アライメント

スコア関数（ギャップを含む）を最大にするような文字の対応つけを探す

1. ギャップなしアライメント
2. ギャップありアライメント

ギャップなし
```
AFDC
AEEC
```
ギャップあり
```
AFAED-C
A--EEGC
```

a. グローバルアライメント (ClustalW)
b. ローカルアライメント (FASTA, BLAST)

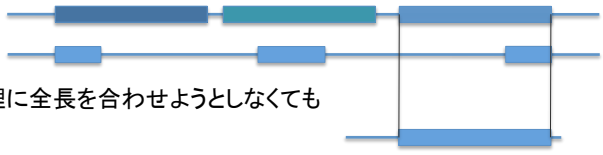
```
ACDEFGHKLM
AFGHKKL
```
➡
```
ACDEFGHK-LM
A---FGHKKL-
```
グローバル
```
FGHK-L
FGHKKL
```
ローカル

動的計画法というアルゴリズムで解く。
そのイメージをつかむためには**ドットマトリックス法**が有効

グローバルアラインメントとローカルアラインメント

配列の全長が一致すると見なされる場合、グローバルアラインメント

たとえばマルチドメインタンパク質（幾つかの要素から構成）と
シングルドメインのアラインメントで



無理に全長を合わせようとしなくても

最も一致度の高い一部分にマッチすれば良い：　ローカルアラインメント

```
ACDEFGHK-LM
A---FGHKKL-
```
無理に全長をあわせて
配列相同性 6/11 = ~55%

```
FGHK-L
FGHKKL
```
一致する部分のみで比べて
配列相同性 5/6 = ~83%

ペアワイズアラインメントのアルゴリズム

ペアワイズアラインメントをおこなうには、さまざまなアラインメントのパターンをすべて作って最もスコアの高いものを選択すればよい、

```
ACDEF      ACDEF-     ACDEF--    ........   ACDEF----
ADEFG      -ADEFG     --ADEFG               ----ADEFG

           ACDEF-     ACDEF--    ........   ACDEF----
           A-DEFG     A--DEFG               A----DEFG

           ACDEF-
           AD-EFG     ..................................

..................................................................
..................................................................
..................................................................
```

単純にすべての組み合わせを考えようとすると、配列長の和の階乗のオーダー
たとえば100! = 10^{158}のオーダーなのでどんなに計算機が速くても無理！
実際には、極端に重なりの少ないものを除いたりして計算量を減らすことができるが、それでも配列長がすこし長くなるとすぐに計算は困難になる

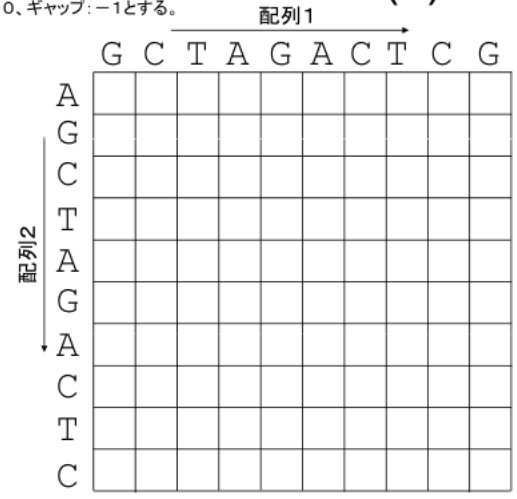
計算の仕方の工夫により実現可能な計算量にすることができる！
動的計画法　<ー　これの雰囲気をつかむためにまずドットマトリクスをみてみよう

ドットマトリックス　：　例1　(1)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

1：GCTAGACTCG
2：AGCTAGACTC

(1) 配列1、配列2を横と縦に並べる



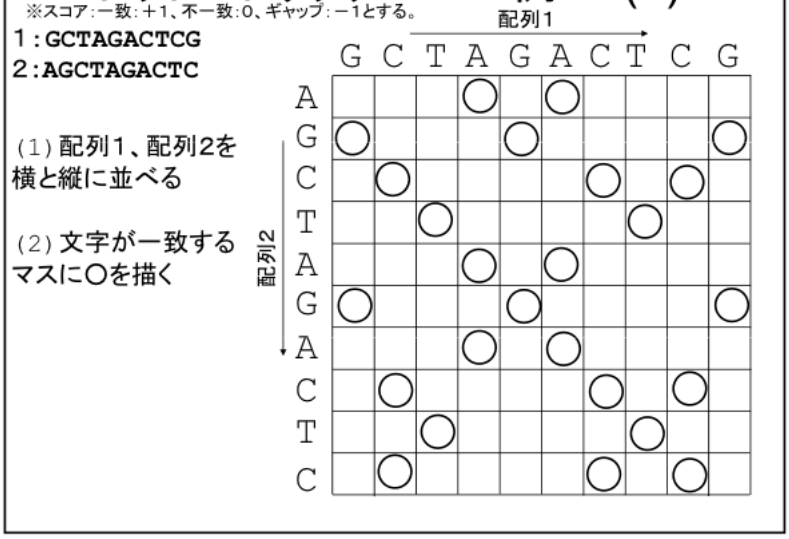
ドットマトリックス　：　例１　(2)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

1：GCTAGACTCG
2：AGCTAGACTC

(1) 配列1、配列2を横と縦に並べる

(2) 文字が一致するマスに〇を描く



ドットマトリックス　：　例１　(3)

