ゲノムデータベースとプログラミング

法政大学 生命科学部　大島研郎

本日の講義資料

kiso2

blast.pl

result.txt

本日の講義で使用するWebページへのリンクが載せてあります.

本日の講義では，Active perl を使います.

- コマンドプロンプトを立ち上げてください

スタート → Windowsシステムツール → コマンドプロンプト

```
> perl -v
```

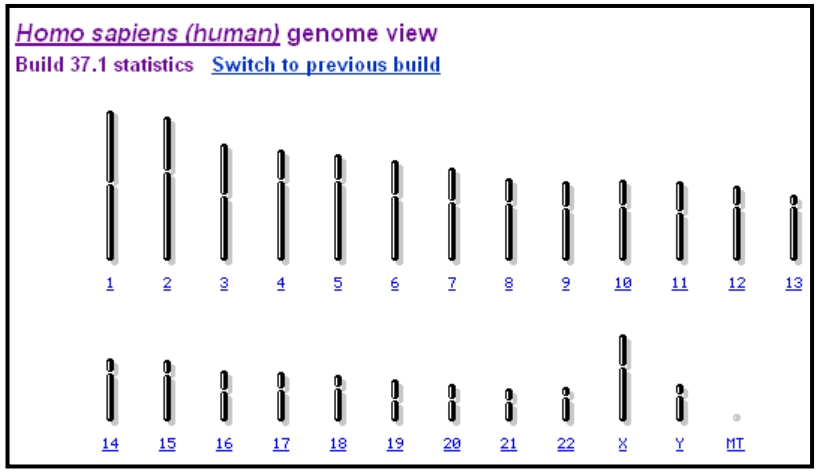
と入力して，エラーが出ないことを確認してください

ゲノムとは

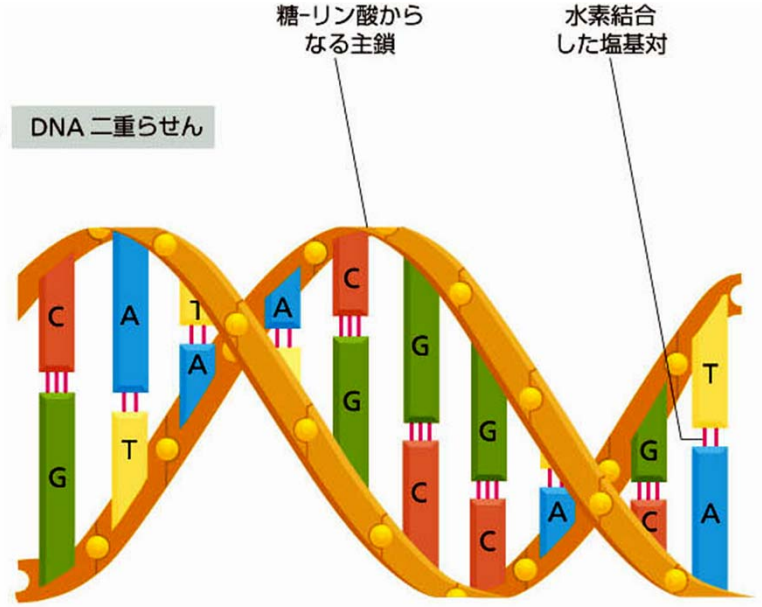
gene（遺伝子）+ -ome（総体）

ゲノム = ある生物のもつ全ての遺伝情報

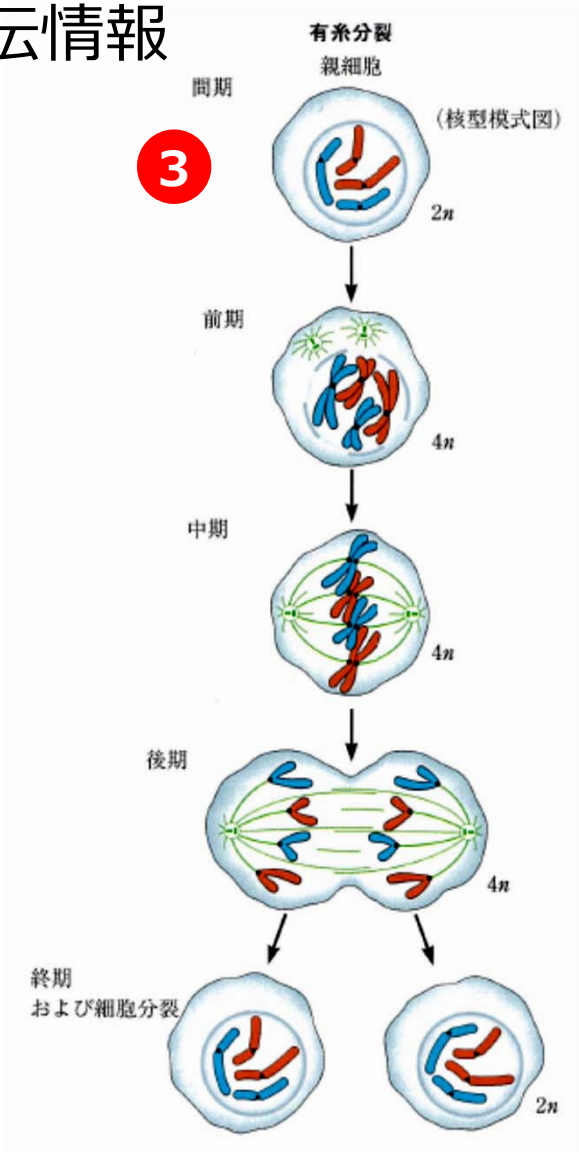
1



2



3



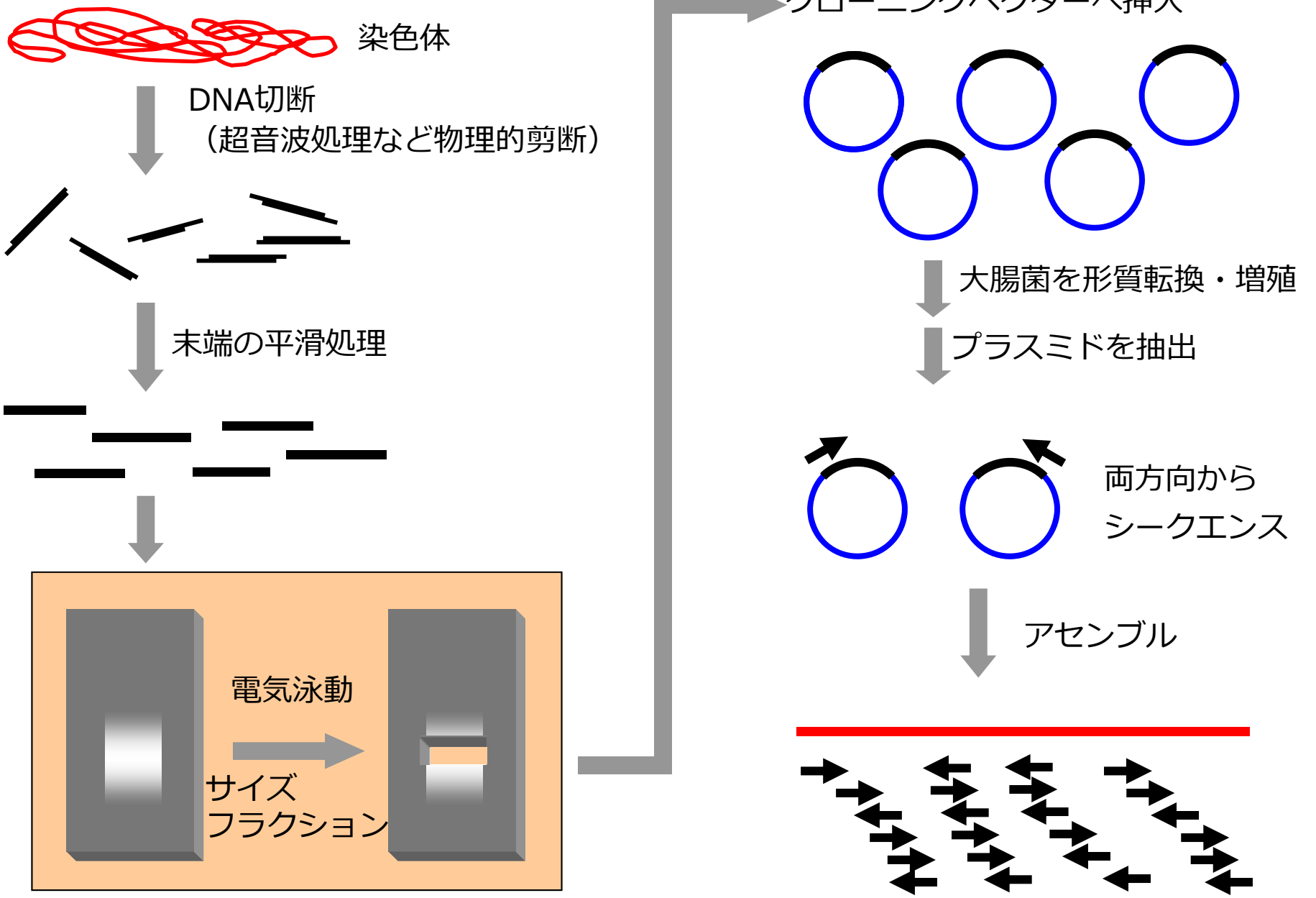
ゲノム解読の歴史

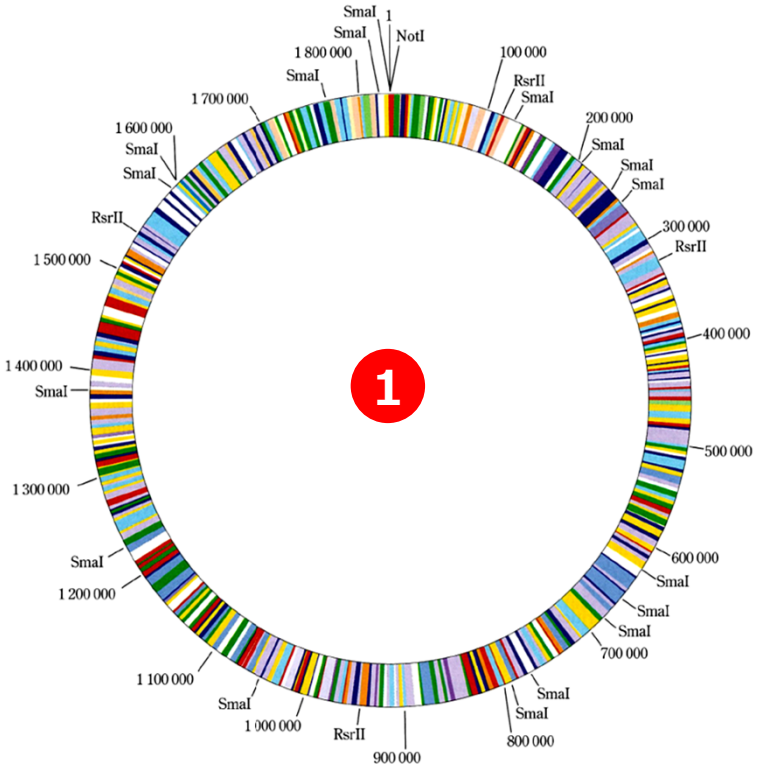
表 1–1 配列が完全に決定されたゲノムの例

生物種	特　徴	生息場所	ゲノムサイズ（一倍体ゲノムあたりの塩基対数，×1000） ①	タンパク質指令遺伝子の数（推定） ②
細　菌				
マイコプラズマの一種 *Mycoplasma genitalium*	既知の細胞ゲノムのうちで最小のゲノムをもつ	ヒトの生殖道	580	468
Synechocystis sp.	光合成を行い，酸素を作り出す（シアノバクテリアの一種）	湖や小川	3573	3168
大腸菌 *Escherichia coli*	実験室でよく使われる	ヒトの腸	4639	4289
ヘリコバクター・ピロリ *Helicobacter pylori*	胃潰瘍を起こし，胃がんの原因となる	ヒトの胃	1667	1590
真核生物				
出芽酵母 *Saccharomyces cerevisiae*	最小のモデル真核生物	ブドウ果皮，ビール	12,069	約 6300
シロイヌナズナ *Arabidopsis thaliana*	顕花植物のモデル生物	土壌と大気	約 142,000	約 26,000
線虫 *Caenorhabditis elegans*	発生を完全に記載できる単純な動物	土　壌	約 97,000	約 20,000
キイロショウジョウバエ *Drosophila melanogaster*	動物発生の遺伝学に貢献	腐りかけの果物	約 137,000	約 14,000
ヒト *Homo sapiens*	最も精力的に研究されている哺乳類	家	約 3,200,000	約 24,000

