Metagenome Assembly with IDBA

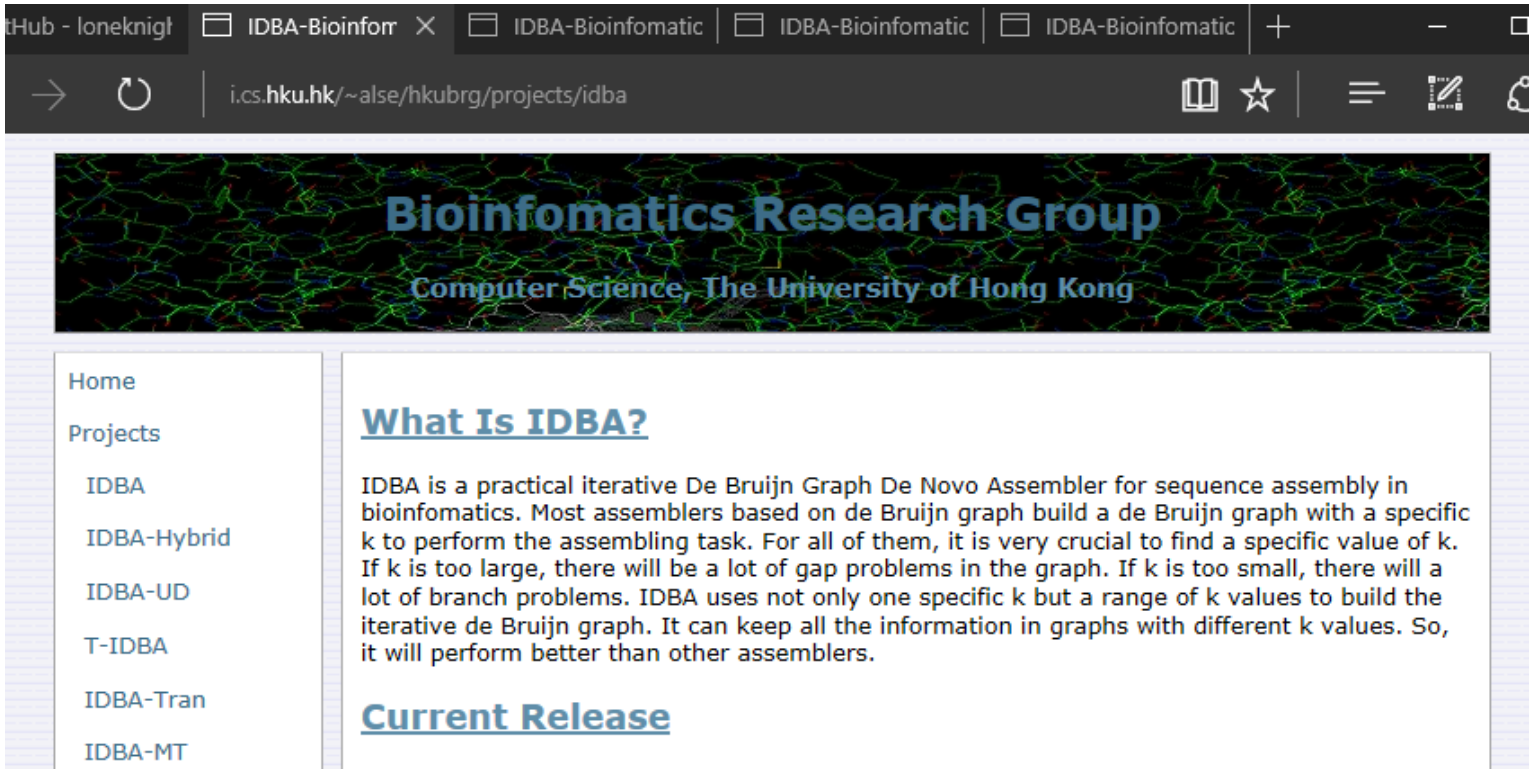
Feiyang Xue

What is Metagenome Assembly

- Assembly sequences into scaffolds

- De Novo Assembly
  - Without reference
- Reference Assembly
  - With reference

What is IDBA


IDBA-Bioinfom
IDBA-Bioinfomatic
IDBA-Bioinfomatic
IDBA-Bioinfomatic
i.cs.hku.hk/~alse/hkubrg/projects/idba
Bioinfomatics Research Group
Computer Science, The University of Hong Kong
Home
Projects
IDBA
IDBA-Hybrid
IDBA-UD
T-IDBA
IDBA-Tran
IDBA-MT
What Is IDBA?
IDBA is a practical iterative De Bruijn Graph De Novo Assembler for sequence assembly in bioinfomatics. Most assemblers based on de Bruijn graph build a de Bruijn graph with a specific k to perform the assembling task. For all of them, it is very crucial to find a specific value of k. If k is too large, there will be a lot of gap problems in the graph. If k is too small, there will a lot of branch problems. IDBA uses not only one specific k but a range of k values to build the iterative de Bruijn graph. It can keep all the information in graphs with different k values. So, it will perform better than other assemblers.
Current Release

IDBA-UD vs IDBA-Hybrid

- IDBA-UD
  - for Short Reads Sequencing data with Highly **Uneven** Sequencing **Depth**.
- IDBA-Hybrid
  - It is an extension of IDBA-UD algorithm. It aims at using a closed related reference genome to help de novo assembly, especially when sequencing depth is low.

- IDBA, IDBA-UD, IDBA-Hybrid and IDBA-Tran all in one package Released Oct 18, 2012

Using IDBA on Proteus

- Data in FASTA format
- Load idba with “module”
  - “module load idba”
- Call “idba” or “idba_hybrid”

```
fx28@proteusa01:~
[fx28@proteusa01 ~]$ module load idba
[fx28@proteusa01 ~]$ idba
not enough parameters
IDBA-Hybrid - Iterative de Bruijn Graph Assembler for hybrid sequencing data.
Usage: idba_hybrid -r read.fa -o output_dir [--reference ref.fa]
Allowed Options:
  -o, --out arg (=out)                  output directory