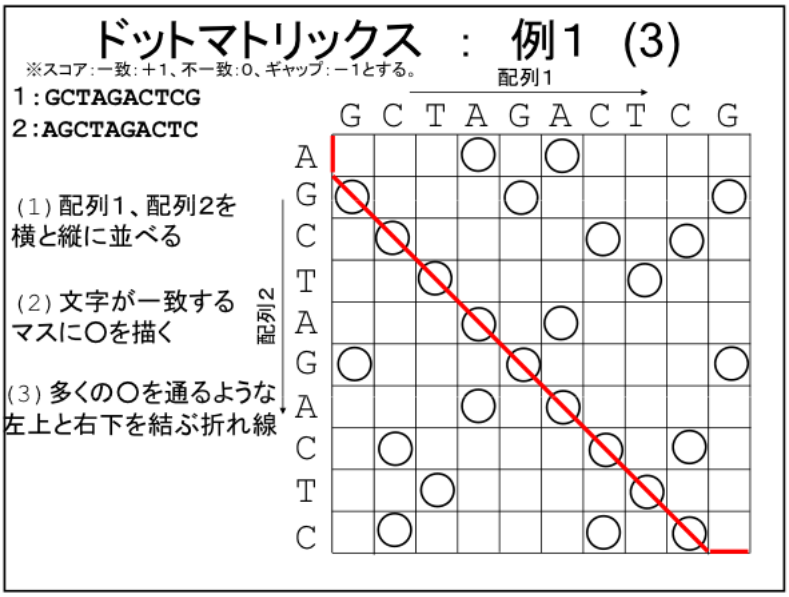
※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

1：GCTAGACTCG
2：AGCTAGACTC

(1) 配列1、配列2を横と縦に並べる

(2) 文字が一致するマスに〇を描く

(3) 多くの〇を通るような左上と右下を結ぶ折れ線



ドットマトリックス　：　例1　(4)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

1：GCTAGACTCG
2：AGCTAGACTC

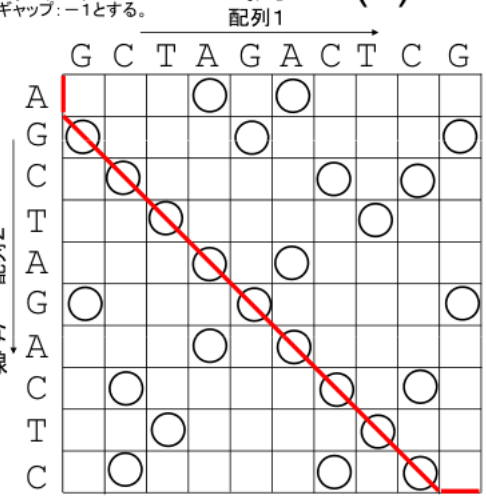
(1) 配列1、配列2を横と縦に並べる

(2) 文字が一致するマスに○を描く

(3) 多くの○を通るような左上と右下を結ぶ折れ線

(4) アライメント

1：-GCTAGACTCG
　　*********
2：AGCTAGACTC-

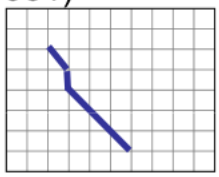


スコア：一致(＋1)×9＋不一致(0)×0＋ギャップ(－1)×2＝7

ローカルアライメントの解法
(Smith & Waterman,1981)

(0)準備

格子の端のスコアを0に設定



(1)前向きステップ

$$F(i,j)=\max\begin{cases} F(i+1,j+1)+s(x_i,y_j) & \text{対角}(d) \\ F(i+1,j)+Gap & \text{鉛直}(v) \\ F(i,j+1)+Gap & \text{水平}(h) \\ 0 & \text{終結}(0) \end{cases}$$

(2)後ろ向きステップ

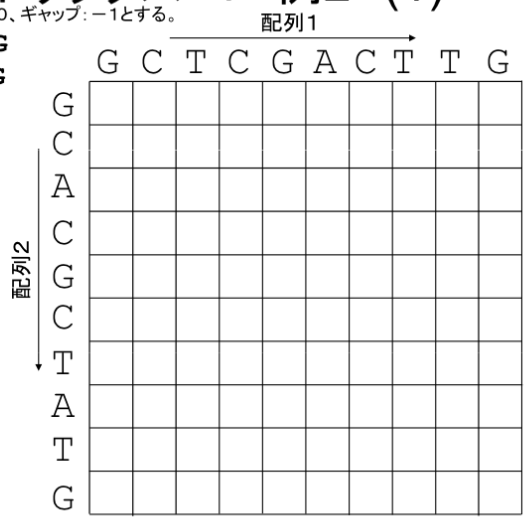
最大のスコアのノードを探し、そのノードを起点にして辿る。パス'0'が現れたら終了

ドットマトリックス　：　例2　(1)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

配列１：**GCTCGACTTG**
配列２：**GCACGCTATG**

(1)配列１、配列2を
横と縦に並べる



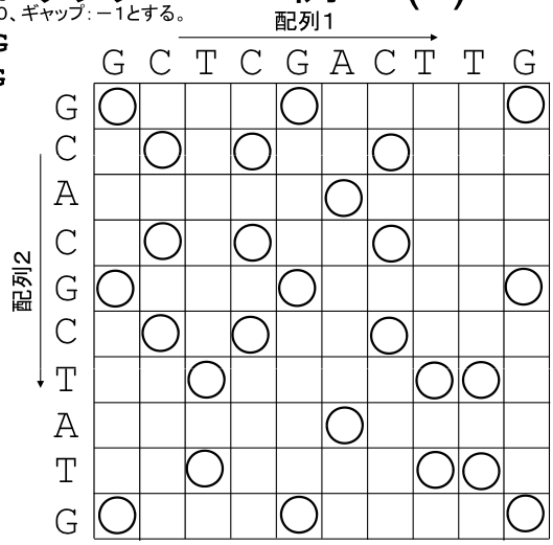
ドットマトリックス　：　例2　(2)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