ゲノムサイズや遺伝子数は，特に細菌と古細菌の場合，同じ種でも系統によって異なる。表のデータは配列決定された特定の系統のもの。遺伝子には何通りものタンパク質を生じるものが多いので，ゲノムによって規定されるタンパク質の総数は遺伝子数よりかなり多い。

ショットガン シークエンス法



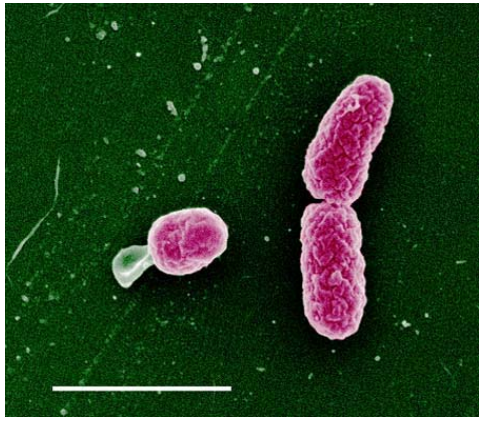


2

DNA は二本鎖なので，**塩基対**(base pair; **bp**)の数で分子の長さを表す。**キロ塩基対**(kilobase pair; **kb**)は 10^3 bp，**メガ塩基対**(megabase pair; **Mb**)は 10^6 bp，**ギガ塩基対**(gigabase pair; **Gb**)は 10^9 bp。まとめると，

1 kb ＝ 1000 bp
1 Mb ＝ 1000 kb ＝ 1,000,000 bp
1 Gb ＝ 1000 Mb ＝ 1,000,000 kb ＝ 1,000,000,000 bp

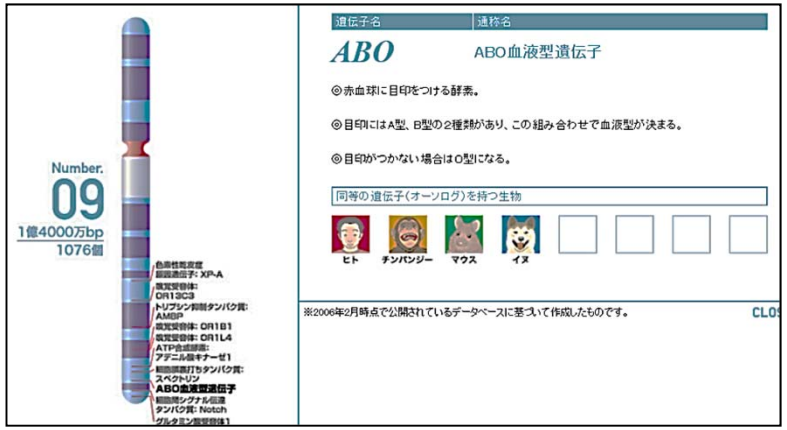
RNA 分子はたいてい一本鎖なので長さの単位に bp は使えず，ヌクレオチドの数で表す。



- 1995年，生物として初めて*Haemophilus influenzae*の全ゲノムが解読された
- その後，多くの生物の全ゲノムが解読され，現在では1万種以上の生物のゲノム情報がデータベースに登録されている

