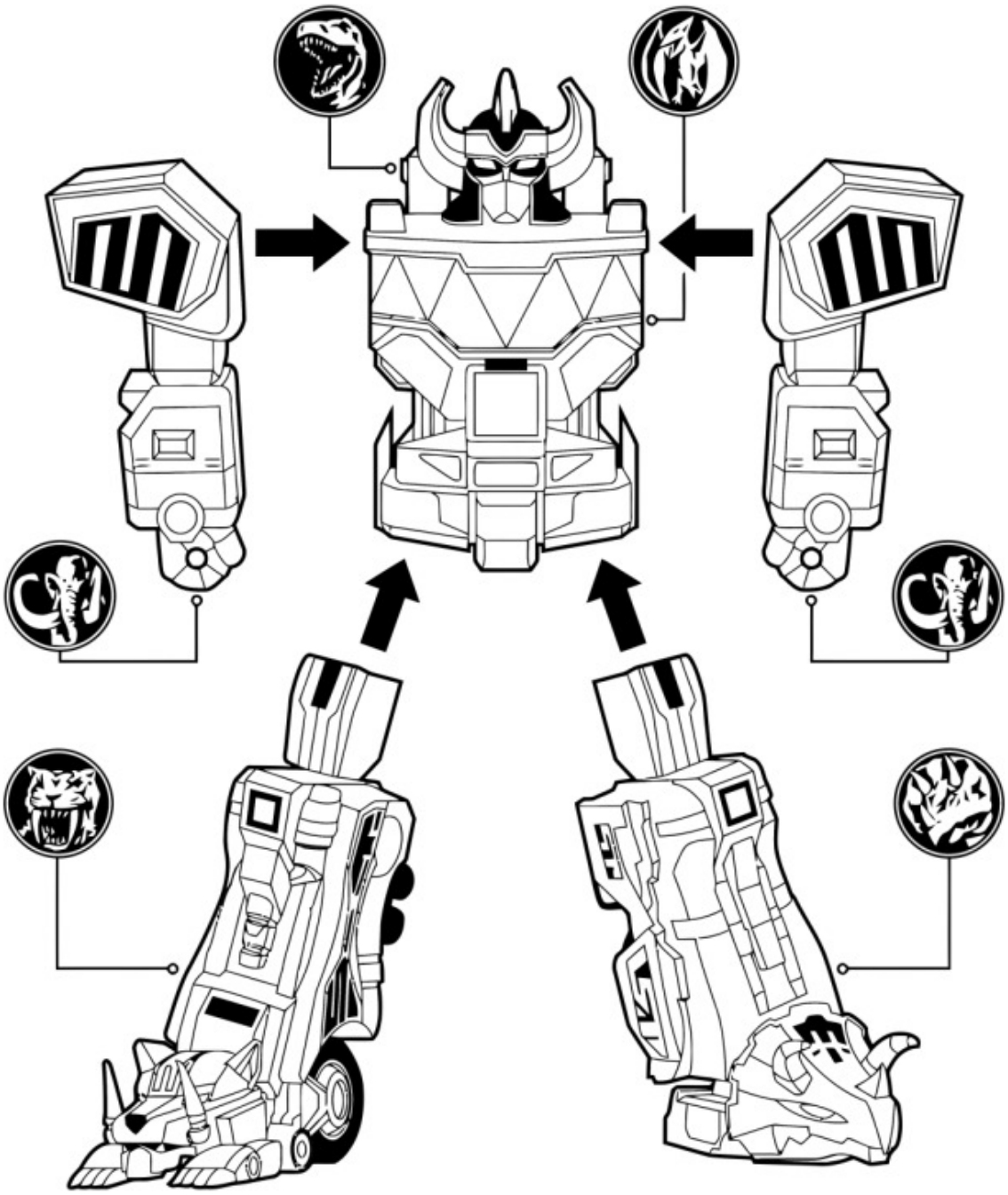
# Assembly Project

3 avril 2023



# 1 Main Goals

The goal of this project is three-fold :

- Write an assembler based on a de Bruijn graph
- Assemble several datasets highlighting various challenges
- Evaluate the produced assemblies' quality

The following points precisely describe what is expected and suggest a possible schedule.

# 2 Available Data

All sequencing data and reference sequences will be gradually uploaded here :
https://nextcloud.univ-lille.fr/index.php/s/Ao8kz6iXkA83mJR

# 3 $K$-mer index

First, you will need a structure able to index a dataset's $k$-mers. You are free to use any existing structure. Here are some suggestions based on your previous works.

- Bloom Filter
- Dictionary / Hash table
- Suffix tree/ Suffix array

Your index should be able to support the following operations :

- $k$-mer insertion
- $k$-mer presence

## 3.1 Milestone 1

Once your index is working, you should be able to insert all $k$-mers of an input Fasta file inside it.

# 4 Graph traversal

Using your $k$-mer index, you can now navigate the de Bruijn graph to produce an assembly.
Possible steps/features :

- List possible successors (respectively, predecessors) of a $k$-mer
- Construct a simple path from a $k$-mer
- Produce a simple assembly without duplicated $k$-mers in the output

## 4.1 Milestone 2

You should be able to reconstruct the first reference using sequencing data without errors.

# 5 $K$-mer filter

To handle sequencing errors, you may want to filter $k$-mers with low abundance (their number of occurrences in the input dataset).
Once again, you are free to use any existing structure. Here are some suggestions based on your previous works :

- Bloom FilterS
- Dictionary / Hash table
- Suffix tree/ Suffix array

### 5.1 Milestone 3

You should be able to reconstruct (partially) the first reference using sequencing data with sequencing errors.

## 6 Assembly evaluation

### 6.1 Reference based evaluation

To assess your assembly quality, you can compare it to the reference using the QUAST tool. It can be found here http://quast.sourceforge.net/quast.html. Note that you can also use the online version http://cab.cc.spbu.ru/quast/.

### 6.2 De novo evaluation

Alternatively, you can use the KAT tool to perform this analysis without using the reference genome. KAT can be found here https://kat.readthedocs.io/en/latest/installation.html. The kmer spectrum analysis can be performed following this guide https://kat.readthedocs.io/en/latest/walkthrough.html#genome-assembly-analysis-using-k-mer-spectra.

### 6.3 Milestone 4

You should be able to get an idea of how much your contigs are similar to the reference genome.

## 7 Graph cleaning

To handle complex sequencing errors, you can apply some modifications to your De Bruijn graph.

- Tip removal
- Bubble removal
- Complex pattern

### 7.1 Milestone 5

You should be able to obtain large (almost) error-free contigs.

## 8 Parameters exploration

You can now assemble your genome using several Kmer sizes or filtering thresholds and evaluate the obtained contigs.

### 8.1 Milestone 6

You should be able to assemble ”perfectly” the reference genome.

## 9 Play !

Now you can try to assemble more complex genomes. Try to assemble Medium.fa and Hard.fa.
If you can guess the reference genome used for those two files, you can compare your assembly to it !