配列１：**GCTCGACTTG**
配列２：**GCACGCTATG**

(1)配列１、配列2を
横と縦に並べる

(2)文字が一致する
マスに○を描く



ドットマトリックス　：　例2　(3)

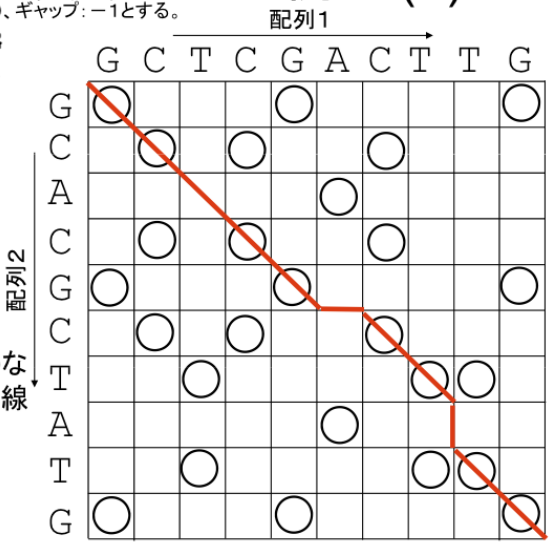
※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

配列1：GCTCGACTTG
配列2：GCACGCTATG

(1) 配列1、配列2を横と縦に並べる

(2) 文字が一致するマスに○を描く

(3) 多くの○を通るような左上と右下を結ぶ折れ線



ドットマトリックス　：　例2　(4)

※スコア：一致：＋1、不一致：0、ギャップ：－1とする。

配列1：GCTCGACTTG
配列2：GCACGCTATG

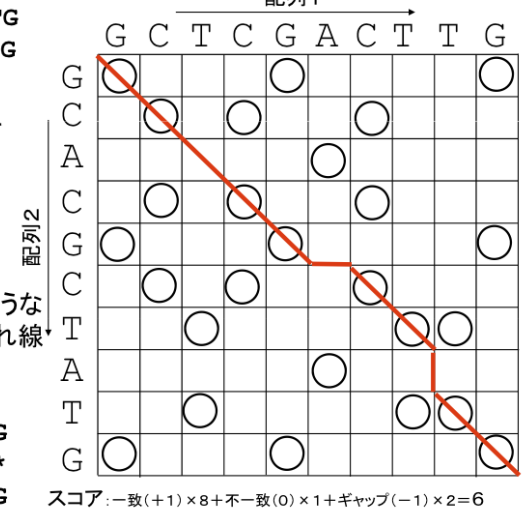
(1) 配列1、配列2を横と縦に並べる

(2) 文字が一致するマスに○を描く

(3) 多くの○を通るような左上と右下を結ぶ折れ線

(4) アライメント

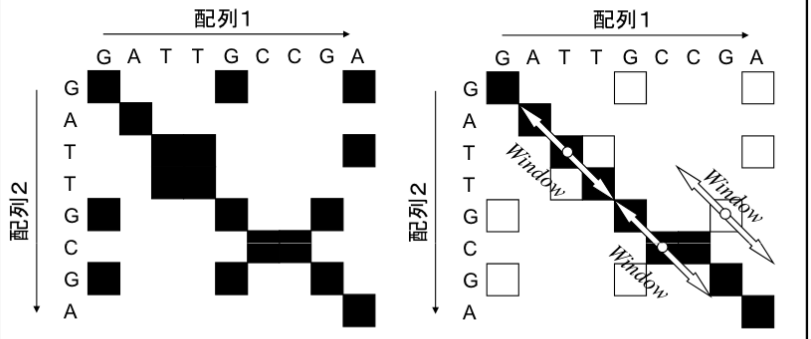
1：GCTCGACT-TG
　　** ** ** **
2：GCACG-CTATG