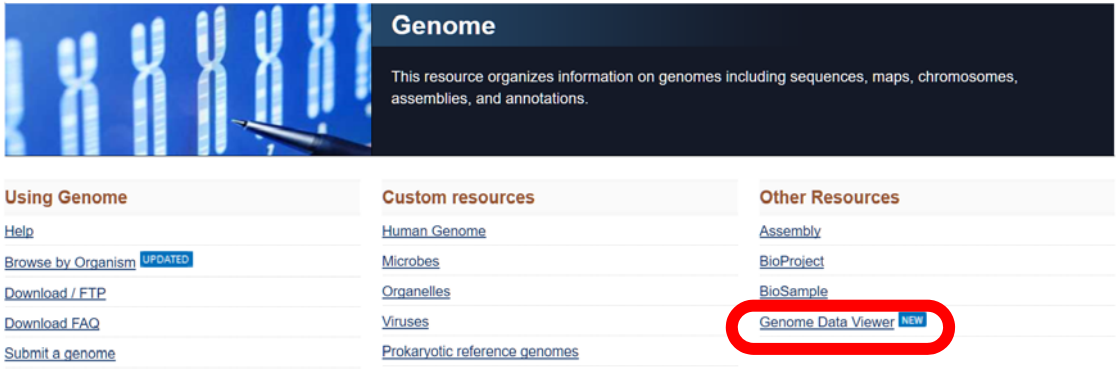
- ヒトゲノムマップ　を開く

http://www.lif.kyoto-u.ac.jp/genomemap/

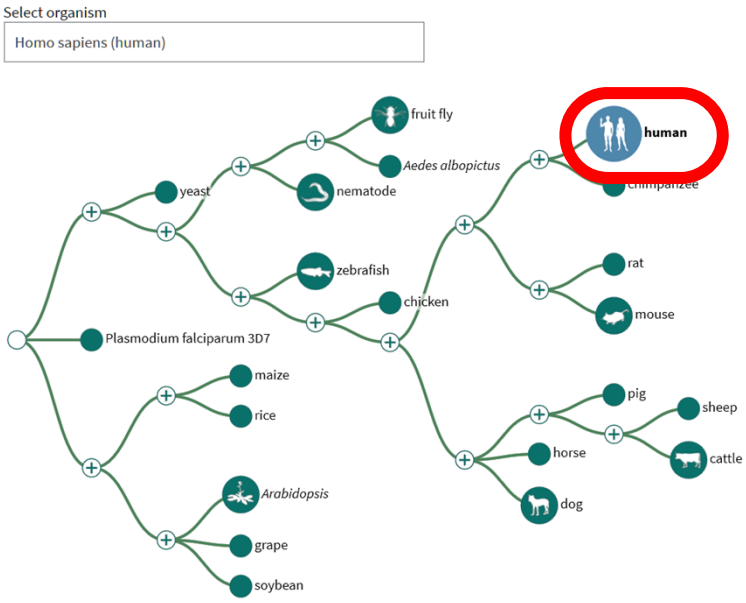


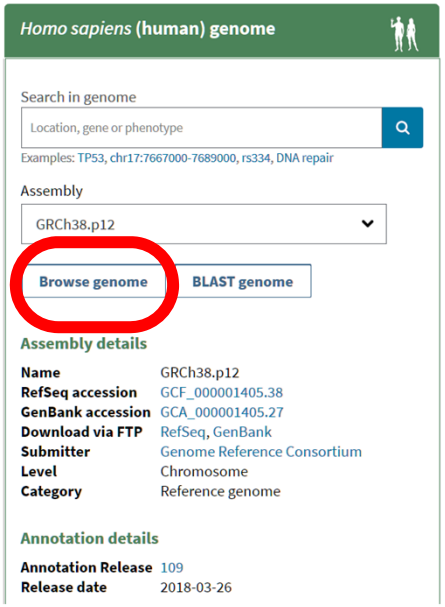
ゲノムブラウザを使ってヒトゲノムを見てみよう

NCBIトップページ右のリンクから「Genome」→「Genome Data Viewer」

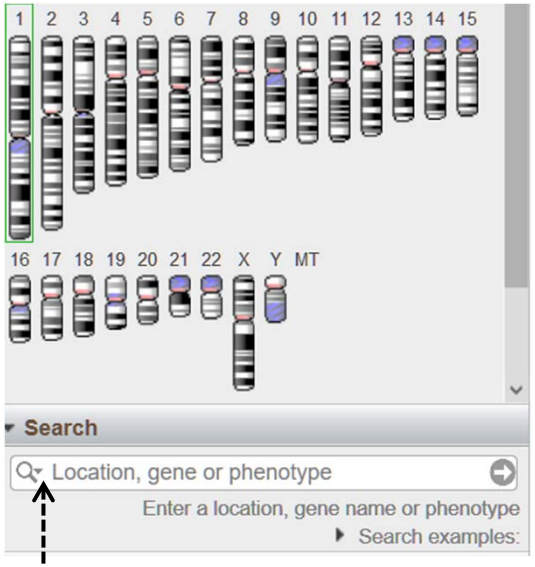


ゲノム解読された真核生物の
系統図が表示される
「human」
↓
「Browse genome」
をクリック





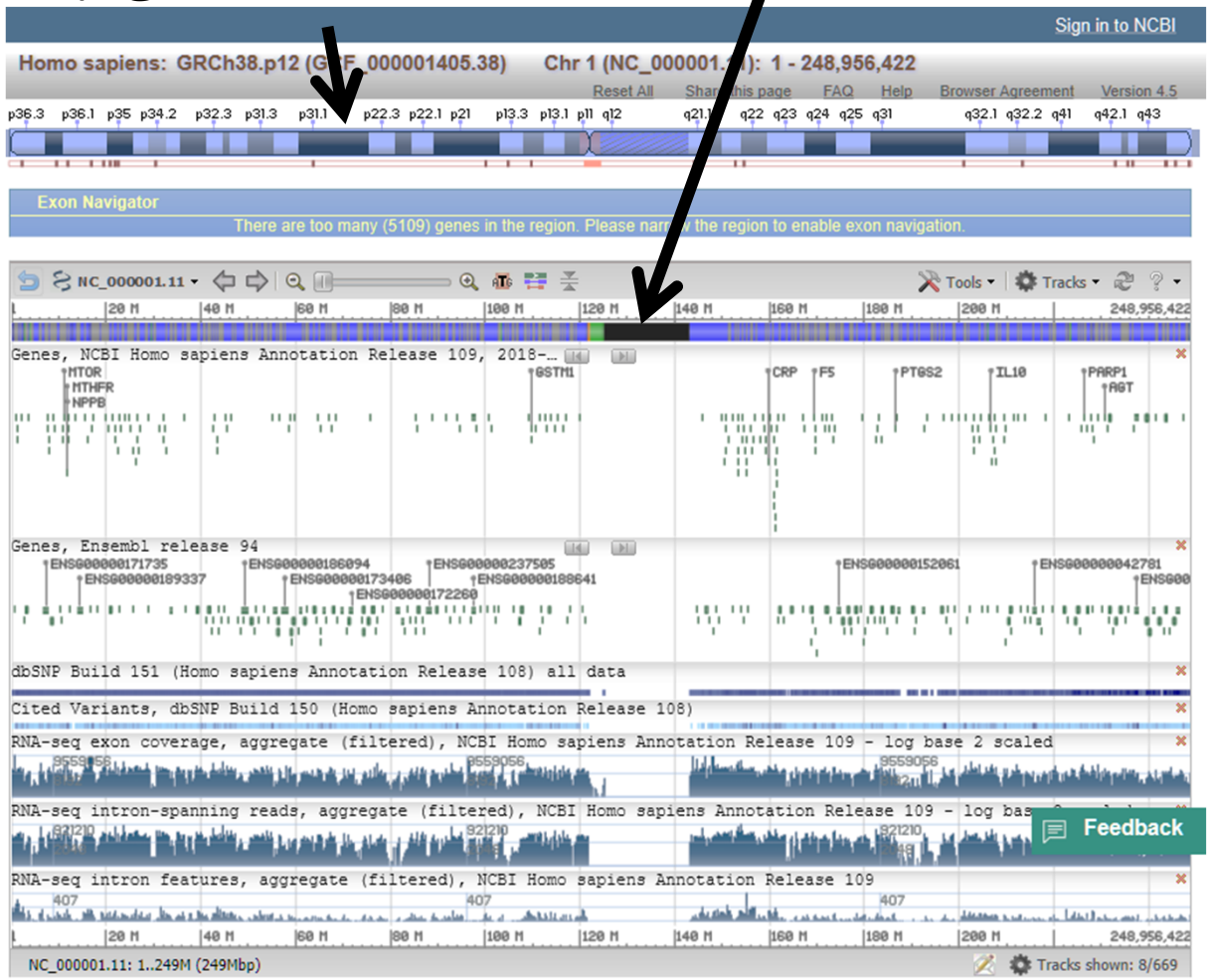
クリックした染色体のゲノムマップが表示される



3 「ABO」と入力してみる

1 青い領域にコードされる遺伝子が下に表示される

2 黒い領域は塩基配列が決まっていない



遺伝子マップ

太線：エキソン

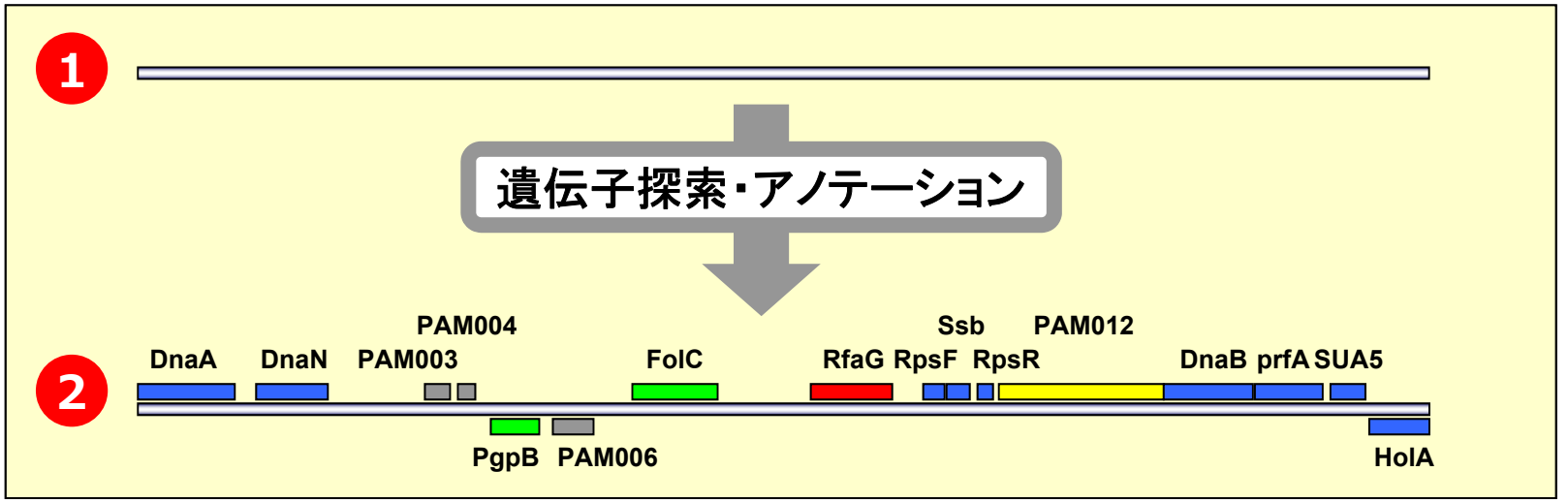
細線：イントロン

遺伝子発現量

セントロメアには遺伝子がコードされていない

遺伝子探索

```
agttattttatttttcattttataattaaaagccaatttaaaaaaggagtattaattatgccat
atattgaaagtattttagcgcgcgaagtgctagattccagaggaaatcctacagtagaagtaga
agtttatacagaatcaggagcgtttggaagagctattgttccttcaggagcttctaccggacaa
tacgaagcagttgaattaagggatggggatgcccaaagattttaggtaaaggcgttttgcaag
ctgttaaaaatgttattgaagttattcaaccagaattagaaggttattctgtcttagaacaaac
tttaattgataaattattaattaaacttgacggaactcctaacaaatctaatttaggagctaac
gctattttaggtgtttctttggcttgtgctaaagctgcagctaactacttaaatcttgagtttt
atcaatatgtaggaggcgttttacctaaacaaatgccagttcctatgatgaatgttatcaacgg
tggagc.....
```

