GENE TECHNOLOGY

Session 3. **Cutting and ligating DNA**

- Knocking gene down
- Amplifying DNA

Genetic (Gene) Engineering impacted on medicine and biotechnology

DNA Amplification

Sequencing

Gene Therapy

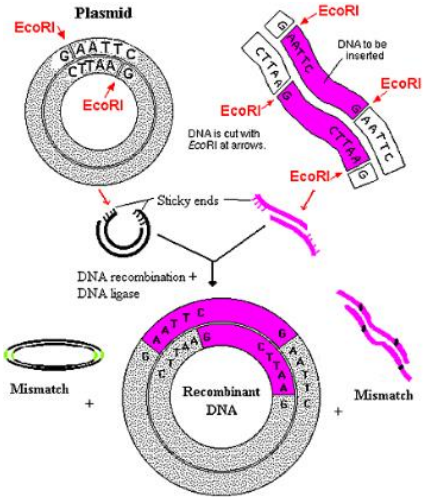
GMO

- Resistance to insects
- Crop yield

The specific DNA cloning (cutting and ligation) is an essential technology in genetic engineering.

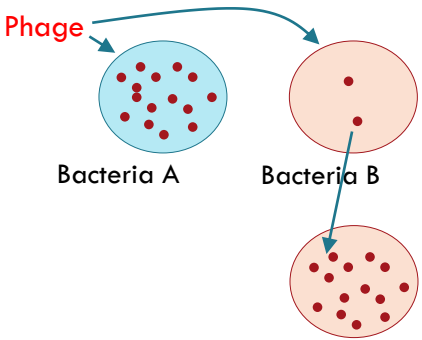
- Before PCR developed, how were biologists able to amplify DNA in large amount?

Cloning the DNA, transfecting it to bacteria, growing and harvesting them



Cutting and ligation for cloning

- DNA Cutting
  - Restriction enzyme
  - Sequence-specific manner
  - First evidence: 1952

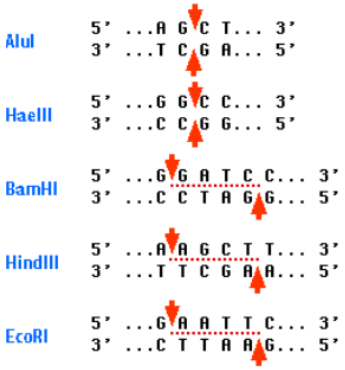


→ This phenomena was later explained with a model "restriction-modification".

- Late 1960s – **Type I** random position far away from a recognition site.
- 1970 – **Type II** HindIII (Smith & Wilcox) sequence-specific and at specific position.

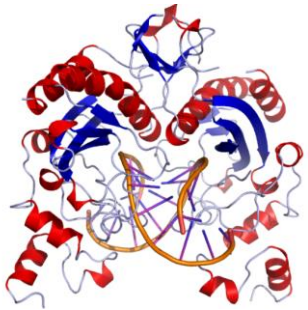
→ **1978 Nobel prize**

Restriction enzymes and recognition sites



PpuMI	5' RGGWCCY 3' 3' YCCWGGR 5'	5' RG GWCCY 3' 3' YCCWG GR 5'

- Sites are palindromic (inverted repeat)



- Blunt end vs Sticky end
- 5' single-stranded vs 3' single-stranded end

➔ REBASE

Restriction enzymes and recognition sites

- In general, the length of site is 4~8 nt.

  Length of recognition provides the specificity

  AluI AGCT → 1/256

  BamHI GGATCC → 1/4096

  *Not*I → 1/65,536

- Some enzymes are less specific.

  BxtX2I RGATCY → 1/1024

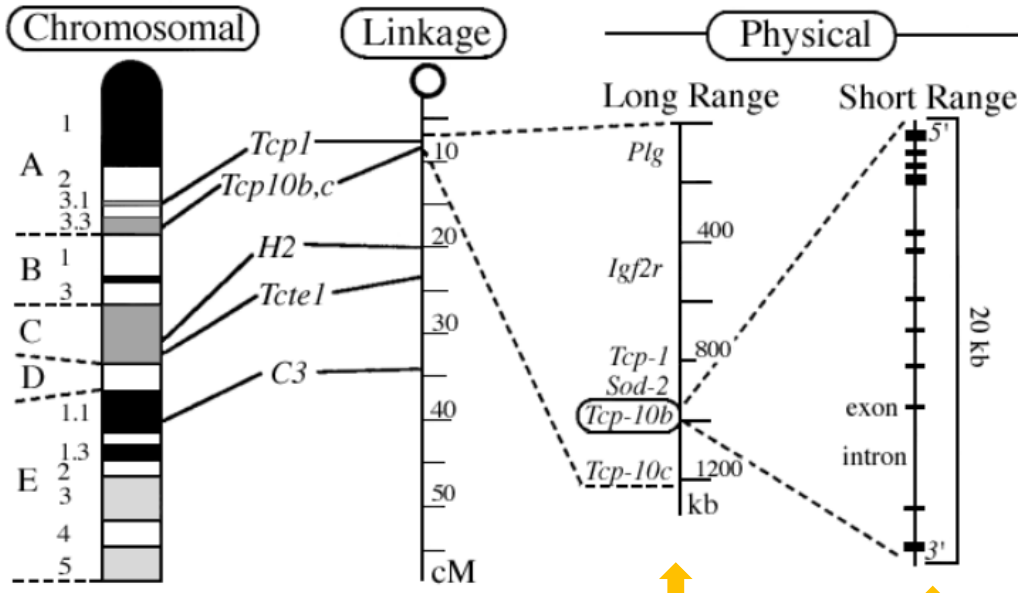
  R = A or G, Y = C or T, W = A or T

- **Type III**: cleave at sites a short distance from recognition site

IUPAC nucleotide code	Base
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
. or -	gap

