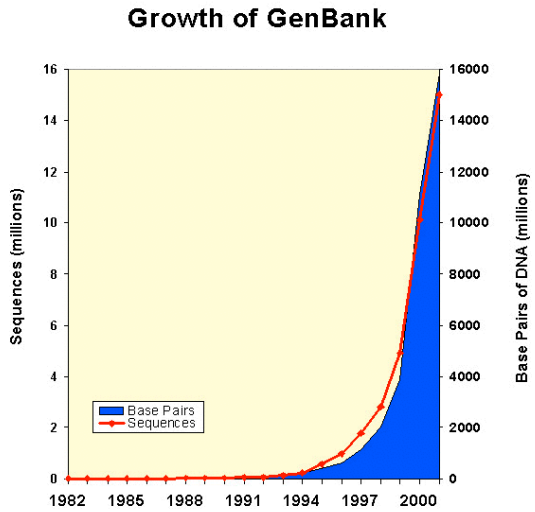
Bases de données

Bases de données de séquences : beaucoup d'information.

Exemple : GenBank

- 17 milliards de nucléotides
- croissance exponentielle (taille doublée tous les 14 mois)



Recherche dans une BD

On a une séquence : trouver son occurrence dans la BD.

Deux problèmes :
- occurrences exactes
- occurences similaires

Problème algorithmique :
On a une séquence (courte) P et une séquence (longue) T :
trouver les occurrences [exactes ou similaires] de P dans T.

On a vu un algorithme [Alignements, page 14] qui prend temps $O(nm)$ pour $|T| = m$, $|P| = n$.

Est-ce qu'il y a des meilleures solutions ? — Oui !

Occurrences exactes

Algorithme naïf avec $O(nm)$ temps de calcul

```
1  pour i ← 1, ... m − n + 1
2       si  P[1] = T[i] alors
3       soit cnt ← 1
4       pour j ← 2, ..., n
5            si  P[j] = T[i + j − 1] alors cnt ← cnt + 1
6            sinon  quitter le boucle j
7       fin boucle j
8       si cnt = n alors occurrence en position i
9  fin boucle i
```

Comment accélérer ?

1. calcul malin de décalage

2. réduction de nombres de comparaisons

exemple :

```
T: xabxyabxyabxz
P: abxyabxz
```

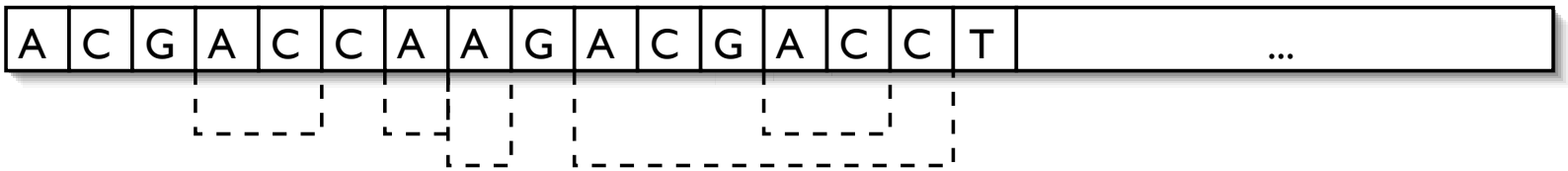
Comparaison de $T[2..9]$ à $P[1..8]$: mismatch dans la dernière position.

1. prochaine comparaison devrait être en position 6 de T
2. éviter la comparaison de $T[6..8]$ à $P[1..3]$ de nouveau

Algorithme «Z»

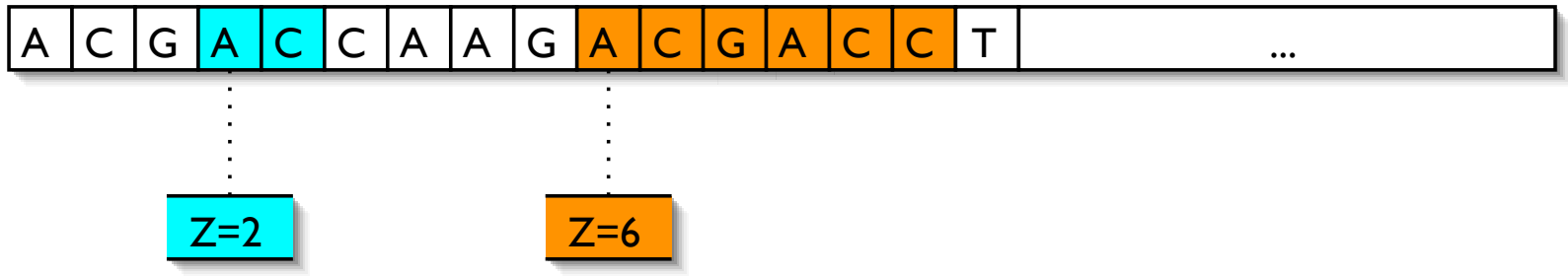
Soit S une séquence.