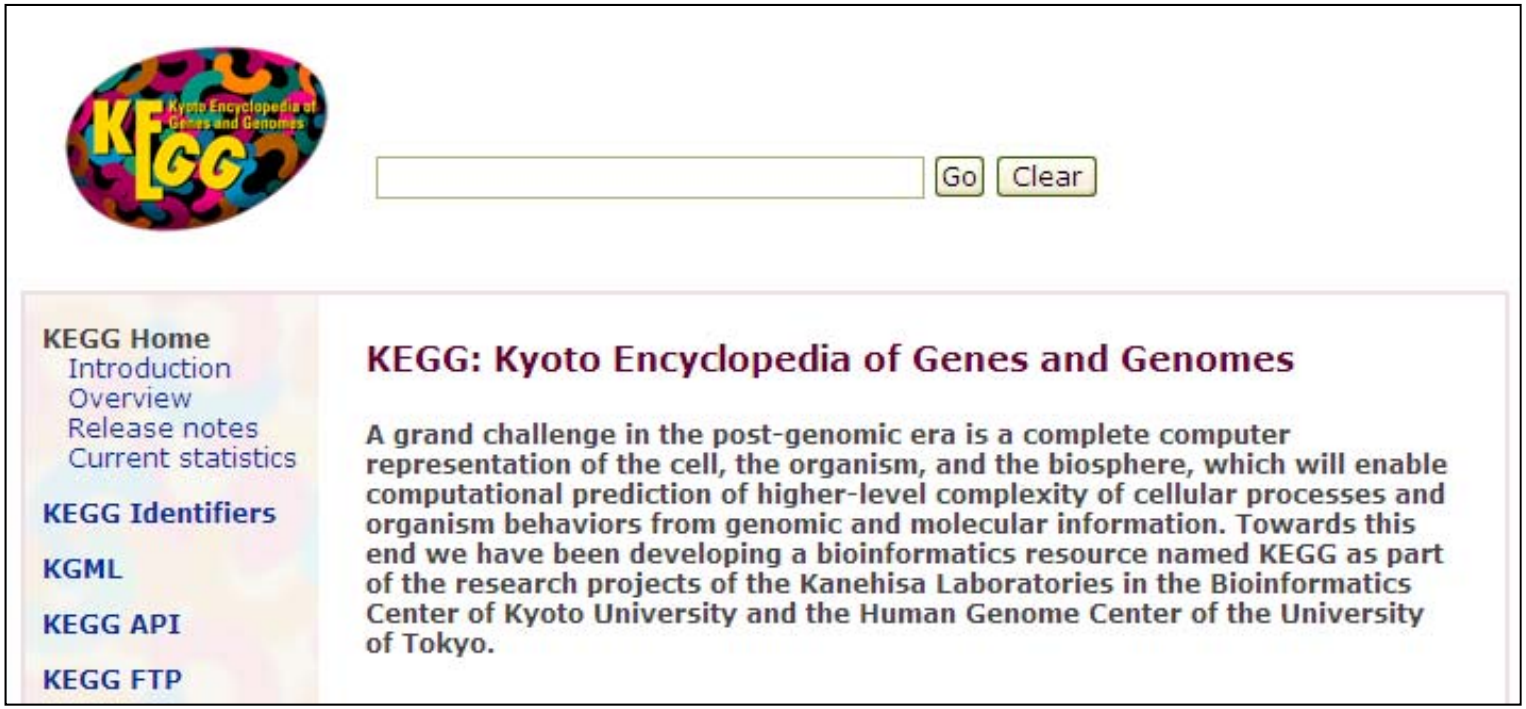


ゲノム配列から生命活動に関わる機能や分子進化に関する考察などを行うためには，
タンパク質をコードしている遺伝子領域を同定することが重要となる．

代謝パスウェイデータベース

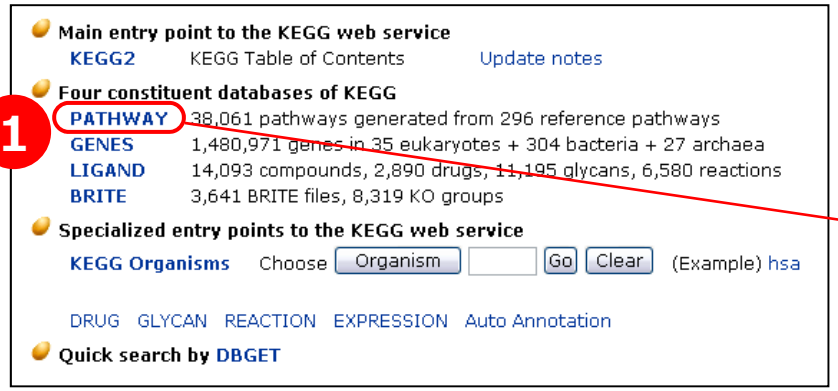
KEGG

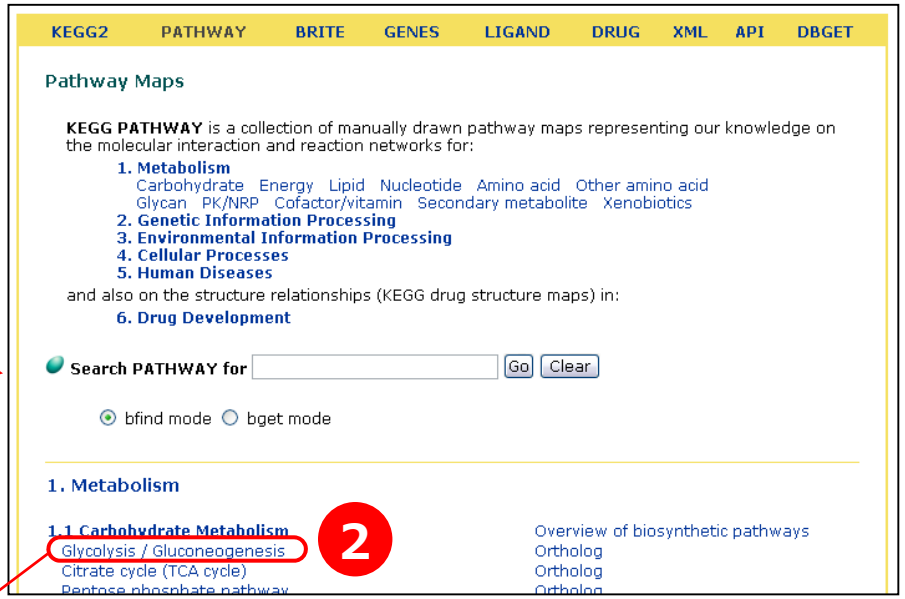
http://www.genome.jp/kegg/

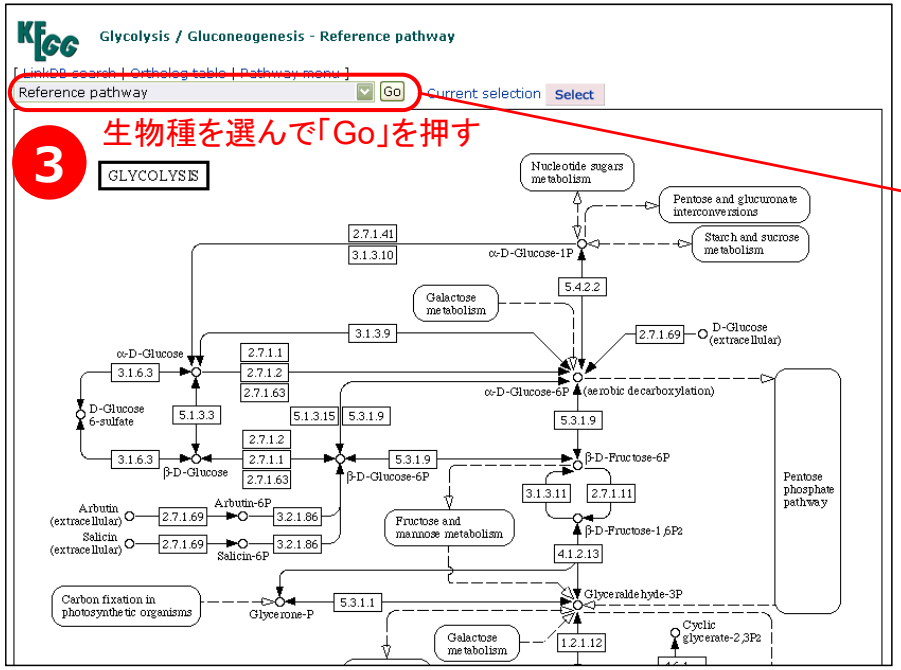


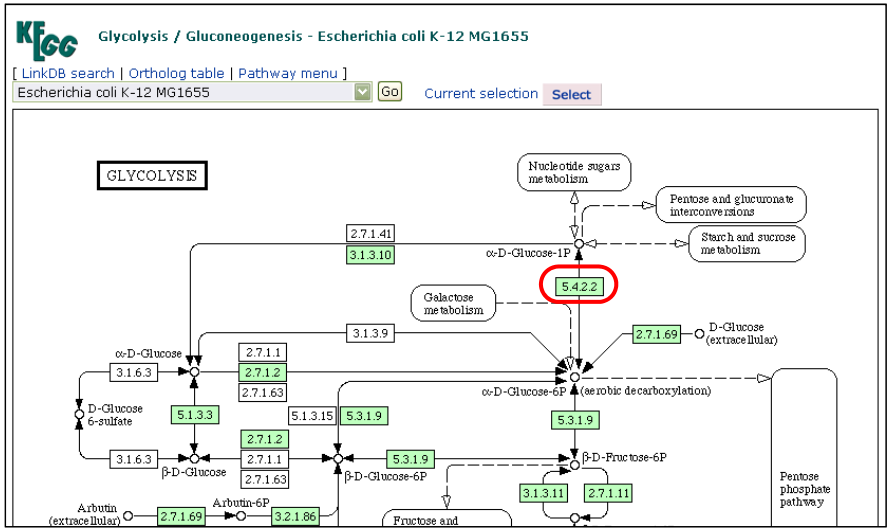
生命システム情報統合データベース。完全にゲノムが決まった生物種（一部、ドラフト配列も含む）の代謝系や一部の制御系（シグナル伝達や細胞周期など）をまとめ、そこから様々な物質データベースや 酵素データベースを参照することができる。

代謝系データベースの参照









・選んだ生物のゲノムにコードされている酵素にだけ色がつく
・例えば「Rickettsia prowazekii」を選ぶと，ほとんど色がつかず，解糖系を持っていないことがわかる

次世代シーケンサー

Roche Diagnostics社 1

Genome Sequencer FLX System（454）

2005年発売

Applied Biosystems社

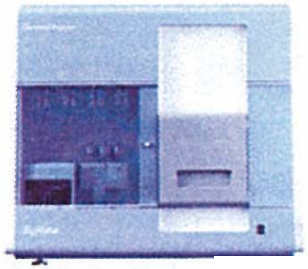
SOLiD 3

2007年発売

Solexa / illumina社

Genome Analyzer IIx

2005発売



ライフテクノロジー社

Ion PGM

イルミナ株式会社

MiSeq 2

PacBio RS II

次世代シーケンサーの比較

	Ion Protonシステム	Ion PGMシステム Ion 318 chip	MiSeq	HiSeq 2000/2500 （SBS v3試薬使用）	PacBio *RS* II
1リード長	～200 base		150/250/300 base	100 base	約10,000 base
リード数	約5,000万リード （1ランあたり）	約400万リード （1ランあたり）	約3,000万リード （1ランあたり） ※ペアエンド解析	1 約3億リード （1レーンあたり） ※ペアエンド解析	約5万リード （1セルあたり）
データ量（リード長200 baseの場合）	約7.5 Gb （平均150 bpのamplicon、 1チップあたり）	約800 Mb （1チップあたり）	約3～9 Gb （1ランあたり） ※ペアエンド解析	2 約30 Gb （1レーンあたり） ※ペアエンド解析	約500 Mb （1セルあたり）
解析手法	Ion semiconductor sequencing法		Sequencing by Synthesis法	Sequencing by Synthesis法	SMRT(Silgle Molecule Real-Time) sequencing法
アプリケーション例	・癌遺伝子などの変異解析 （409遺伝子をターゲットとしたCancer Panelなど）	・癌遺伝子などの変異解析 （50遺伝子をターゲットとしたCancer Panelなど）	・微生物の新規ドラフト配列決定 ・癌遺伝子などの変異解析 ・PCR産物のディープシーケンス	・ゲノム変異解析 ・ChIP解析 ・small RNA解析 ・mRNA解析 ・cDNA配列解析	・ゲノムドラフト解析 ・cDNA配列解析

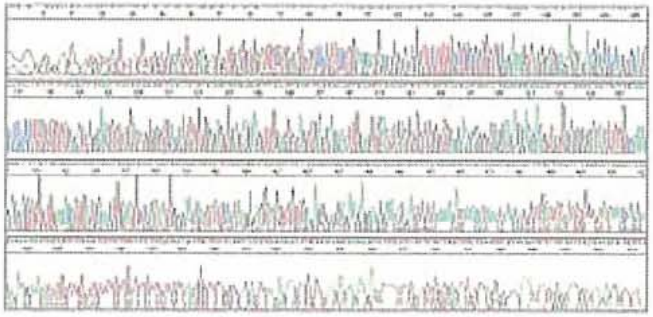
イルミナ株式会社

次世代シーケンサー：Genome Analyzer

MiSeq

1 従来型キャピラリーシーケンサー

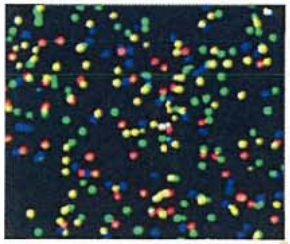
- 酵素反応＋電気泳導＋塩基読取（384x600塩基）
- コスト、時間がかかる
- 例）「ヒトゲノムプロジェクト」
  約13年、3000億円かかった



2 次世代シーケンサー

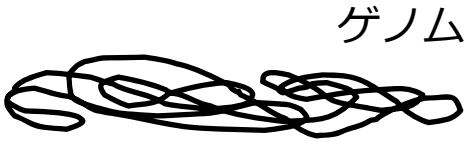
- 酵素反応＋電気泳導＋塩基読取 （100,000,000x50塩基）
- これまでの技術と比べて、「100分の1のコストで100倍のデータ」
- 例）現在ヒトゲノム1人読むのに 数週間、数千万円