スコア：一致(＋1)×8＋不一致(0)×1＋ギャップ(－1)×2＝6

対角上の平均化によるスムージング

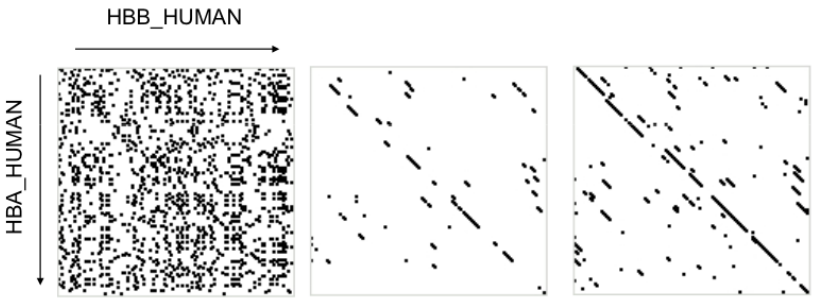
配列１：GATTGCCGA
配列２：GATTGCGA



（1）単純に一致している座標を黒く塗る
Window=1, *Threshold*=1に相当

（2）長さ*Window*の連続したペアが比較し、
一致度が*Threshold*以上であれば黒く塗る
Window=3, *Threshold*=2の場合

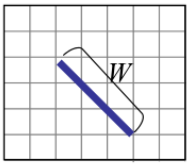
ドットマトリックスの例



Matrix=ID,W=1,T=1

Matrix=ID,W=5,T=3

Matrix=BLOSUM62,
W=7,T=10

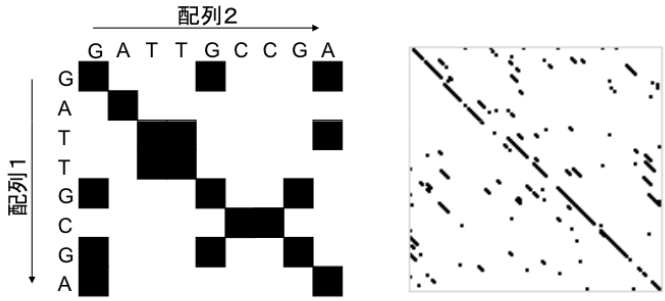


文字が一致しているペアを黒く塗る
→ 長さW のwordの総スコアがT 以上なら中心を黒く塗る

スコアは、最も簡単には一致・不一致スコア
対数オッズスコアを使うとより高感度になる

ドットマトリックス法の特徴

- アルゴリズムが平易
- 非常に長い配列の比較にも対応
- 部分一致、繰り返しなど特殊なケースにも対応できる。
- あくまでグラフィカルな対応なので、具体的な文字列対応（アライメント）は与えない。

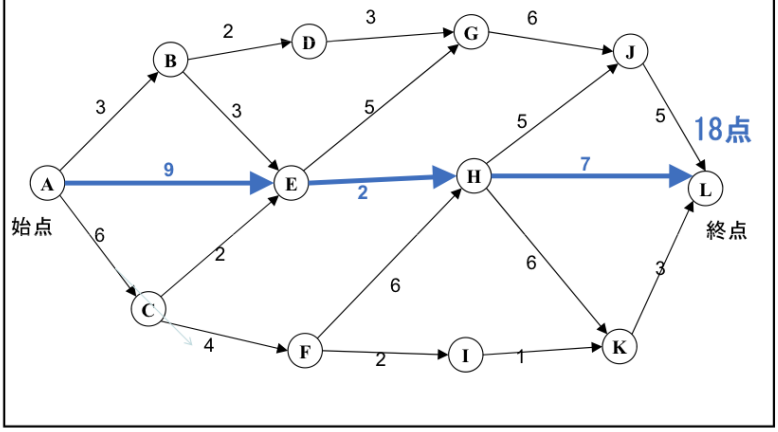


動的計画法によるアライメント

- アライメント問題は、有向グラフの最適経路問題と等価
- 有向グラフの最適経路問題は動的計画法（Dynamic Programming)と呼ばれるアルゴリズムで解ける。
- *O(NM)*の計算量（文字列長の積に比例）