Sous-mots maximaux qui correspond à un préfixe : boîtes Z



Algorithme «Z» 2

Déf. Pour $1 < i \leq |S|$, $Z_i(S)$ est la longueur du sous-mot qui commence en position i, identique à un préfixe de S. S'il n'existe pas un tel préfixe, alors $Z_i(S) = 0$.



Algorithme «Z» 3

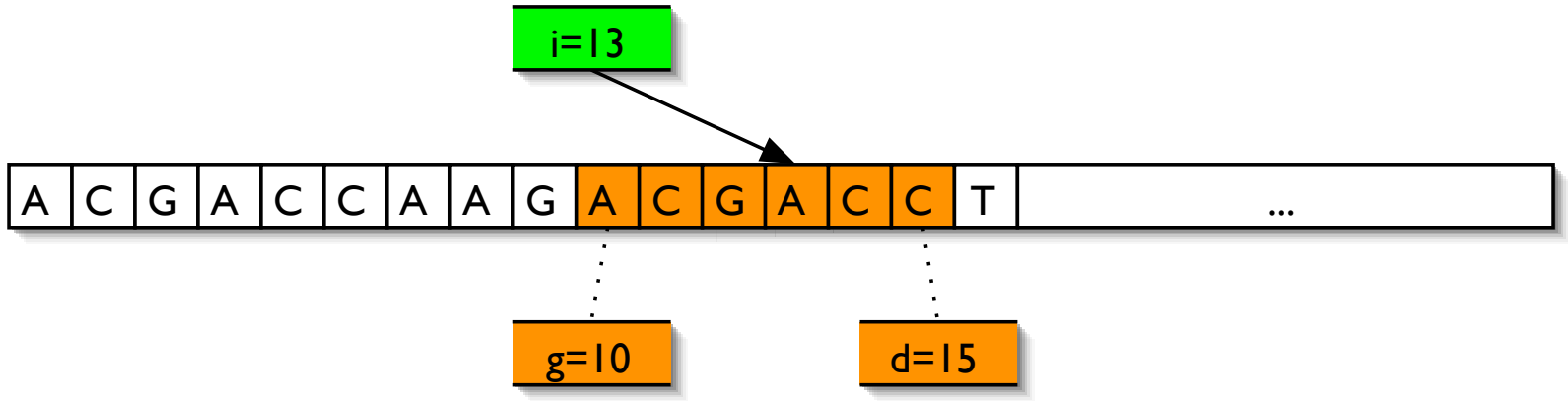
1. Concatenation de P et T : $S = P\#T$
2. Calculer $Z_i(S)$.
3. Si $Z_i(S) = n$ pour $i > n + 1$, alors on a une occurrence $P = T[i - (n + 1)..(i - 2)]$.

Mais comment est-ce qu'on peut calculer Z_i ?

Algorithme «Z» 4

Déf : d_i est la position maximale du côté droit d'une boîte Z qui commence dans $S[1..i]$. g_i est le côté gauche de la même* boîte Z.

$$d_i = \max\{j + Z_j - 1 : 1 < j \leq i\}.$$



* (s'il y en a plusieurs, alors choix arbitraire)

Algorithme «Z» 5

Calcul de Z_i, g_i et d_i.

Initialisation : Calculer Z_2, si $Z_2 > 0$, alors $d_2 = Z_2 + 1$, $g_2 = 2$, sinon $d_2 = g_2 = 1$.