→ 1週間 数百万円 ・・・



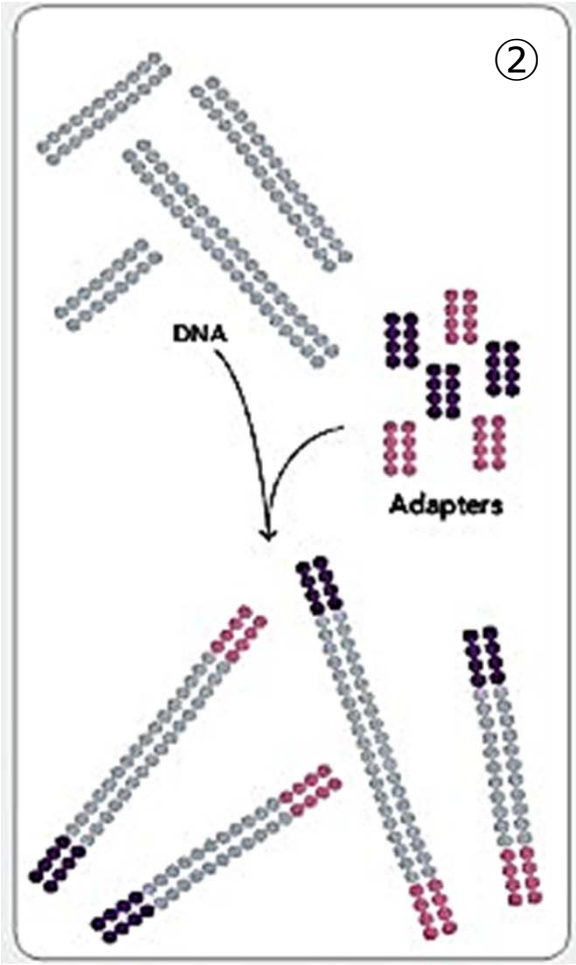
illumina

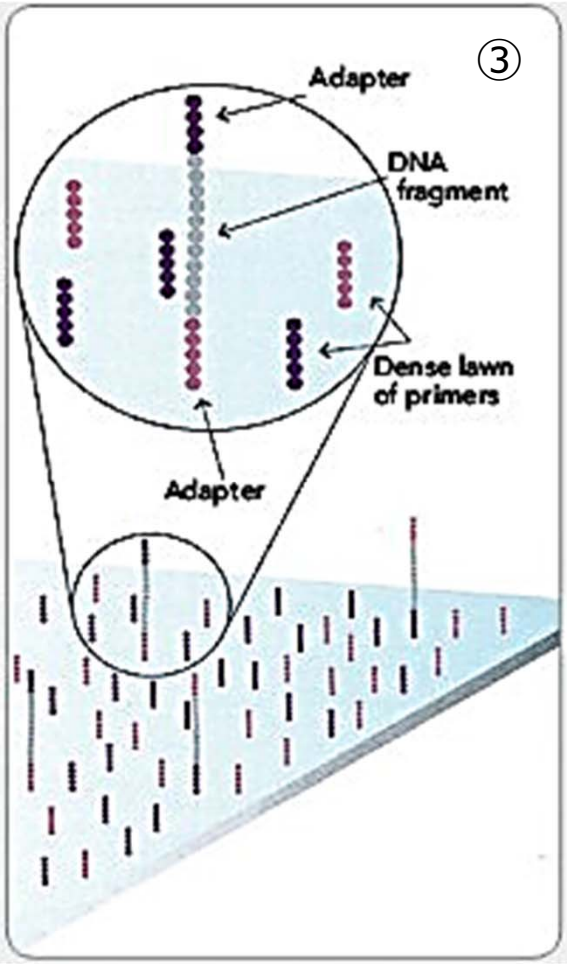
サンプル調製　～　フローセルへの固定



① 断片化

- ゲノムDNAを抽出し、**断片化**する
- DNA断片の両端に2種類の**アダプター**を連結させる





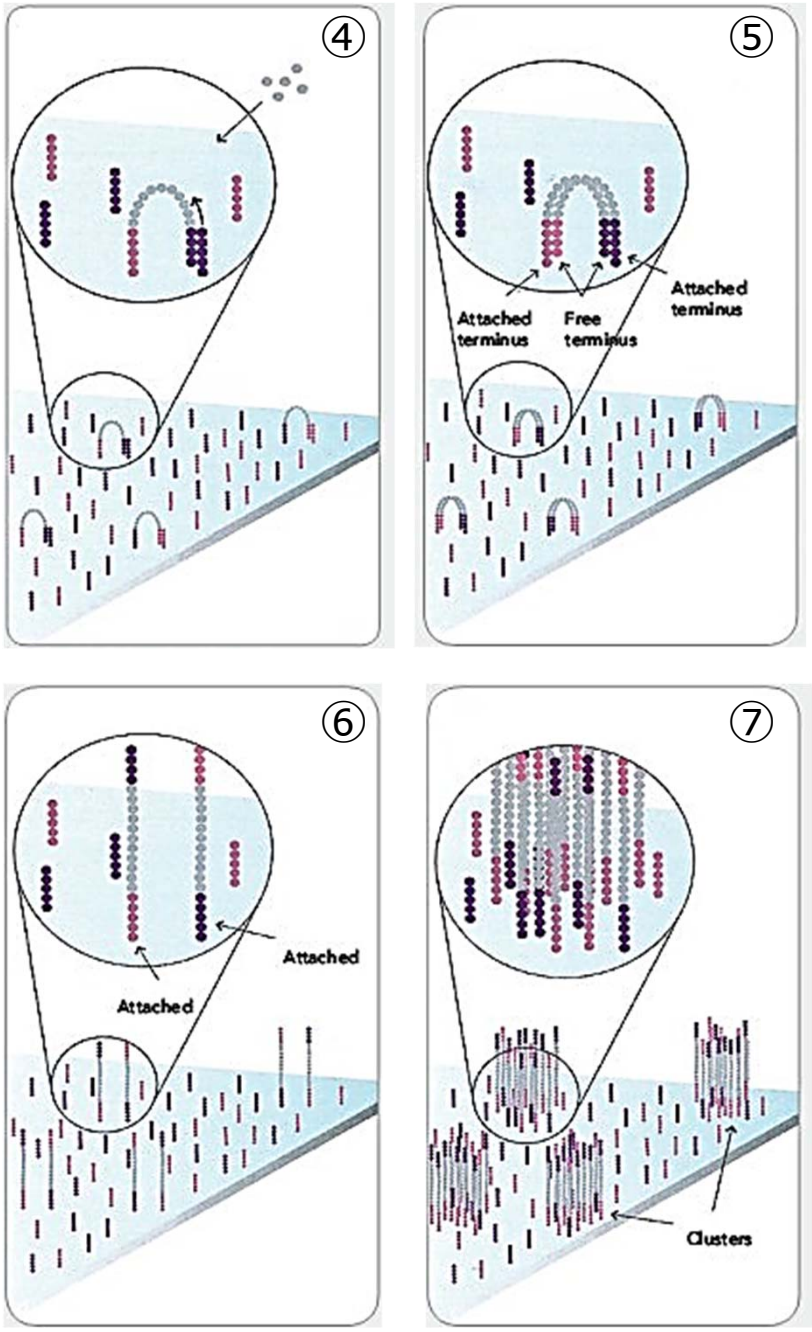
- 1本鎖にして、5’末端を**フローセル**上に固定する
- フローセル上には、あらかじめアダプターと相補的に結合するプライマーが高密度に配置されている

ブリッジPCR

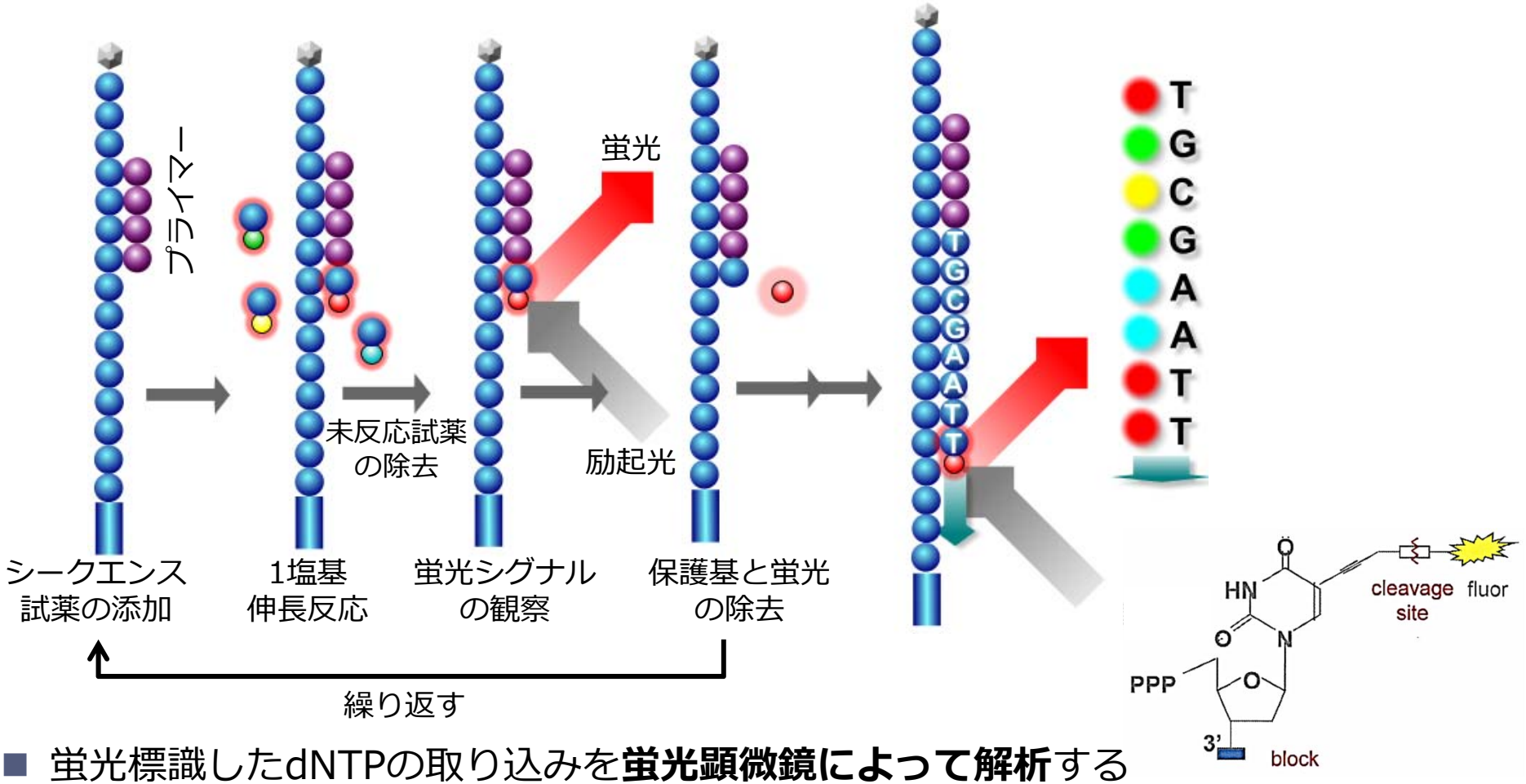
- 固定された1本鎖DNAは、もう一方のアダプターの側でプライマーと結合する (**橋がかかったような構造**になる ④)
- **DNAポリメラーゼ**による伸長反応を行う ⑤
- 変性させると、フローセル上には根元がアダプター配列の1本鎖DNAが2本できあがる ⑥
- この反応を繰り返すことで、狭い面積の中でDNAを増幅することができる

  → フローセル上に多数の**DNAの「束」**ができる ⑦

- これらを鋳型として、配列解析を行う



Sequencing-by-synthesisによる塩基配列決定

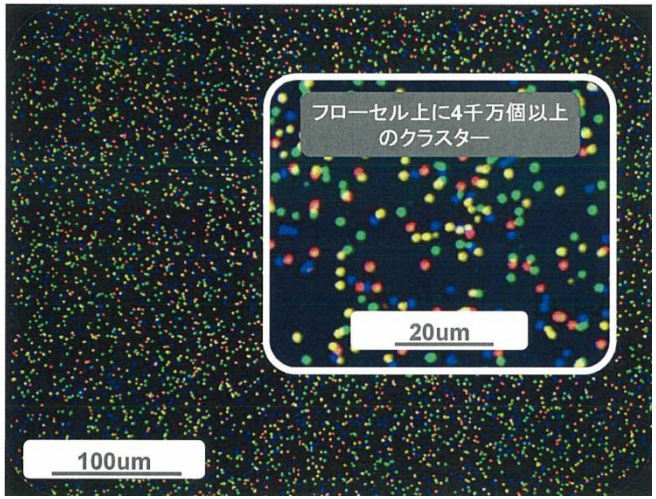


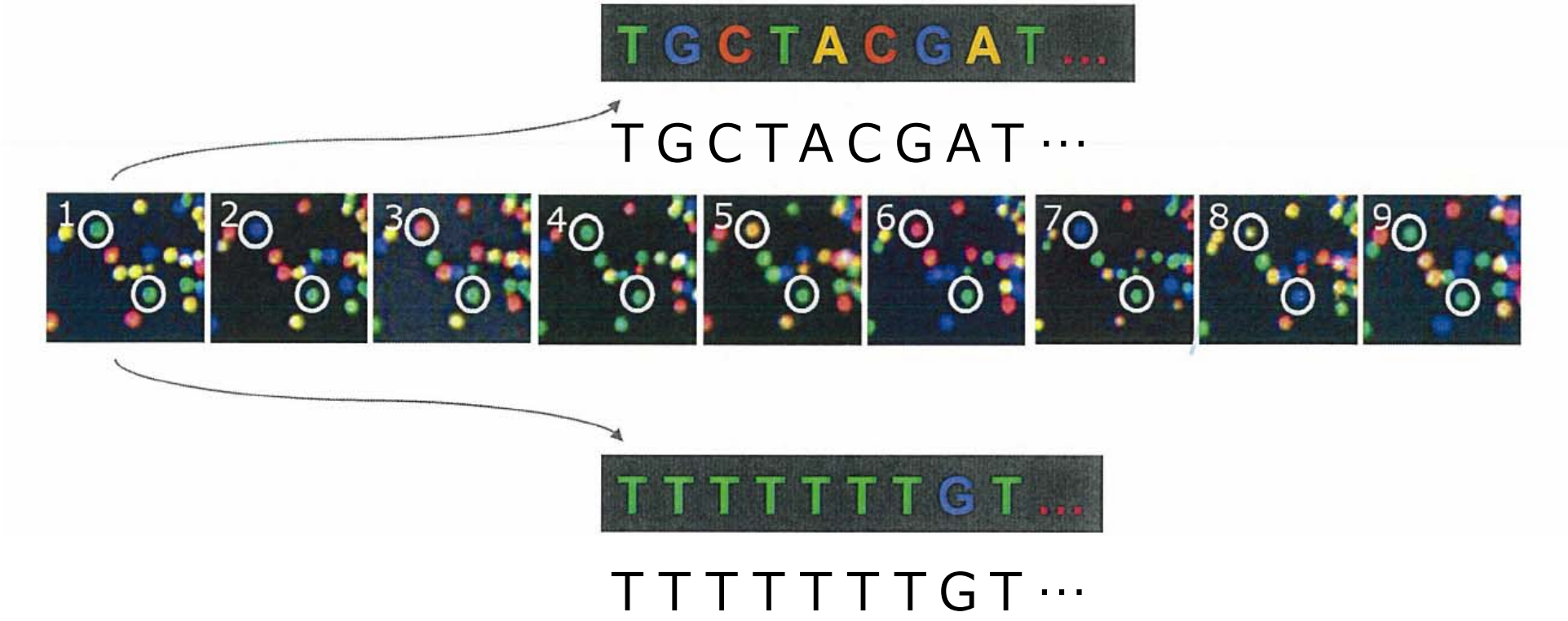
- 蛍光標識したdNTPの取り込みを**蛍光顕微鏡によって解析**する
- このdNTPは3’末端がブロックされており、1回の伸長反応で1塩基しか伸ばせない
- そのため、1塩基ごとにどのdNTPが取り込まれたかを観察し、蛍光物質とブロックを外して次の伸長反応を行うというステップで、解析を進めていく