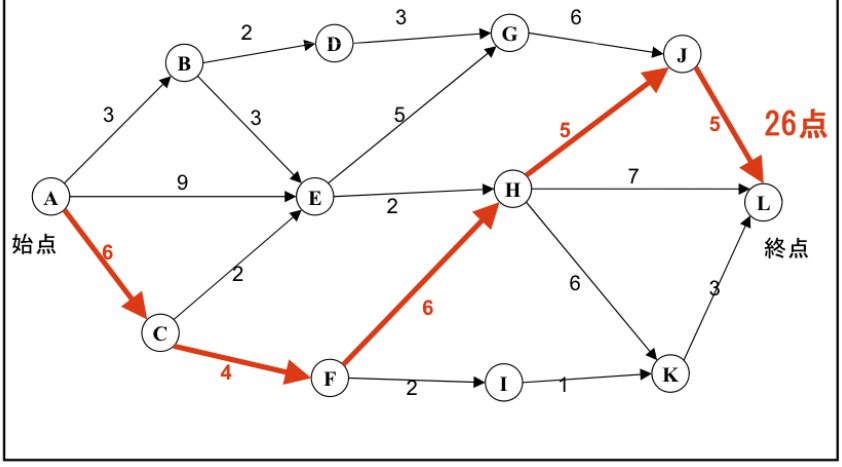
最適経路問題

始点Aから終点Lにいたるエッジの得点の合計が
最大となる経路を探す



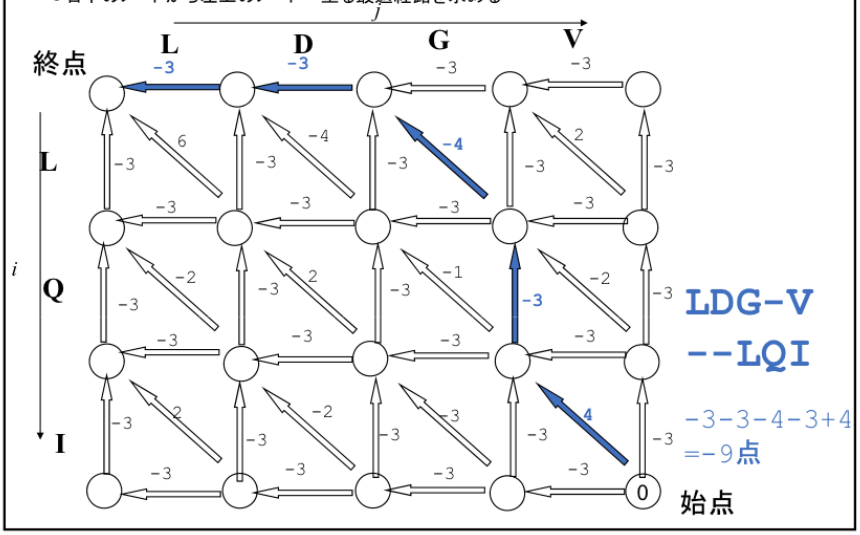
最適経路問題

始点Aから終点Lにいたるエッジの得点の合計が
最大となる経路を探す



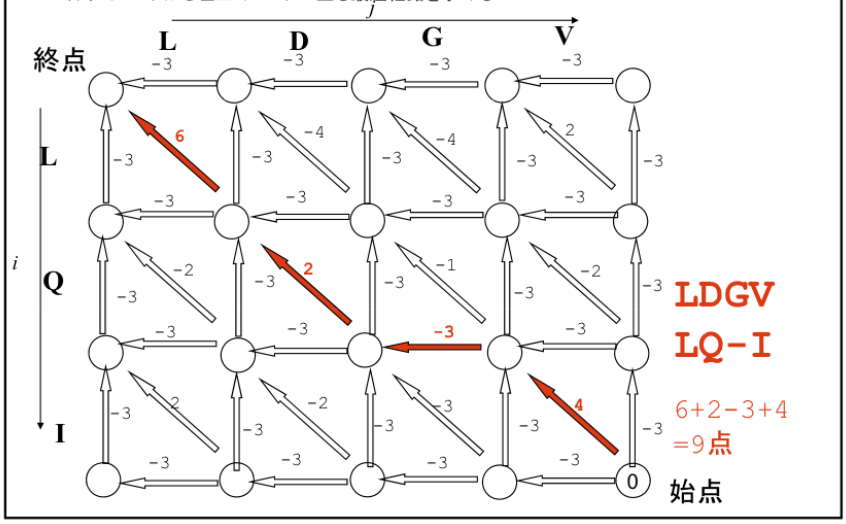
アライメントを最適経路問題として考える

- 鉛直、水平に比較したい文字列を並べる
- 対角線のエッジには一致スコア、鉛直水平のエッジにはギャップスコアを書き込む
- 右下のノードから左上のノードへ至る最適経路を求める



アライメントを最適経路問題として考える

- 鉛直、水平に比較したい文字列を並べる
- 対角線のエッジには一致スコア、鉛直水平のエッジにはギャップスコアを書き込む
- 右下のノードから左上のノードへ至る最適経路を求める

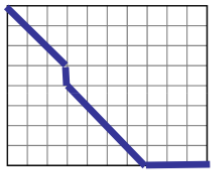


グローバル・アライメントの解法
(Needleman & Wunsh,1970)

(0)準備

右端の列、下端の行の格子点のスコアを0に設定

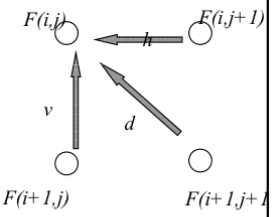
始点



終点

(1)前向きステップ

$$F(i,j)=\max\begin{cases} F(i+1,j+1)+S(x_i,y_j) & \text{対角}(d) \\ F(i+1,j)+Gap & \text{鉛直}(v) \\ F(i,j+1)+Gap & \text{水平}(h) \end{cases}$$

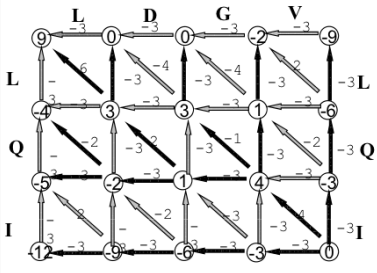


(2)後ろ向きステップ

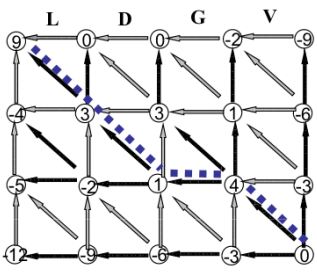
始点を起点にして辿る。終点に到着したら終了。

動的計画法の手続き

(1)前向き（Forward）



(2)後ろ向き（TraceBack）



O(NM)

LDGV
LQ-I

グローバルとローカルの格子上の違い

ACDEFGHKLM
AFGHKKL

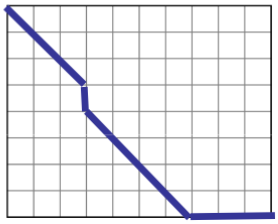
➡

ACDEFGHK-LM
A---FGHKKL-

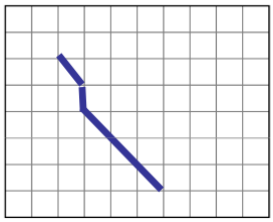
グローバル

FGHK-L
FGHKKL

ローカル

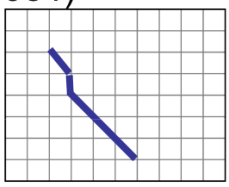


グローバル



ローカル

ローカルアライメントの解法 (Smith & Waterman,1981)



(0)準備

格子の端のスコアを0に設定

(1)前向きステップ

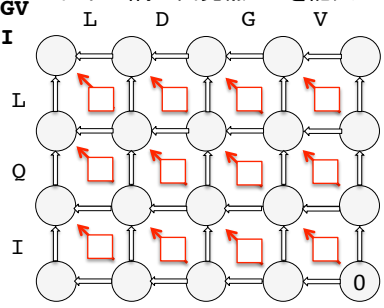
$$F(i,j)=\max\begin{cases} F(i+1,j+1)+s(x_i,y_j) & \text{対角}(d) \\ F(i+1,j)+Gap & \text{鉛直}(v) \\ F(i,j+1)+Gap & \text{水平}(h) \\ 0 & \text{終結}(0) \end{cases}$$