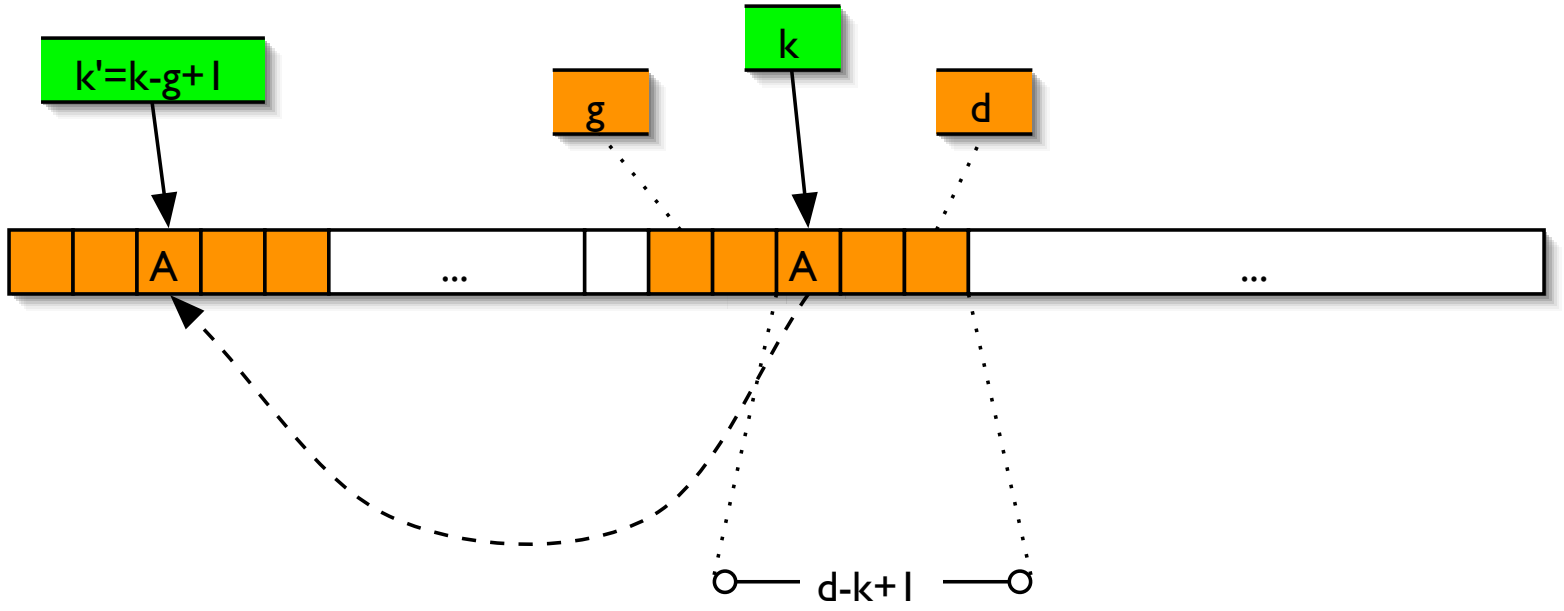
Récurrence : si on a Z_i, g_i, d_i pour $1 < i \leq k - 1$, on peut calculer les valeurs pour k d'une façon très effective.

Algorithme «Z» 6

Récurrence - cas 1. Si $k > d_{k-1}$, alors il faut trouver une nouvelle boîte Z qui commence en position k. Z_k est la longueur de cette boîte. Si $Z_k > 0$, alors $g_k = k$, $d_k = k + Z_k - 1$. Sinon, $g_k = d_k = k - 1$.

