[cd-hits][https://github.com/weizhongli/cdhit/wiki/3.-User's-Guide]

cd-hit根据设定的相似度阈值将蛋白聚类，常用于序列的识别。每个cluster都具有一个代表性序列。输入蛋白数据为fasta格式，输出两个文件,代表性序列的fasta文件，文本文件列出clusters列表

```
cd-hit -i db -o db100 -c 1.00 -n 5 -M 16000 -d 0 -T 8
```

db为输入文件名称，db100为输出文件名称

-c 1.0 表示100%一致性，clustering阈值; 0.9 表示90%一致性, clustering阈值

-c 默认为0.9，为cd-hit整体序列一致性，为比对中一致性氨基酸总数除以短的氨基酸序列的总长

-n 5 表示使用word长度,默认为5

-s 长度差异阈值,默认为0.0，如果设置为0.9，那么比对中短的序列长度至少得为cluster中代表性序列长度的90%

-S 氨基酸序列的长度差异，默认为999999，若设置为60，那么比对中短的序列不能比代表性序列长度短60以上

-aL 长序列的比对覆盖度,默认为0, 若设置为0.9，则比对必须覆盖长序列的90%以上

-AL 长序列的比对覆盖度,默认为9999999, 若设置为60，针对比对长度为400的序列，那么比对必须满足大于等于340

-aS 短序列的比对覆盖度，默认为0，若设置为0.9，比对须满足覆盖90%该短序列

-AS 短序列必读覆盖度，默认为999999，若设置为60，针对比对长度为400的序列，那么比对长度须满足大于等于340

-A 针对两序列的最小比对覆盖度

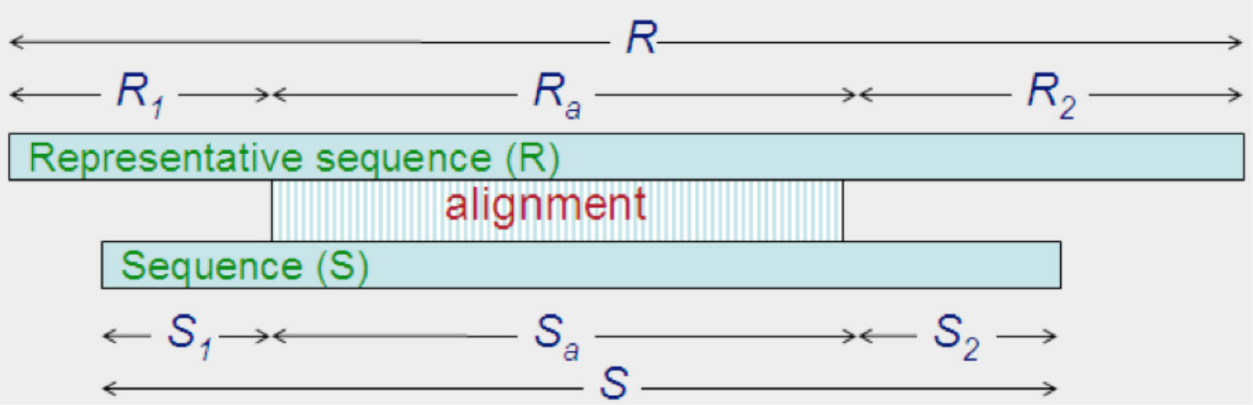
-d 0 在.clstr输出文件中使用fasta头的直到空格位置的序列名称,默认为20

-M 16000 使用16GB RAM

-T 8 使用8线程

```
Choose of word size:
-n 5 for thresholds 0.7 ~ 1.0
-n 4 for thresholds 0.6 ~ 0.7
-n 3 for thresholds 0.5 ~ 0.6
-n 2 for thresholds 0.4 ~ 0.5
```

针对不同的word长度选择对应的阈值



对应的aL/AL/aS/AS/s/S/U/uL/uS：

```
aL = Ra / R
AL = R - Ra
aS = Sa / S
AS = S - Sa
s = Sa / Ra
S = R / S
U = S1 + S2
uL = U / R
uS = U / S
```