(2)後ろ向きステップ

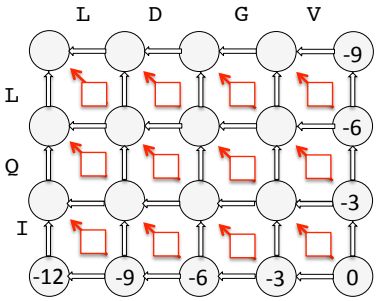
最大のスコアのノードを探し、そのノードを起点にして辿る。パス' 0'が現れたら終了

配列1：**LDGV**
配列2：**LQI**

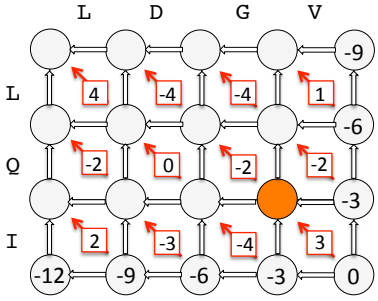
右下の隅が出発点ー0を記入



GAPペナルティを一律に-3とすると黒枠矢印
を進むごとに-3加算、右端と下端は一通りしかない



スコア行列（BLOSUM62）を参照して、赤枠を埋める
たとえば、左上隅、はS(L,L)で4

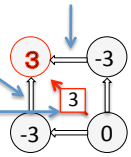


右下隅から、空の〇を埋めてゆく。それぞれ3通りの経路があり、そのうちスコアが最大になるものを選ぶ。

この経路では元の〇のスコア-3に
黒矢印を通ることに寄るスコア-3を加えて
-6 となる

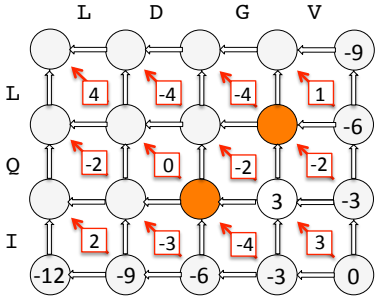
この経路でも同様に
-6となる。

斜めに上がれば、元
のスコア0に赤枠の
スコア3を加えスコア
3となる。

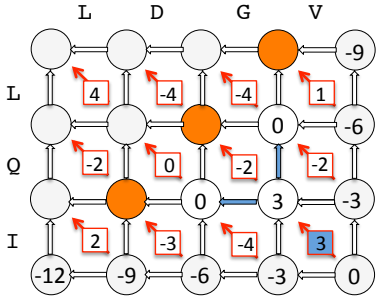


そこで、左上の〇のスコアを3とし、斜めの経路にマークをつける

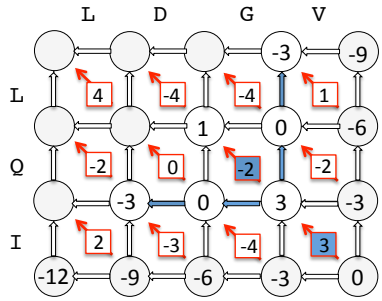
この時点で新たに2カ所の値が求められる。

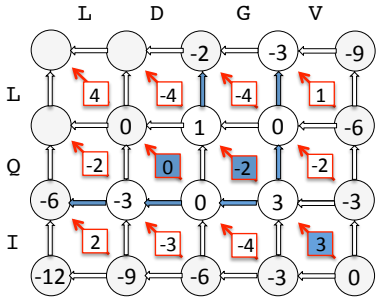


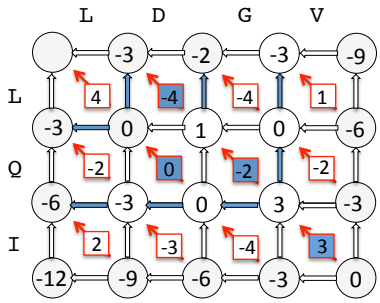
それぞれの経路を記録しながらスコアを求めてゆく



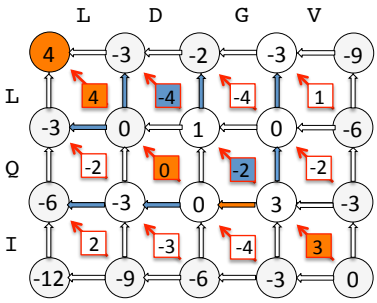
それぞれの経路を記録しながらスコアを求めてゆく



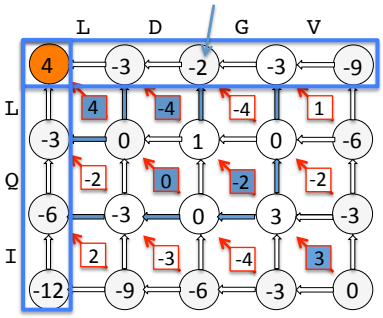




出発点から最大値の左上隅に至る経路が判れば、
ドットマトリクスの時と同様にアラインメントが求まる。



上端、または左端で最大の数値を捜す、この場合は
左上隅の4だが、たとえばここになる場合もありうる。



すなわち横向きに移動している G に対応する箇所
がない(GAP)であると考えれば以下のアラインメント
が求まる

配列1：**LDGV**
配列2：**LQ-I**

配列相同性検索

→クエリ配列を配列データベースと比較、相同な配列を探す

- 機能未知遺伝子の機能予測（アノテーション）
  機能既知の配列との類似→機能の類似を示唆
- 立体構造予測
  構造既知の配列との類似→構造の類似を示唆
- 遺伝子発見
  既知遺伝子と類似している領域の発見→遺伝子の存在を示唆