Algorithme «Z» 7

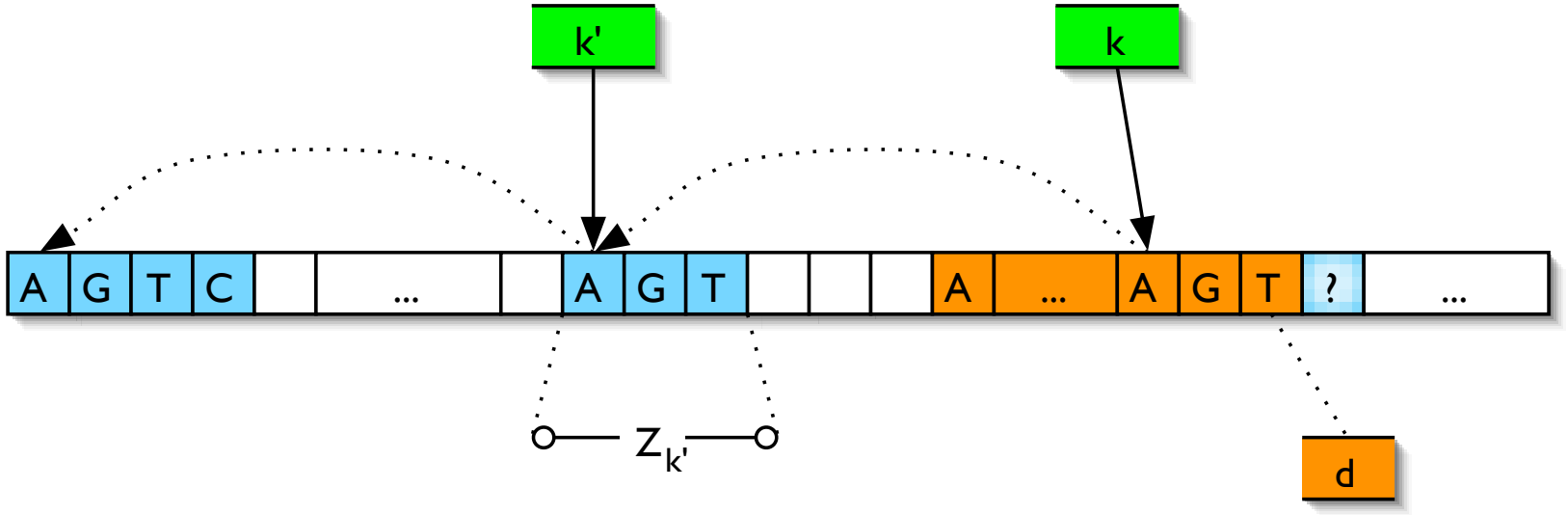
Récurrence - cas 2. $k \leq d_{k-1}$



Cas 2a. Si $Z_{k'} < d_{k-1} - k + 1$, alors $Z_k = Z_{k'}$, $d_k = d_{k-1}$, et $g_k = g_{k-1}$

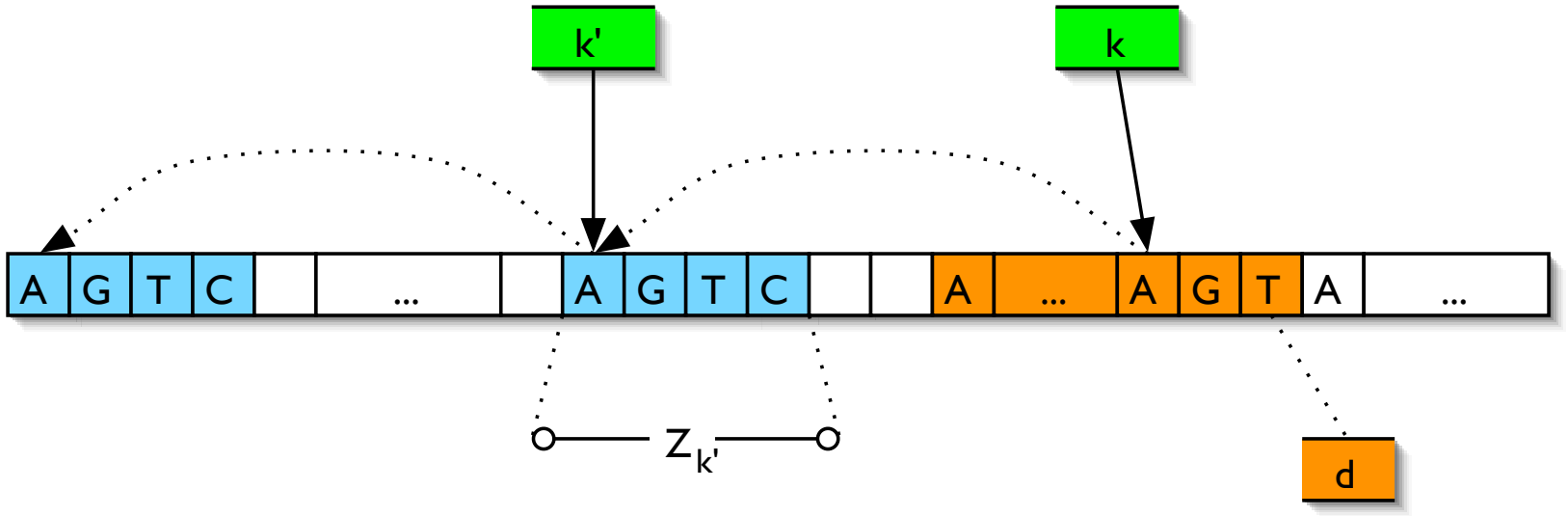
Algorithme «Z» 8

Cas 2b. Si $Z_{k'} = d_{k-1} - k + 1$, alors $S[k..d_{k-1}] = S[k'..k' + Z_{k'} - 1] = S[1..Z_{k'}]$. Il faut comparer les caractères $S[d_{k-1}+1], \ldots$ à $S[Z_{k'}+1], \ldots$ pour trouver le premier mismatch en position q. Alors $Z_k = q - k$, $d_k = q - 1$, $g_k = k$.