画像蛍光シグナルから塩基への返還

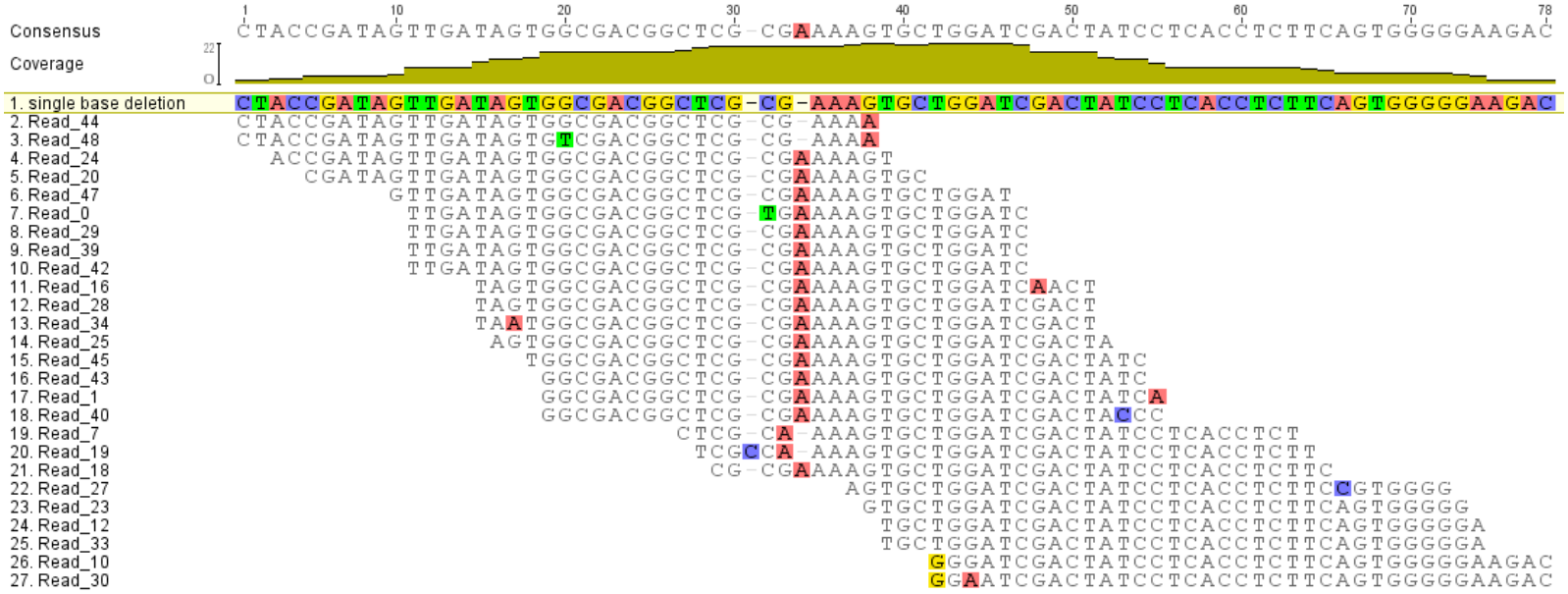
- **1塩基 伸長するごとに**蛍光イメージを取得する
- それぞれのDNAの「束」の蛍光色の変化を調べることで、塩基配列を決定する
- 数千万～数億本の塩基配列が得られる

画像イメージの取り込み





- 一つ一つの断片の塩基配列が短いと，アセンブルするのが困難
- 次世代シーケンサーで読み取ることができる塩基配列長は短いので，既に全塩基が解読されているゲノム配列（**リファレンス配列**）を利用した**リシークエンス**や、リファレンス配列への**マッピング**などに用いられることが多い



マッピング：Bowtie, Bowtie2, BWA など

アセンブル：Velvet，EDENA，Phrap など

ビューア：Tablet，IGV など

有償ではあるが、CLC Genomics Workbenchなどの解析ソフトも良く使われる

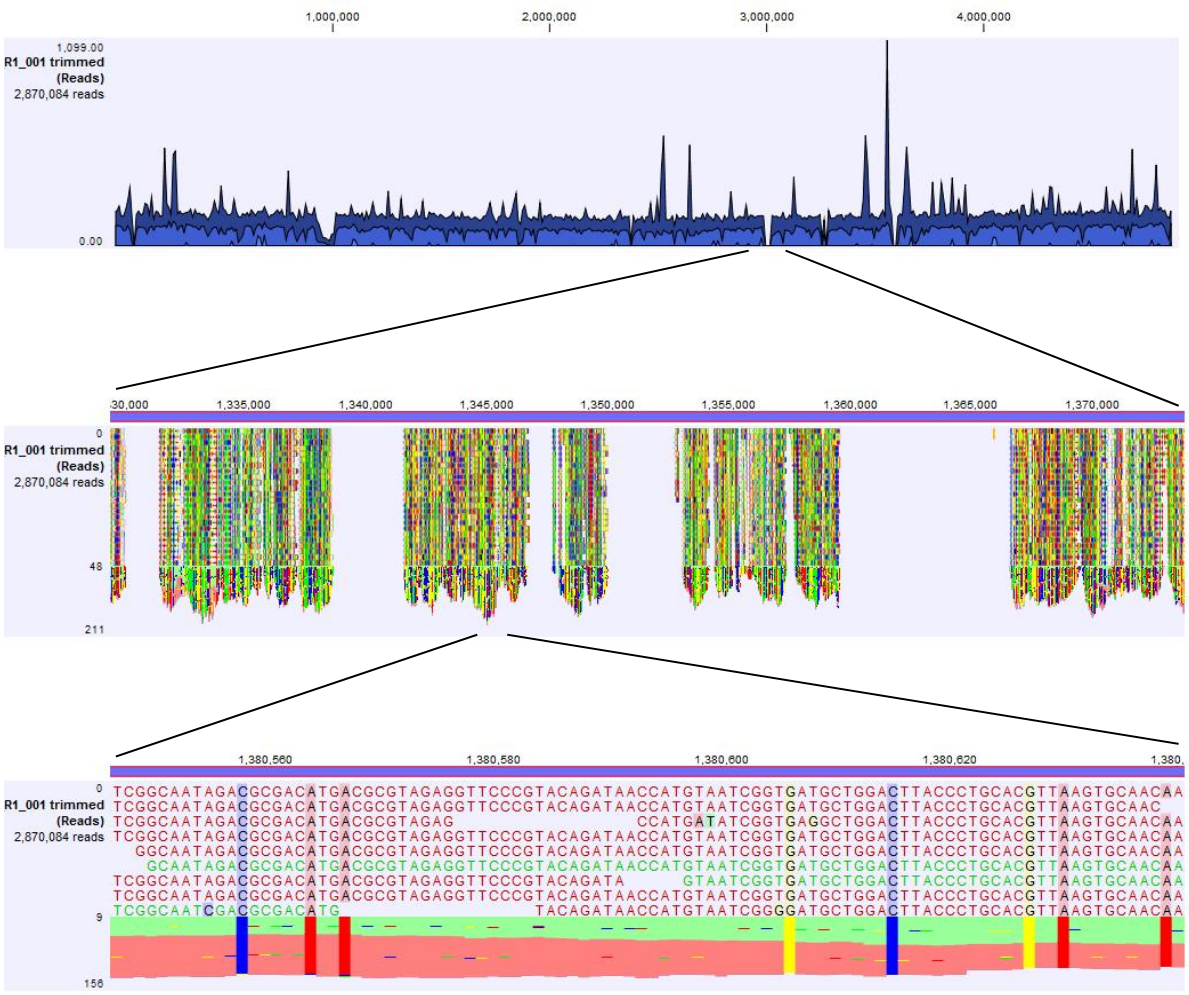
Pectobacterium carotovorum ssp. *carotovorum*

Pectobacterium carotovorum PR1 strain （病原性 強い）

P. carotovorum PR1株のゲノムを抽出

MiSeqを用いてシークエンス（約300万リード）

P. carotovorum ssp. *carotovorum* のゲノム（リファレンス配列）に対してマッピング



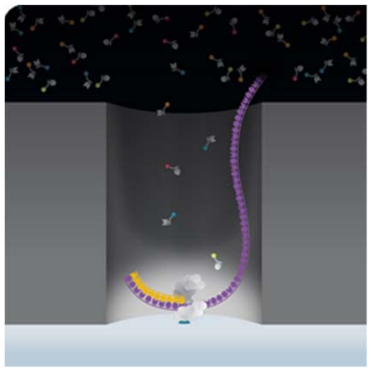
- **遺伝子の有無**
- **ゲノム構造の比較**
- **SNPの検出**

等の**比較ゲノム解析**ができる

第３世代シーケンサー

Pacbio RS II DNA Sequencing System

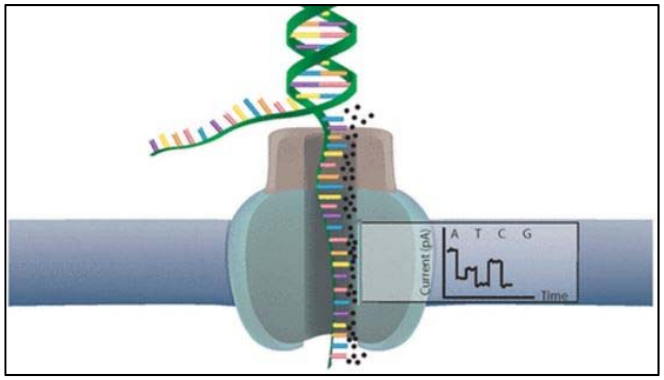
1



- DNA**１分子**を鋳型としてDNAポリメラーゼによるDNA合成を行う
- １分子レベルでリアルタイムに塩基を読み取る
- 長いリード(平均10,000bp)が出力される