E-value

配列相同性のもう一つの指標

後で用いる配列相同性検索プログラム：BLASTで用いられる

ログオッズスコアの和： アラインメントが長いほど高くなる ー＞補正

ランダムな配列の比較で、偶然にスコアSが生じる可能性

0~1で、低いほど、相同性が高いと考えられる

ひとつの目安として、0.0001 = 10^{-4} より小さければホモロジーが有ると考える

BLASTの出力では、指数表記で表されるので注意

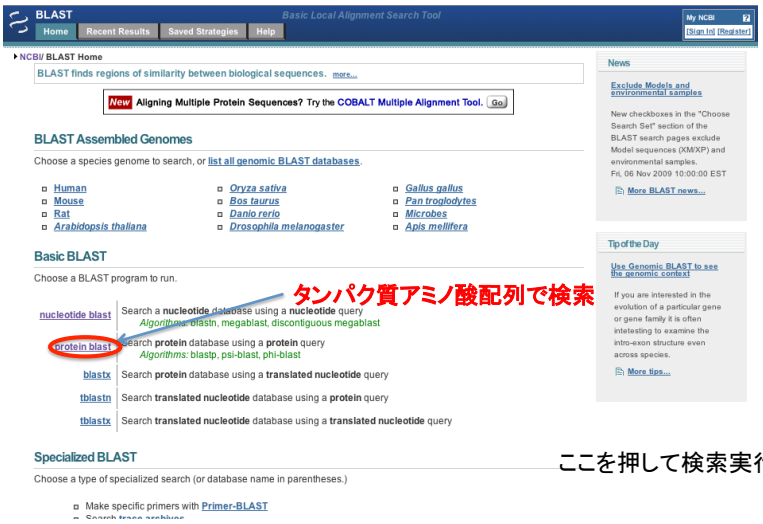
例えば、 10^{-4} は 1.0e-4 と表記される

0.24 は 2.4e-1, 0.000000000098 は、9.8e-11 と表記される

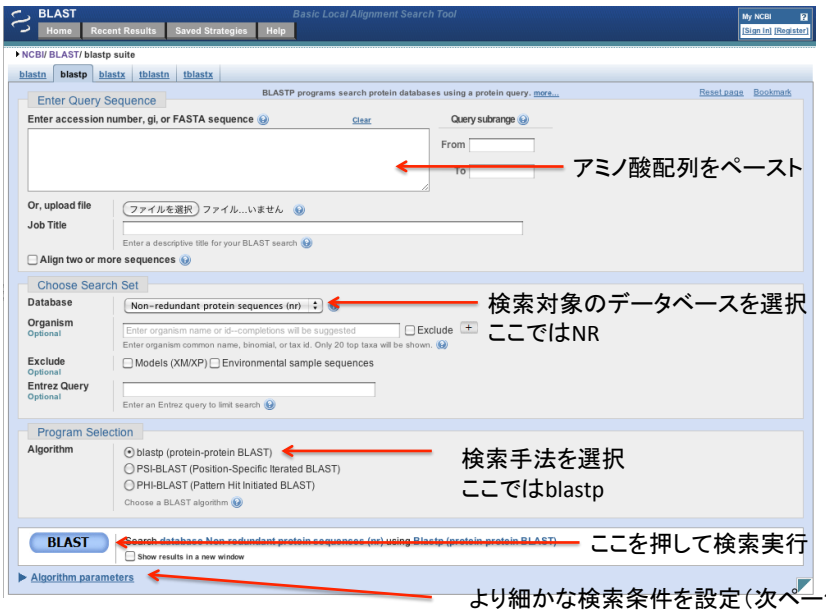
BLASTホームページ

http://blast.ncbi.nlm.nih.gov/Blast.cgi

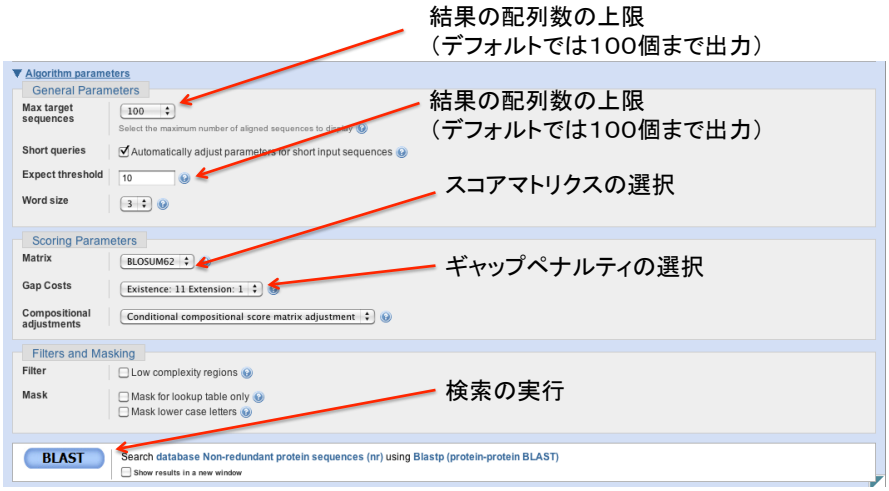
Google等で、「NCBI BLAST」で検索