Algorithme «Z» 9

Cas 2c. Si $Z_{k'} > d_{k-1} - k + 1$, alors $S[k..d_{k-1}] = S[1..d_{k-1} - k + 1]$ comme en 2b, mais $S[d_{k-1} + 1] \neq S[k' + (d_{k-1} - k) + 1]$ et $S[k' + (d_{k-1} - k) + 1] = S[d_{k-1} - k + 2]$. Alors $Z_k = d_{k-1} - k + 1$, $d_k = d_{k-1}$, $g_k = g_{k-1}$.



Algorithme «Z» : temps de calcul

Thm. Les valeurs Z_i, d_i, g_i sont calculées dans un temps $O(|S|)$.

Preuve.

1. Nombre d'itérations : $k = 2, \ldots, |S|$

2. Nombre de comparaisons

- mismatch : 1 par itération (cas 1 ou 2b)
- match : $(d_k - d_{k-1})$ au plus dans chaque pas (cas 1 ou 2b)

$\Rightarrow d_2 + (d_3 - d_2) + \cdots + (d_{|S|} - d_{|S|-1}) = d_{|S|}$ au maximum, or $d_{|S|} \leq |S|$.

Algorithme de Boyer-Moore

Le meilleur algorithme pour recherche d'un mot dans un texte (p.e. logiciels de traitement de texte)

Temps de calcul linéaire (difficile à démontrer) mais temps «typique» sous-linéaire !

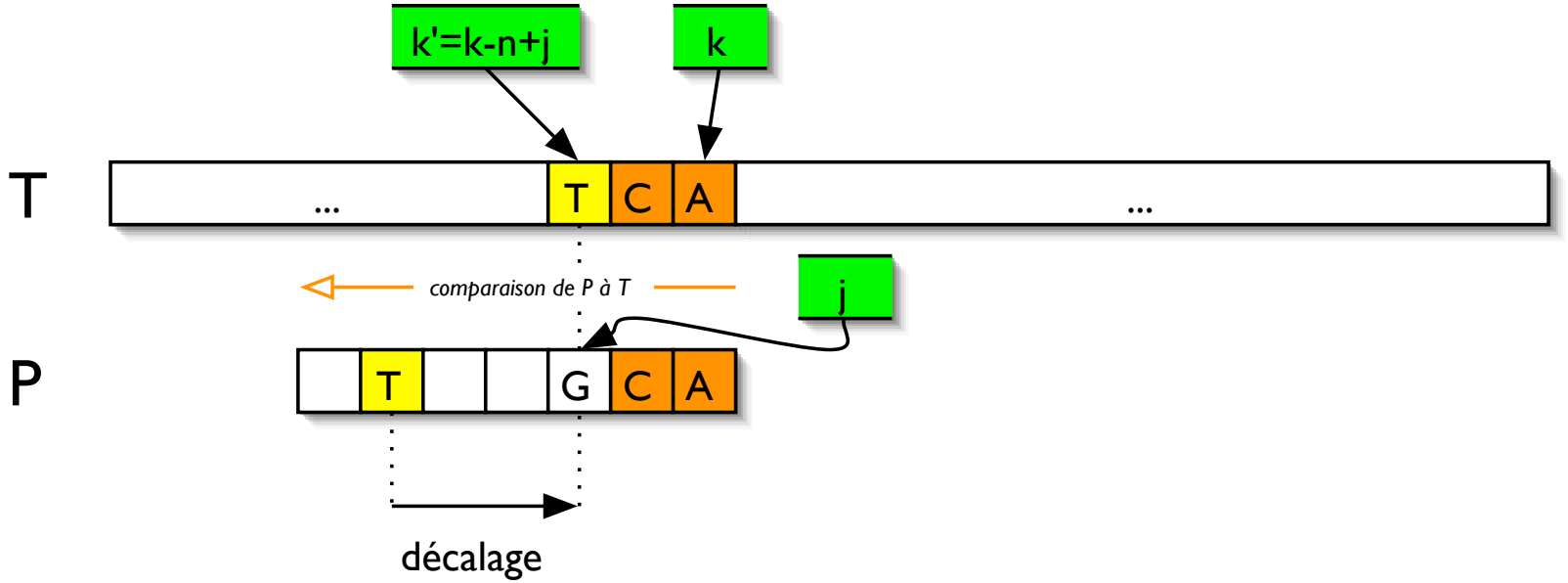
Comparaison de P de droite à gauche.

