tutorials/