MinION

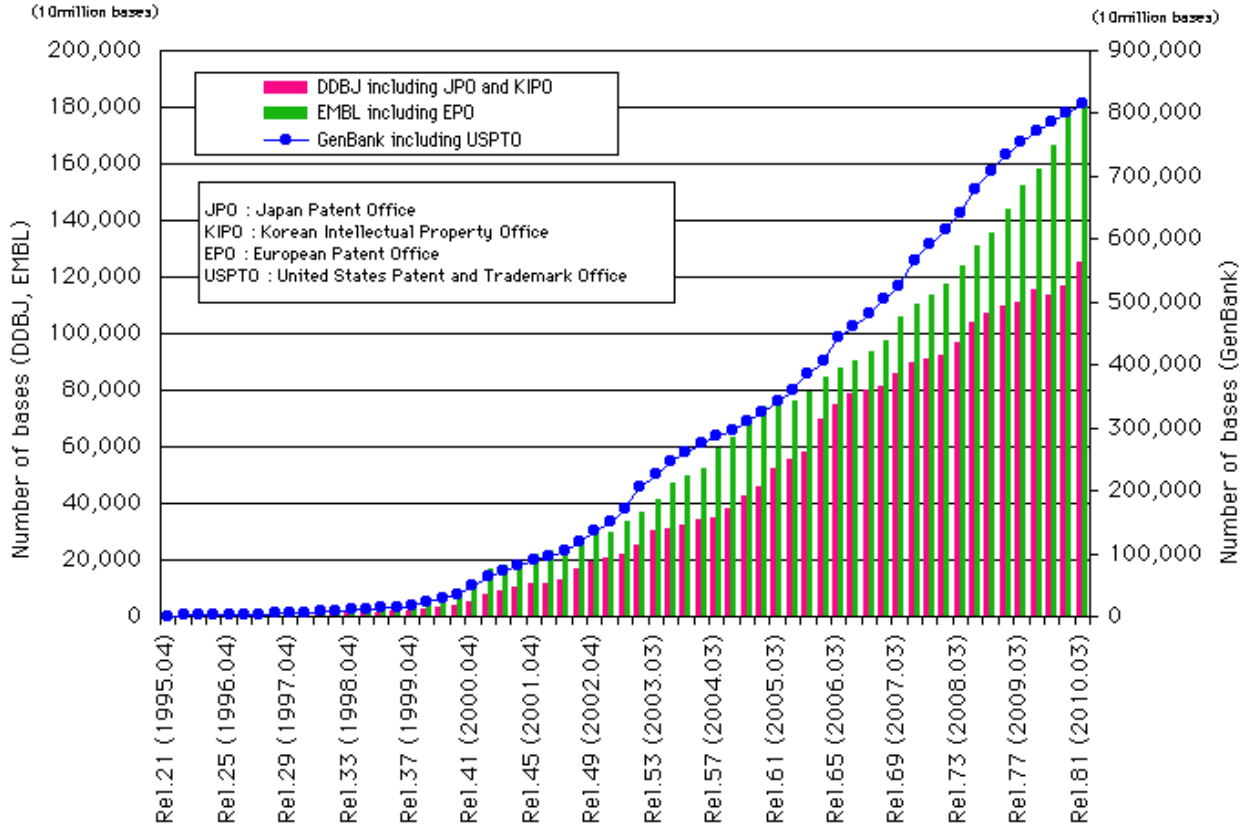
2



- USBメモリー用のシーケンサー
- DNAポリメラーゼを用いて１本鎖DNAに解きほぐす
- **ナノポア**を通過させる　→　電流の変化を検知して配列を決定する

塩基データ登録数の推移

- **シークエンス技術の進歩**によって，塩基配列決定の速度はますます加速している



- 遺伝子の検出，アノテーション，機能予測，進化系統解析，比較解析などを効率よく行い，大量のシーケンスデータを有効に活用することが重要

ゲノムにコードされる遺伝子を網羅的に使用してホモロジー検索を行ったり，比較ゲノム解析を行いたい

大量のデータを処理するためのプログラミング技術が必要

バイオインフォマティクス分野では，Perl，C++，Java，Python などが良く使われていますが，本日はPerlを用いて実習を行います

Perlの特徴

- テキスト処理が得意
- 歴史が長いのでライブラリーが豊富
- 掲示板やショッピングカートなど、CGIという仕組みの多くがPerlで書かれている
- LinuxやMacOSに標準でインストールされているほか、Windowsにもインストール可能

プログラミング言語の人気ランキング（2016）

【第1位】 Java
【第2位】 C言語
【第3位】 C++
【第4位】 Python
【第5位】 C#
【第6位】 Visual Basic
【第7位】 JavaScript
【第8位】 PHP
【第9位】 Perl
【第10位】 アセンブリ言語

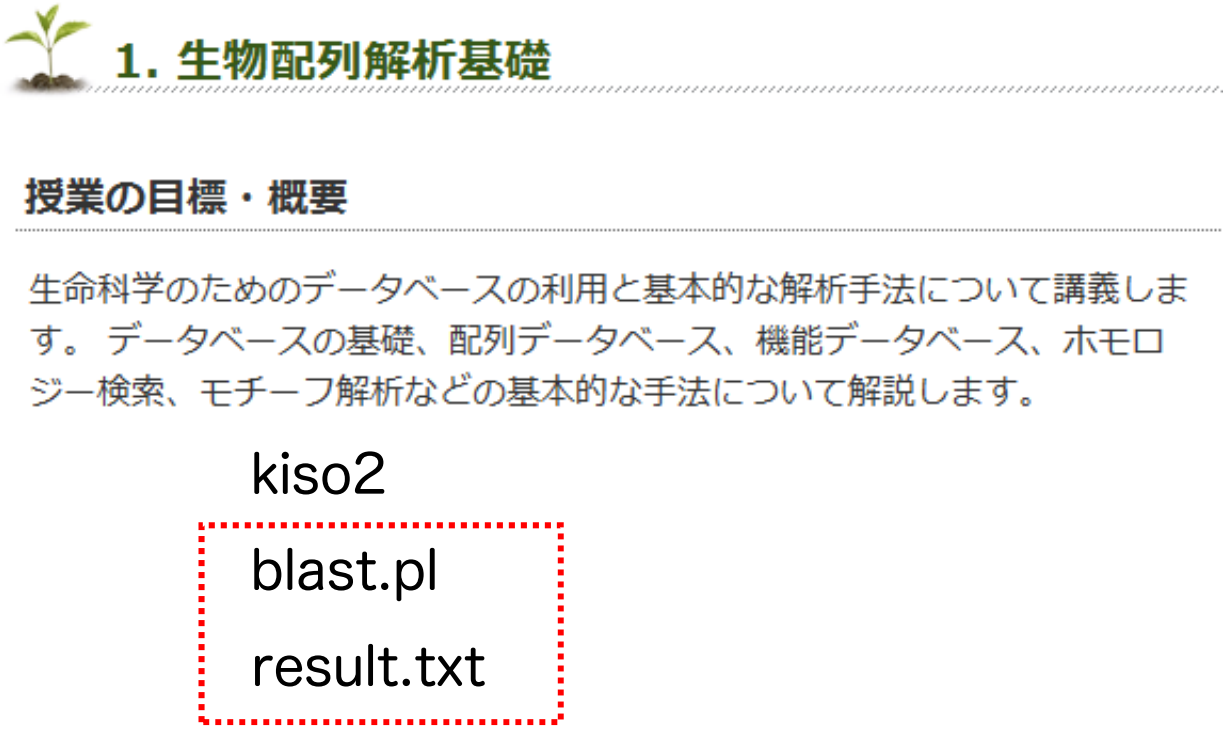
perlを用いたデータ処理

- 大量のQueryに対して**BLAST検索**を行うと，結果が羅列した形で出力されます
- この中から，**必要な情報だけを取り出す**ためのプログラムをperlで組んでみましょう
- Queryのアクセッション番号と，検索の結果ヒットしたタンパク質のアクセッション番号とのリストを作成し，下に示したように検索結果を整理したいと思います

Query	1	2	3	4	5
gi|49176138|ref|NP_416237.3|	ref|NP_009965.1|	ref|NP_010402.1|	ref|NP_012405.1|	ref|NP_012934.1|	ref|NP_015093.1|
gi|16132212|ref|NP_418812.1|	ref|NP_014926.1|	ref|NP_012380.1|	ref|NP_012969.1|	ref|NP_012770.1|	ref|NP_014585.1|
gi|16131851|ref|NP_418449.1|	ref|NP_009755.1|	ref|NP_011646.1|	ref|NP_013146.1|	ref|NP_013847.1|	ref|NP_013523.1|
gi|16131757|ref|NP_418354.1|	ref|NP_010335.1|	ref|NP_012586.1|			
gi|16131754|ref|NP_418351.1|	ref|NP_011756.1|	ref|NP_013932.1|	ref|NP_010104.1|	ref|NP_011671.1|	
gi|16131018|ref|NP_417595.1|	ref|NP_009362.1|	ref|NP_014992.1|	ref|NP_014792.1|	ref|NP_014227.1|	ref|NP_011015.1|
gi|16130827|ref|NP_417401.1|	ref|NP_009938.1|	ref|NP_011705.1|	ref|NP_011575.1|	ref|NP_011569.1|	ref|NP_012819.1|
gi|16130826|ref|NP_417400.1|	ref|NP_012863.1|	ref|NP_013282.1|	ref|NP_015308.1|	ref|NP_012835.1|	ref|NP_010022.1|
gi|16130686|ref|NP_417259.1|	ref|NP_011770.1|	ref|NP_012044.1|	ref|NP_014056.1|	ref|NP_015042.1|	ref|NP_015038.1|
gi|16130106|ref|NP_416673.1|	ref|NP_009965.1|	ref|NP_014276.1|	ref|NP_012639.1|	ref|NP_013060.1|	ref|NP_013066.1|

```
BLASTP 2.2.5 [Nov-16-2002]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= gi|16131851|ref|NP_418449.1| glucosephosphate isomerase
[Escherichia coli K12]
         (549 letters)

Database: yeast.aa
           6298 sequences; 2,974,038 total letters



                                                                  Score    E
Sequences producing significant alignments:                       (bits) Value

ref|NP_009755.1| Glucose-6-phosphate isomerase; Pgi1p                641   0.0
ref|NP_011646.1| Ygr130cp                                            30   0.98
ref|NP_013146.1| spindle pole body component; Stu2p                  29   1.7
ref|NP_013847.1| (putative) involved in cell wall biogenesis; Ec...   28   3.7
ref|NP_013523.1| Ylr419wp                                            28   3.7

>ref|NP_009755.1| Glucose-6-phosphate isomerase; Pgi1p
          Length = 554

 Score =  641 bits (1654), Expect = 0.0
 Identities = 326/549 (59%), Positives = 401/549 (73%), Gaps = 16/549 (2%)

Query: 7   TQTAAWQALQKHFDEM-KDVTIADLFAKDGDRFSKFSATFDD----QMLVDYSKNRITEE 61
           T+  AW  LQK ++   K +++   F KD  RF K + TF +    ++L DYSKN + +E
Sbjct: 13  TELPAWSKLQKIYESQGKTLSVKQEFQKDAKRFEKLNKTFTNYDGSKILFDYSKNLVNDE 72

Query: 62  TLAKLQDLAKECDLAGAIKSMFSGEKINRTENRAVLHVALRNRSNTPILVDGKDVMPEVN 121
            +A L +LAKE ++ G   +MF GE IN TE+RAV HVALRNR+N P+ VDG +V PEV+
Sbjct: 73  IIAALIELAKEANVTGLRDAMFKGEHINSTEDRAVYHVALRNRANKPMYVDGVNVAPEVD 132
```

◆デスクトップ上に，「kiso」フォルダを作成してください



result.txt

blast.pl

の２つのファイルをダウンロードして，kisoフォルダに入れてください

- コマンドプロンプトを立ち上げてください

 スタート → Windowsシステムツール → コマンドプロンプト

まず，kisoフォルダに移動します

```
> cd␣
```

「cd（スペース）」と入力した後（まだEnterキーは押さない），kisoフォルダをコマンドプロンプト上にドラッグ＆ドロップしてください

下記のように表示されますので，Enterキーを押してください

```
> cd C:¥Users¥iu¥Desktop¥kiso
```

- **blast.pl**をメモ帳やワードパッドなどを使って開いてください

```
#! /usr/local/bin/perl
```

- 以下のように編集して，上書き保存してください

```
#! /usr/local/bin/perl
print "Hello!¥n";
```

¥n は改行を表します

「￥」は，バックスラッシュ「＼」を押してください

Windows上だと「￥」と表示されます

以下のコマンドを入力して，プログラムを実行してください

```
> perl blast.pl
```

変数

- 変数は，「**$文字列**」で表します
- 以下のように編集して保存してください

```
#! /usr/local/bin/perl
$a = "Hello!¥n";
print $a;
```

「；」を入力し忘れないように注意してください

Perlでは**行の終わり**に「；」をつける決まりになっています

以下のコマンドを入力して，プログラムを実行してください

```
> perl blast.pl
```

<STDIN>

- **<STDIN>**は，１回呼び出すごとに**標準入力**から１行のデータを読み出す命令です．STDINとは，standard inputつまり標準入力の略です．

- 以下のプログラムに修正して保存してください．

  blast.pl

```
#! /usr/local/bin/perl
$a = <STDIN>;          #  標準入力から1行のデータを読み出し，$aに代入する
print $a;              #  $aを出力する
```

- 以下のコマンドを打ち込みblast.plを実行すると，**入力待ち**になります．キーボードで何か文字を入力すると，入力した文字がそのまま出力されます．

```
> perl blast1.pl
```

リダイレクト

- キーボードから文字列を打ち込む代わりに，既存のテキストファイルから**データを読み込ませる**こともできます.