Boyer-Moore 2

```
 1  soit k ← n
 2  faire
 3      si  P[n] = T[k] alors
 4      soit cnt ← 1
 5      pour j ← n − 1, . . . , 1
 6          si  P[j] = T[k − n + j] alors cnt ← cnt + 1
 7          sinon  quitter le boucle j
 8      fin boucle j
 9      si cnt = n alors occurrence en position k
10      décalage : k ← k + d
11  tant que k ≤ m.
```

Deux idées pour le décalage en ligne 10 :
le mauvais caractère et le bon suffixe.

Le mauvais caractère



Déf. Pour chaque lettre c, soit $R(c)$ la position de sa dernière occurrence en P : $R(c) = \max\{j : P[j] = c\}$. Si c n'apparait pas dans P, $R(c) = 0$.

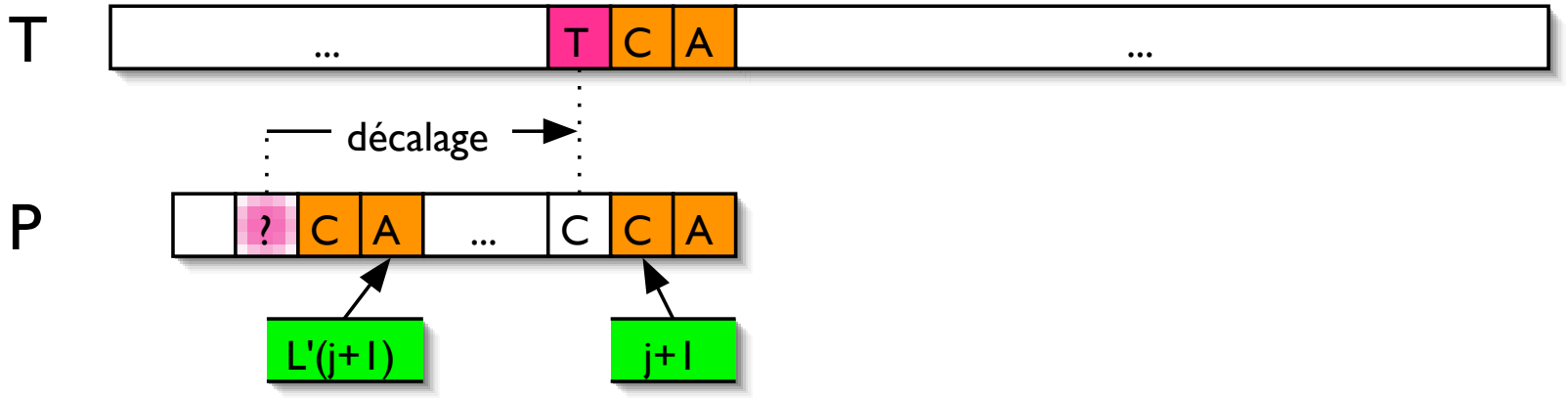
Décalage : si le premier mismatch est $T[k'] \neq P[j]$, alors décalage par $\max\left\{1, j - R(T[k'])\right\}$

Le bon suffixe — mismatch

Déf. Pour chaque $1 < j < n$, soit $\bar{j} = n - j$, $P_j = P[j..n]$, et

$$L'(j) = \max\{0\} \cup \left\{ i : i < n, \; P[i - \bar{j}..i] = P_j, P[i - \bar{j} - 1] \neq P[j - 1] \right\}.$$

$L'(j)$ est le côté droit de la dernière copie du suffixe $P_j = P[j..n]$ dans P telle que le caractère précédent n'est pas $P[j - 1]$.



Le bon suffixe — 2

Comment calculer les $L'(j)$?

1. Calculer les $Z_i(P^R)$ où P^R est l'image miroir de P : $Z_i(P^R)$ donne la longueur du sous-mot de P qui termine en position $n - i + 1$, identique à un suffixe de P.