*Eco*P15I	*Escherichia coli*	5'CAGCAGN_{25}NN 3'GTCGTCN_{25}NN	5'---CAGCAGN_{25} NN---3' 3'---GTCGTCN_{25}NN ---5'

Genetic and physical chromosome map



Finding restriction sites

```
import re

enzymes = {
    'BclI':    'TGATCA',
    'BfmI':    'CTRYAG',
    'Cac8I':   'GCNNGC',
    'EcoRI':   'GAATTC',
    'HindIII': 'AAGCTT',
    }

enzymes_mod = enzymes.copy()

amb = {
    'R': '[AG]',
    'Y': '[CT]',
    'N': '[AGCT]',
    'W': '[AT]',
    'M': '[AC]',
    'S': '[CG]',
    'K': '[TG]',
    'V': '[ACG]',
    'H': '[ACT]',
    'D': '[AGT]',
    'B': '[CGT]',
    }

for key in enzymes_mod.keys():
    for ambkey in amb.keys():
        enzymes_mod[key] = enzymes_mod[key].replace(ambkey, amb[ambkey])

seq = 'GATCTGACTAGCGAGCGTGATCAAGCTTGTGTAGGAATTCCTTGATGCTGTAGCGCGAGCTGA'

for i in range(0, len(seq) - 6):
    testseq = seq[i:i + 6]
    for key in enzymes_mod.keys():
        if re.search(enzymes_mod[key], testseq):
            pos = i + 1
            print key, '\t', pos, '\t', testseq, '\t', enzymes[key]
```

Pattern matching

If statement

```
if 1==1: print "true"
else: print "false"
```

- **Regular expression** is very useful in pattern matching.
- [CT]: C or T
- [CT][AG]: C or T and A or G
- [AB*]: A or AB or ABB or ABBB, …
- [AB+]: AB or ABB or ABBB,…
- [AB?]: A or AB
- A{6}: AAAAAA
- A{4,6}: AAAA, AAAAA, AAAAAA

More..

```
dna = "AACGGAATTCCCTCTC"
if dna.find("GAATTC")>=0: print "match"
else: print "mismatch"
if dna.find("GAA[CT]TC")>=0: print "match"
else: print "mismatch"
```

match
mismatch

→ Regular expression module is required

Pattern matching

- **Regular expression** are very useful in pattern matching.

```
import re
dna = "AACGGAATTCCCTCTC"
if re.search('GAA[CT]TC', dna): print "match"
else: print "mismatch"
if re.match('GAA[CT]TC', dna): print "match"
else: print "mismatch"
if re.match('GAA[CT]TC', dna[4:10]): print "match"
else: print "mismatch"
```

match
mismatch
match

Pattern matching

```
rna = "AACGGAAUUCCCUCUC"
if re.search('C{3}', rna): print "match"
else: print "mismatch"
if re.search('U*C*T', rna): print "match"
else: print "mismatch"
if re.search('U*C*U', rna): print "match"
else: print "mismatch"
```

match
mismatch
match

Character/String replacement

Dictionary usage

```
enzymes = {
    'BclI':    'TGATCA',
    'BfmI':    'CTRYAG',
    'Cac8I':   'GCNNGC',
    'EcoRI':   'GAATTC',
    'HindIII': 'AAGCTT',
    }

print enzymes.keys()
print enzymes.values()
print enzymes.items()
```

```
['HindIII', 'BfmI', 'BclI', 'EcoRI', 'Cac8I']
['AAGCTT', 'CTRYAG', 'TGATCA', 'GAATTC', 'GCNNGC']
[('HindIII', 'AAGCTT'), ('BfmI', 'CTRYAG'), ('BclI', 'TGATCA'), ('EcoRI', 'GAATTC'), ('Cac8I',
'GCNNGC')]
```

code2.1 cut.py

```
#!/usr/bin/python
import re

enzymes = {
    'BclI':    'TGATCA',
    'BfmI':    'CTRYAG',
    'Cac8I':   'GCNNGC',
    'EcoRI':   'GAATTC',
    'HindIII': 'AAGCTT',
    }

enzymes_mod = enzymes.copy()

amb = {
    'R': '[AG]','Y': '[CT]','N': '[AGCT]','W': '[AT]',
    'M': '[AC]','S': '[CG]','K': '[TG]','V': '[ACG]',
    'H': '[ACT]','D': '[AGT]','B': '[CGT]',
    }

for key in enzymes_mod.keys():
    for ambkey in amb.keys():
        enzymes_mod[key] = enzymes_mod[key].replace(ambkey, amb[ambkey])

seq = 'GATCTGACTAGCGAGCGTGATCAAGCTTGTGTAGGAATTCCTTGATGCTGTAGCGCGAGCTGA'

for i in range(0, len(seq) - 6):
    testseq = seq[i:i + 6]
    for key in enzymes_mod.keys():
        if re.search(enzymes_mod[key], testseq):
            pos = i + 1
            print key, '\t', pos, '\t', testseq, '\t', enzymes[key]
```

```
Cac8I      11      GCGAGC      GCNNGC
BclI       18      TGATCA      TGATCA
HindIII    23      AAGCTT      AAGCTT
EcoRI      35      GAATTC      GAATTC
BfmI       48      CTGTAG      CTRYAG
Cac8I      55      GCGAGC      GCNNGC
```