  -r, --read arg                        fasta read file (<=128)
      --read_level_2 arg                paired-end reads fasta for second level scaffolds
      --read_level_3 arg                paired-end reads fasta for third level scaffolds
      --read_level_4 arg                paired-end reads fasta for fourth level scaffolds
      --read_level_5 arg                paired-end reads fasta for fifth level scaffolds
  -l, --long_read arg                   fasta long read file (>128)
      --reference arg                   reference genome
      --mink arg (=20)                  minimum k value (<=124)
      --maxk arg (=100)                 maximum k value (<=124)
      --step arg (=20)                  increment of k-mer of each iteration
      --inner_mink arg (=10)            inner minimum k value
      --inner_step arg (=5)             inner increment of k-mer
      --prefix arg (=3)                 prefix length used to build sub k-mer table
      --min_count arg (=2)              minimum multiplicity for filtering k-mer when building the grap
h
      --min_support arg (=1)            minimum supoort in each iteration
      --num_threads arg (=0)            number of threads
      --seed_kmer arg (=30)             seed kmer size for alignment
      --min_contig arg (=200)           minimum size of contig
      --min_region arg (=500)           minimum size of region in reference genome
      --similar arg (=0.95)             similarity for alignment
      --max_mismatch arg (=3)           max mismatch of error correction
      --min_pairs arg (=3)              minimum number of pairs
      --max_gap arg (=50)               maximum gap in reference
      --no_local                        do not use local assembly
      --no_coverage                     do not iterate on coverage
      --no_correct                      do not do correction
      --pre_correction                  perform pre-correction before assembly

[fx28@proteusa01 ~]$
```

Using IDBA somewhere Else?