2. $L'(j) = \max\left\{i : i < n, Z_{n-i+1}(P^R) = n - j + 1\right\}$

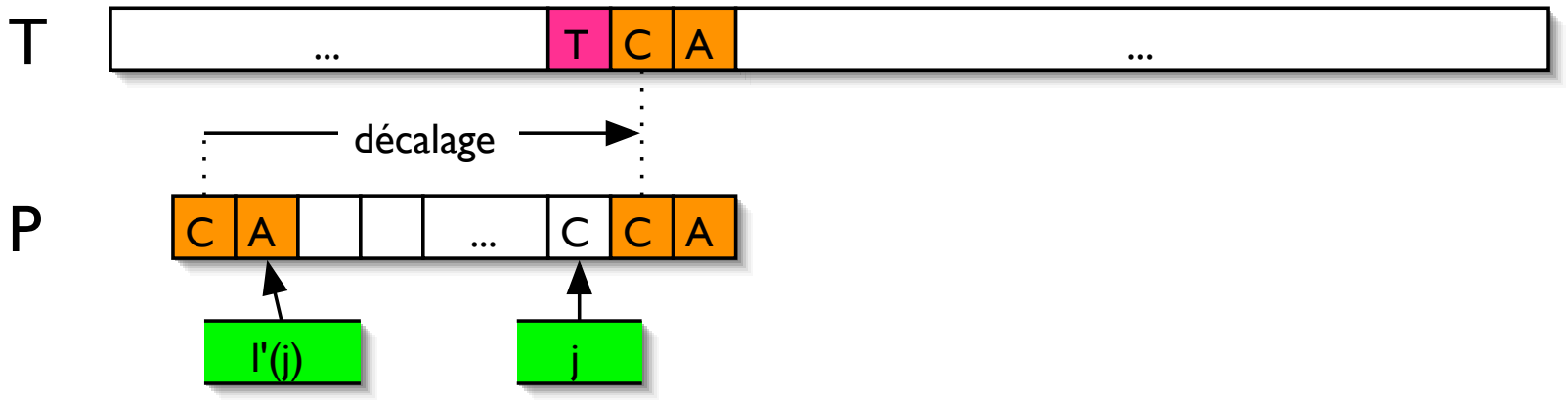
```
1  pour j ← 1, ..., n − 1 ; L'(j) ← 0
2  pour i ← 1, ..., n − 1
3       j ← n − Z_{n−i+1}(P^R) + 1
4       L'(j) ← i
```

Le bon suffixe — $L'(j+1) = 0$

Déf. Pour chaque j soit

$$l'(j) = \max\{0\} \cup \left\{ i : i \geq j, P[i..n] = P[1..n-i+1] \right\}.$$

$l'(j)$ est la longueur maximale d'un suffixe de $P[j..n]$ qui est identique à un préfixe de P.

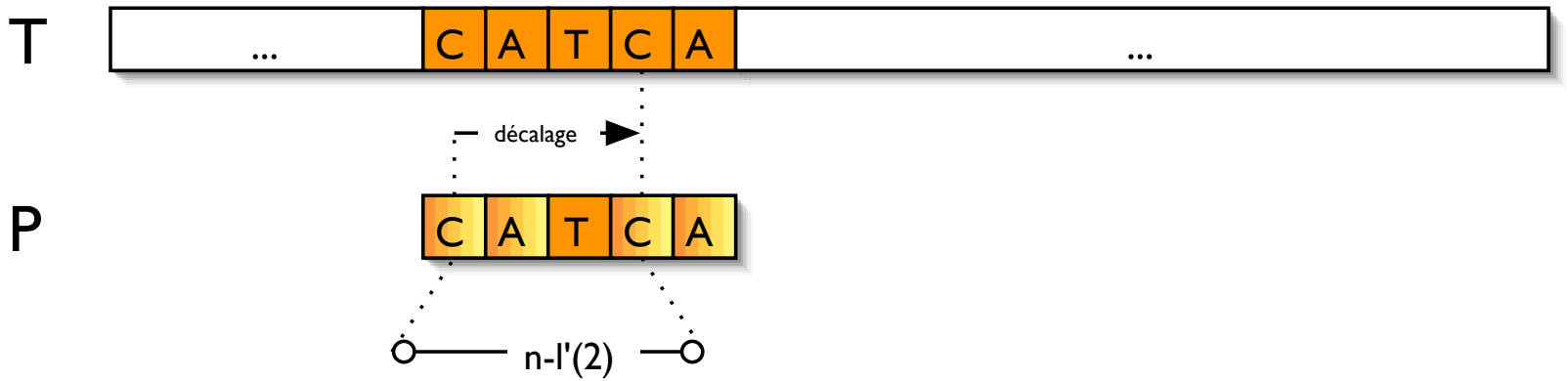


Le bon suffixe — 4

Thm.

$$l'(j) = \max\left\{Z_i(P) : i \geq j, Z_i(P) = n - i + 1\right\}$$
$$= n + 1 - \min\left\{i : i \geq j, Z_i(P) = n - i + 1\right\}$$

Décalage après une occurrence trouvée :