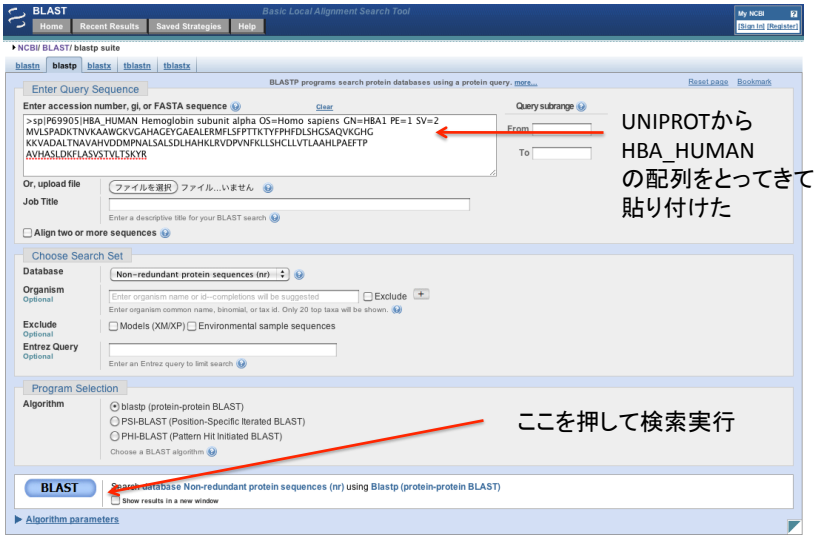
タンパク質アミノ酸配列に対するBLAST検索ページ



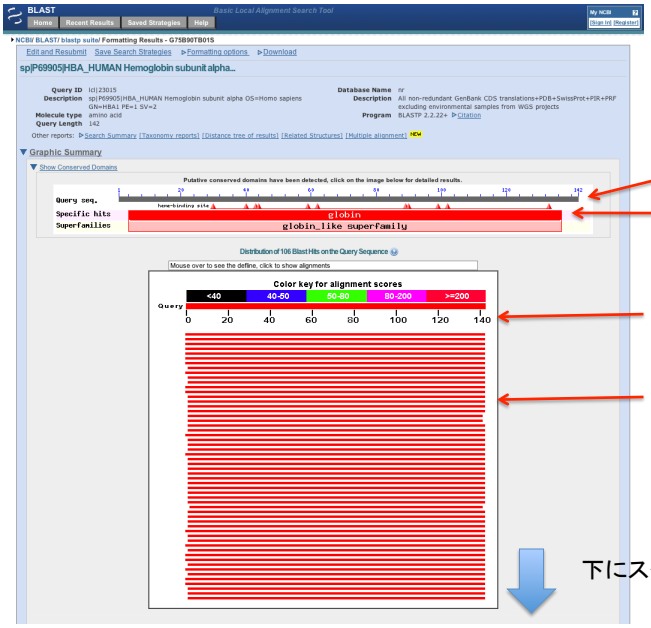
細かい条件設定のページ（前ページのAlgorithm parametersをクリックした場合）



タンパク質アミノ酸配列に対するBLAST検索ページ



BLAST検索結果

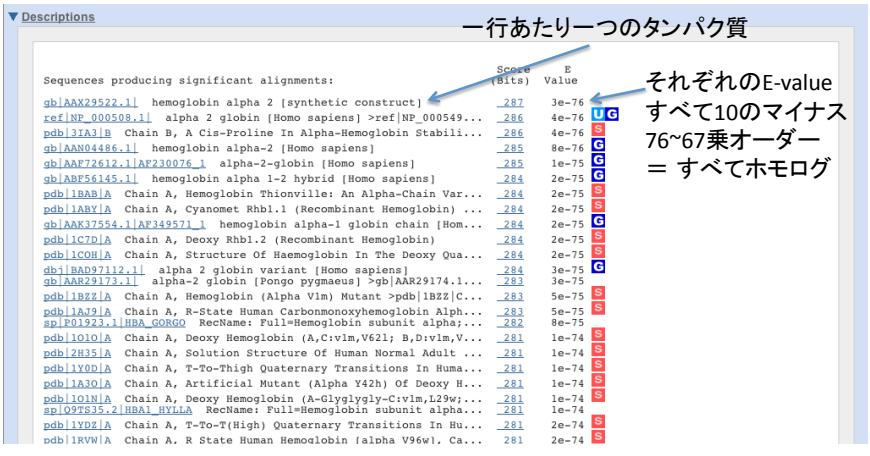


クェリー配列長

の領域における
予測機能

クェリー配列(~140)
の全長にわたって
高い相同性の配列が
赤い横棒の数だけある
ここでは上限の100本

下にスクロール(次ページ)



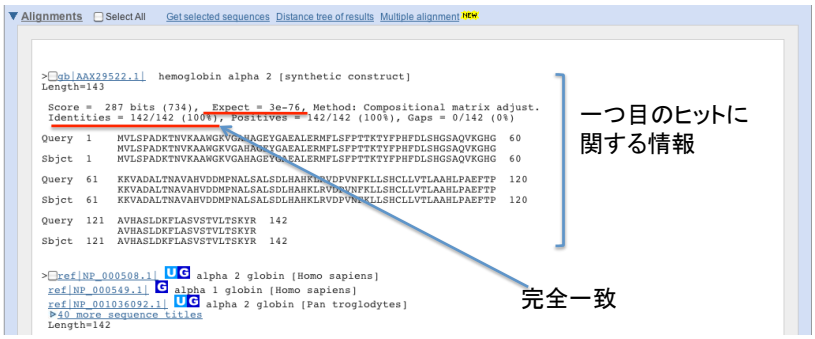
一行あたり一つのタンパク質

それぞれのE-value
すべて10のマイナス
76~67乗オーダー
= すべてホモログ

下にスクロール(次ページ)

クエリーとヒットのアラインメント

1本目



最後