输出.clstr：

- > 表示一个新的cluser
- * 表示该序列为该cluster的代表性序列
- % 表示该序列和代表性序列相同

---

cd-hit-2d

CD-HIT-2D意味着比较两蛋白数据集(db1,db2)。识别db2中和db1中序列相似的序列。输入两蛋白序列集(db1, db2)为fasta格式，输出两文件：一个fasta格式的db2中不同于db1的蛋白序列，一个text文件列出db1和db2中相似的序列。

```
cd-hit-2d -i db1 -i2 db2 -o db2novel -c 0.9 -n 5 -d 0 -M 1600 -T 8
```

这里db1和db2位输入fasta格式蛋白序列文件, db2novel为输出文件, 0.9表示90%的一致性，为比较阈值, 5表示word长度

-c 1.0 表示100%一致性，clustering阈值; 0.9 表示90%一致性, clustering阈值

-b 比对的band_width，默认为20

-n 5 表示使用word长度,默认为5

-l 舍弃序列长度阈值，默认为10

-t tolerance for redundance，默认为2

-d 0 在.clstr输出文件中使用fasta头的直到空格位置的序列名称,默认为20

-s 长度差异,默认为0，若设置为0.9，则短序列需要至少为代表性长序列的90%

-S 氨基酸长度差异阈值，默认999999，若设置为60，短序列相差代表性长序列至多为60

-s2 长度差异阈值，默认为1.0，同一个cluster中，db1中序列需要大于或等于db2中序列长度；若设置为0.9，同一个cluster中，db1中序列长度可以大于等于db2中序列长度的90%

-S2 长度差异阈值，默认为0，同一个cluster中，db1中序列须大于等于db2中序列；若设置为60，那么db2中序列可大于db1中序列60aa

因此既可以设置-s2或交换db1和db2的输入顺序，来取消默认条件下，db2中序列不能长于db1的情况：

```
cd-hit-2d -i db1 -i2 db2 -o db2novel -c 0.9 -n 5 -d 0 -M 1600 -T 8 -s2 0.9
```

```
cd-hit-2d -i db2 -i2 db1 -o db1novel -c 0.9 -n 5 -d 0 -M 1600 -T 8
```

word长度选择同cd-hit

Incremental clustering

It is easy to make incremental update with cd-hit /cd-hit-2d. For example:

```
nr is the nr database of last month
month is the new sequences of nr of this month
```

In last month, you ran:

```
cd-hit -i nr -o nr90 -c 0.9 -n 5
```

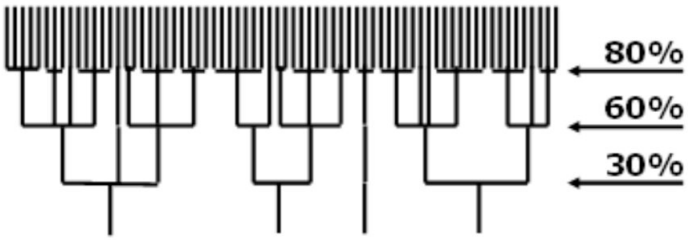
This month, you can run incremental clustering

```
cd-hit-2d -i nr90 -i2 month -o month-new -c 0.9 -n 5
cd-hit -i month-new -o month90 -c 0.9 -n 5
cat month90 >> nr90
clstr_merge.pl nr90.clstr month-new.clstr > temp.clstr
cat temp.clstr month90.clstr > this_month_nr90.clstr
```

This approach is much faster than runing from scratch. It also preserves stable cluster structure.

Hierarchically clustering

With multiple-step, iterated runs of CD-HIT, you perform a clustering in a neighbor-joining method, which generates a hierarchical structure.



Commands:

```
cd-hit -i nr -o nr80 -c 0.8 -n 5
cd-hit -i nr80 -o nr60 -c 0.6 -n 4
psi-cd-hit.pl -i nr60 -o nr30 -c 0.3
```

This way is faster than one-step run from nr directly to nr30. It can also helps correct errors by one-step clustering (see last paragraph in algorithm limitation section).