- Source can be obtained and built
- Tested building on Ubuntu machine in Bossone 615 Lab
- Tested building on Proteus
- Root/admin is not necessary
- https://github.com/loneknightpy/idba
- (may need to build autoconf and automake before IDBA)

idba_install.sh

```
#!/bin/sh

# set variable and switch path (assume ~/.local/bin is in PATH)
home_local=$HOME/.local/
cd $home_local

# download all the files
wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.gz
wget http://ftp.gnu.org/gnu/automake/automake-1.15.tar.gz
git clone https://github.com/loneknightpy/idba

# extract files
tar -xf autoconf-2.69.tar.gz
tar -xf automake-1.15.tar.gz

# build autoconf
cd autoconf-2.69
./configure --prefix=$home_local
make -j4
make install
cd ..

# build automake
cd automake-1.15
./configure --prefix=$home_local
make -j4
make install
cd ..

# build idba
cd idba
aclocal
autoconf
automake --add-missing
./configure --prefix=$home_local
make -j4
make install
cd ..
```

Sample job script with IDBA-UD

- Takes a long time
- Eats a long memory

fx28@proteusa01:~/genomics_tutorial_5

```
#$ -S /bin/bash
### execute the job from the current working directory, i.e. the directory in which the qsub command is
given
#$ -cwd
### join both stdout and stderr into the same file
#$ -j y
### set email address for sending job status
#$ -M fx@drexel.edu
### project - basically, your research group name with "Grp" replaced by "Prj"
# -P rosenclassPrj

#$ -l vendor=amd
#$ -q all.q

### select parallel environment, and number of job slots
#$ -pe shm 8
### request 8 hrs of wall clock time "h_rt" = "hard real time" (format is HH:MM:SS, or integer seconds)
#$ -l h_rt=12:00:00
### a hard limit 16 GB of memory per slot - if the job grows beyond this, the job is killed
#$ -l h_vmem=16G
### want nodes with at least 16 GB of free memory per slot
#$ -l m_mem_free=16G

. /etc/profile.d/modules.sh

### These four modules must ALWAYS be loaded
module load shared
module load proteus
module load sge/univa
module load gcc

module load idba
cd /home/fx28/genomics_tutorial_5
idba -l CSJP002A_R.fasta -o CSJP002A_R_idba_ud_out/ --min_contig 80 --mink 20 --maxk 60 --step 20
idba -l CSJP002B_R.fasta -o CSJP002B_R_idba_ud_out/ --min_contig 80 --mink 20 --maxk 60 --step 20
idba -l CSJP002C_R.fasta -o CSJP002C_R_idba_ud_out/ --min_contig 80 --mink 20 --maxk 60 --step 20
                                                                                    39,0-1        75%
```

Sample (Incomplete) Output with IDBA-UD

```
[fx28@proteusa01 CSJP002A_R_idba_ud_out_1st_run]$ ls -lh
total 30G
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 align-20
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 align-40
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 align-60
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:41 begin
-rw-r--r-- 1 fx28 rosenclassGrp 1.4G Feb 14 05:44 contig-20.fa
-rw-r--r-- 1 fx28 rosenclassGrp 587M Feb 14 05:44 contig-40.fa
-rw-r--r-- 1 fx28 rosenclassGrp 533M Feb 14 05:44 contig-60.fa
-rw-r--r-- 1 fx28 rosenclassGrp 1.9G Feb 14 05:44 graph-20.fa
-rw-r--r-- 1 fx28 rosenclassGrp 640M Feb 14 05:44 graph-40.fa
-rw-r--r-- 1 fx28 rosenclassGrp 558M Feb 14 05:44 graph-60.fa
-rw-r--r-- 1 fx28 rosenclassGrp  24G Feb 14 05:44 kmer
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 local-contig-20.fa
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 local-contig-40.fa
-rw-r--r-- 1 fx28 rosenclassGrp    0 Feb 14 05:44 local-contig-60.fa
-rw-r--r-- 1 fx28 rosenclassGrp  819 Feb 14 05:41 log
```

contig-20.fa

- Sample segmenets
  - `>contig-20_0 length_5205 read_count_0`
  - `AGGCAGCA.....AATTGGCA`
  - `.....`
  - `>contig-20_16764383 length_20 read_count_0`
  - `AGCACCAGCGCTGCTGCCAG`