- 「result.txt」という**BLAST検索結果のファイル**を用意しておきました．中身を見てください.

```
BLASTP 2.2.19 [Nov-02-2008]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.
          .
          .
```

リダイレクト

- 「result.txt」を読みこませるために，リダイレクトという機能を使います．

```
> perl blast.pl < result.txt
```

❶

- 「result.txt」の一番最初の行だけが表示されます．

- 結果を，画面でなく，ファイルに出力することもできます．

```
> perl blast.pl < result.txt > output1.txt
```

❷ ❸

whileループ

- 読み込むデータが何行であっても良いように，**whileループ**を利用してみましょう．

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {   # データを1行ずつ$aに
        print $a;        # 代入して，出力する
}
```

読み込むデータ（行）がある限り「**真**」となり，{ }の中が繰り返されます．

- 実行してみましょう．全てのデータが出力されるはずです．

```
> perl blast.pl < result.txt > output1.txt
```

パターンに当てはまるかどうかを調べる「=~ / /」

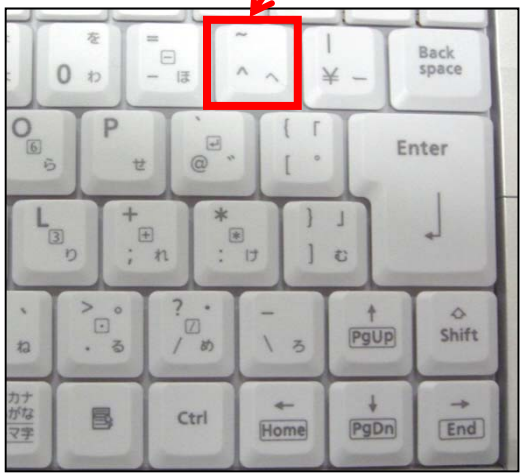
- 文字列「DNA」を含む行を表示してみましょう.

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /DNA/) {      # DNAを見つけたら
      print $a;           # 出力する
  }
}
```

- 「=~」をパターン結合演算子，「**/文字列/**」をマッチ演算子といいます.
- 文字列の一部に一致すれば「**真**」を，一致しなければ「**偽**」を返します
- **$a =~ /DNA/** は，$aにDNAという文字列が含まれていれば「**真**」となり，if文の { } が実行されます.

「~」チルダ

Shiftを押しながら



- 文字列「Query=」を含む行を表示してみましょう.

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query=/) {  # Query=を見つけたら
      print $a;          # 出力する
  }
}
```

質問配列の行だけを抽出することができるようになりました

```
Query= gi|13507740|ref|NP_109689.1| DNA polymerase III beta subunit
Query= gi|13507741|ref|NP_109690.1| similar to j-domain of DnaJ
Query= gi|13507742|ref|NP_109691.1| DNA gyrase subunit B [Mycoplasma
Query= gi|13507743|ref|NP_109692.1| DNA gyrase subunit A [Mycoplasma
Query= gi|13507744|ref|NP_109693.1| seryl-tRNA synthetase [Mycoplasma
                                    .
                                    .
                                    .
```

正規表現による検索

- マッチ演算子の / と / の間には，文字列だけでなく，**パターン**と呼ばれるものを入れることができます
- パターンとは，「Mで始まる文字列」や「3文字の文字列」など，**文字列の特徴**を記述したものです
- このパターンの記述方法を**正規表現**といいます．

例えば，

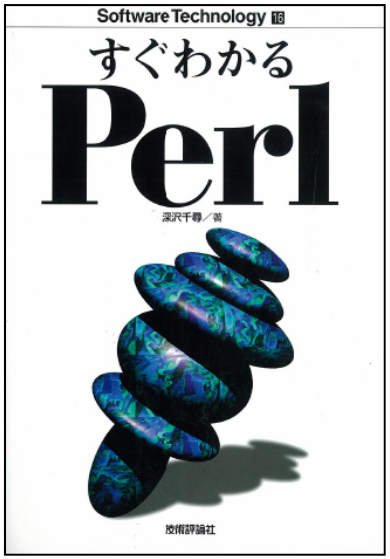
DNA

DNNA

DNNNNNNA

DNNNNNNNNNNNNNNNNA

これらすべてを検索するには，**/DN+A/** と記述します



文字クラス

以下のものは、次のような1文字にマッチします。

	[abc]	aかbかcのどれか
	[a-z]	任意の小文字
	[^abc]	aでもbでもcでもない文字
3	¥d	数字 (digit)
	¥D	数字以外
	¥w	英数字 (word)
	¥W	英数字以外
2	¥s	空白文字 (space)
	¥S	空白文字以外
	¥b	単語境界 (word boundary)
1	.	任意の一文字

位置指定

パターンの位置を指定します。

4	^	先頭
	$	末尾

エスケープ

/、^、$ などの、正規表現的に意味のある特殊記号自体を検索したい局面では、¥でエスケープします。

	^¥^	^という字で始まる行にマッチ
5	¥¥	¥自体にマッチ

繰り返し

以下の記号を使って、文字または文字クラスの繰り返しとマッチします。ここでは文字または文字クラスをxと書きます。

x*	0回以上の繰り返し
x+	1回以上の繰り返し。xx*と同じ
x?	0回か1回
x{5}	5回繰り返し。xxxxxと同じ
x{3,}	3回以上繰り返し。xxx+と同じ
x{3,5}	3回以上5回以下繰り返し。xxxx?x?と同じ

6

グループと選択

文字列を繰り返すときは( )を使ってグループ化します。

su(mo)+	sumo, sumomo, sumomomoなどにマッチする

いくつかのパターンのどれかにマッチさせるときは|を使います。

love|kiss	loveかkissにマッチする
stud(y|ies)	studyかstudiesにマッチする
su(mi|mo){2,3}	sumimi, sumimo, sumomi, sumomo, sumimimi, sumimimo, sumimomi, sumomimi, sumomomi,sumomimo, sumimomo, sumomomoのいずれかにマッチする

正規表現による検索

- Gene indexを含む文字列を抽出してみましょう.

```
Query= gi|13507742|ref|NP_109691.1| DNA gyrase
```

Query= と **ref** ではさまれた連続した文字列を含む行を抽出するには

「.」（**任意の文字**） と 「**+**」（**１文字以上の連続文字**）

を使って以下のようにします

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= .+ref/) {
       print $a;
  }
}
```

スペース

カッコを使った記憶

- マッチ演算子のパターンの中で括弧（ ）を使うと，その括弧で囲まれた部分が，$1, $2,...という**特殊変数**に格納されます．

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print $1;
  }
}
```

改行

- 改行されるように，**”¥n”** を入れます

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print "¥n",$1;
  }
}
```

- BLAST検索の結果，ヒットしたタンパク質の情報

（例えば

>ref|NP_072866.1| topoisomerase IV, subunit A）

を含む行を抽出し，タブ区切りで表示してみましょう

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print "¥n",$1;
  }
  if ($a =~ />ref/) {
       print "¥t",$a;
  }
}
```

- ヒットしたタンパク質情報のref番号だけを抽出してみましょう
  >ref|NP_072866.1| topoisomerase IV, subunit A）

- "|"ではさまれた文字列を取り出したいのですが，以下の表現ではうまくいきません

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print "¥n",$1;
  }
  if ($a =~ />ref|.+|/) {
       print "¥t",$a;
  }
}
```

「|」
Shiftを押しながら

- ”|”は正規表現で使用する特殊な文字であるため，別の意味になってしまうからです
- ここで使う ”|” が正規表現でないことを示すために，¥（バックスラッシュ）を前につけます

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
      print "¥n",$1;
  }
  if ($a =~ />ref¥|.+¥|/) {
      print "¥t",$a;
  }
}
```

- 括弧を使って，ref番号だけを抽出してみましょう

  >ref|NP_072866.1| topoisomerase IV, subunit A

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print "¥n",$1;
  }
  if ($a =~ />ref¥|(.+)¥|/) {
       print "¥t",$1;
  }
}
```

質問配列と**ヒットした配列**のアクセション番号を抽出できるようになりました

QueryのGene Index	ヒットしたタンパク質のref番号	
gi|13507740|	NP_072661.1	
gi|13507741|	NP_072662.1	
gi|13507742|	NP_072663.1	NP_072865.1
gi|13507743|	NP_072664.1	NP_072866.1
gi|13507744|	NP_072665.1	
gi|13507745|	NP_072666.1	
gi|13507746|	NP_072667.1	
gi|13507747|	NP_072668.1	NP_072998.1
gi|13507748|	NP_072669.1	
.		
.		
.		

- **最も相同性の高い配列**の情報だけを表示するようにしてみましょう
- 新たに $b という変数を使い、これが **0 か 1 かを指標**にします
- 質問配列（Query= の行）を見つけたら **$b = 1** にします
- その後の最初に出てくるヒット配列（>ref の行）を見つけたら、番号を抽出して **$b = 0** に戻します（次の質問配列を見つけるまで抽出しません）

```
#! /usr/local/bin/perl
while ($a = <STDIN>) {
  if ($a =~ /Query= (.+)ref/) {
       print "¥n",$1;
       $b = 1;
  }
  if ($a =~ />ref¥|(.+)¥|/ && $b == 1) {
       print "¥t",$1;
       $b = 0;
  }
}
```

質問配列と**最も相同性の高いヒット配列**のアクセション番号を抽出できるようになりました

QueryのGene Index	ヒットしたタンパク質のref番号
gi|13507740|	NP_072661.1
gi|13507741|	NP_072662.1
gi|13507742|	NP_072663.1
gi|13507743|	NP_072664.1
gi|13507744|	NP_072665.1
gi|13507745|	NP_072666.1
gi|13507749|	
gi|13507750|	
gi|13507751|	
gi|13507752|	
gi|13507753|	NP_072670.1
gi|13507754|	NP_072671.1
.	
.	
.	

＜課題１＞

- 質問配列のGene Indexのうち，**数字の部分だけ**を取り出して，以下のような出力結果になるプログラムを作成してください
  （44ページ参照）

```
13507740    NP_072661.1
13507741    NP_072662.1
13507742    NP_072663.1
13507743    NP_072664.1
13507744    NP_072665.1
13507745    NP_072666.1
13507746    NP_072667.1
13507747    NP_072668.1
13507748    NP_072669.1
            .
            .
```

＜課題２＞

- **E-valueの値**を抽出して，以下のような出力結果になるプログラムを作成してください

質問配列のGene Index	ヒットしたタンパク質のref番号	E-value
13507740	NP_072661.1	e-148
13507741	NP_072662.1	e-125
13507742	NP_072663.1	0.0
13507743	NP_072664.1	0.0
13507744	NP_072665.1	0.0
13507745	NP_072666.1	1e-077
	.	
	.	

課題１と同じファイル名にならないように，**output2.txt**というファイル名で結果を出力してください

- 出力したテキストファイル（**output1.txt** と **output2.txt**）を、メールに添付して提出してください
- 送付先は「kenro@hosei.ac.jp」です
- メールの件名は「**Perl課題**」にしてください
- メール本文に、以下のように「氏名」「所属」「学生証番号」「本日の講義の感想」を記載してください

氏名：○ ○　○ ○

所属：× × × ×専攻　△ △ △ △研究室

学生証番号：□ □ □ □ □

講義の感想：