Calcul du décalage

Décalage en Ligne 10 de Boyer-Moore :

Cas 1 : mismatch en $T[k'] \neq P[j]$
calculer d_{mc} pour le mauvais caractère
si $L'(j+1) > 0$, alors $d_{\mathsf{bs}} = n - L'(j+1)$
sinon $d_{\mathsf{bs}} = n - l'(j+1)$
décalage par $\max\{d_{\mathsf{mc}}, d_{\mathsf{bs}}\}$

Cas 2 : match
décalage par $n - l'(2)$ (égale à n si $l'(2) = 0$)

Algorithme de Rabin-Karp

Déscription pour séquences binaires

Déf. Pour une séquence binaire S, soit

$$\mathcal{F}(S) = \sum_{i=1}^{|S|} S[i]2^{|S|-i}$$

Idée 1 : si $P = T[k..k+n-1]$, alors $\mathcal{F}(P) = \mathcal{F}(T[k..k+n-1])$.

$\Rightarrow$ comparaison de séquences comme
arithmétique sur des nombres binaires

Arithmétique binaire

arithmétique ne nous aide pas encore

1. pour comparer deux nombres binaires $a_1, a_2 > 0$, on doit comparer $B = 1 + \max\{\lceil \log_2 a_i \rceil\}$ bits : ça prend $O(B)$. Ici, $B = |P| = n$ quand on compare $\mathcal{F}(P)$ et $\mathcal{F}(T[k..k+n-1])$.

2. calcul direct de $\mathcal{F}(S)$ prend $O(|S|^2)$

Règle de Horner

Soit $A(x)$ un polynôme :

$$A(x) = \sum_{j=0}^{n-1} a_j x^j.$$

Évaluation efficace de $A(x)$ en un point donné x_0 : règle de Horner

$$A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(a_{n-2} + x_0(a_{n-1}))\cdots)).$$

$(n-1)$ additions et $(n-1)$ multiplications

$\Rightarrow$ calcul de $\mathcal{F}(S)$ en $O(|S|)$.

Arithmétique modulaire

Idée 2 : soit $q > 1$ un nombre quelconque ; si $P = T[k..k+n-1]$, alors

$$\mathcal{F}(P) \bmod q = \mathcal{F}(T[k..k+n-1]) \bmod q.$$

Rappel de l'arithmétique modulaire

1. $a \bmod q = \min\{r : r \geq 0, \exists b \text{ t.q. } a = bq + r\}$ — valeur unique

2. $(a \pm b) \bmod q = \Big((a \bmod q) \pm (b \bmod q)\Big) \bmod q$

3. $(ab) \bmod q = \Big((a \bmod q)(b \bmod q)\Big) \bmod q$

Rabin-Karp — 2

Comment calculer $\mathcal{F}_q(S) = \mathcal{F}(S) \bmod q$?
Règle de Horner+arithmétique modulaire :

1. $H_1 = \mathcal{F}_q(S[1..1]) = S[1]$
2. $H_2 = \mathcal{F}_q(S[1..2]) = (2H_1 + S[2]) \bmod q$
j. $H_j = \mathcal{F}_q(S[1..j]) = (2H_{j-1} + S[j]) \bmod q$

$H_{|S|} = \mathcal{F}_q(S)$.

Thm. Pour toute $j > 1$, $2H_{j-1} + S[j] < 2q$.

$\Rightarrow$ Si q est «petit», alors chaque opération sur des mots-machine (4 ou 8 octets) : temps total $O(|S|)$.

Rabin-Karp — 3

Calculer $t_k = \mathcal{F}(T[k..k+n-1]) \bmod q$:

1. $t_1 = \mathcal{F}(T[1..n])$
2. $t_2 = (2t_1 - u_n T[1] + T[n+1]) \bmod q$
k. $t_k = (2t_{k-1} - u_n T[k-1] + T[k+n-1]) \bmod q$

où $u_n = 2^n \bmod q$.

Analyse du temps de calcul :

: u_n calculé en $O(n \log n)$ par élévations répétées au carré
: t_1 calculé en $O(n)$
: t_k calculé en $O(1)$

$\Rightarrow$ temps total : $O(m)$.

Rabin-Karp — 4

Comment choisir q ? Au hasard ! (entre 2 et K)

Probabilité d'erreur en position k fixe :
- on a $\mathcal{F}_q(P) = \mathcal{F}_q(T[k..k+n-1])$ mais $\mathcal{F}(P) \neq \mathcal{F}(T[k..k+n-1])$
- donc q est un diviseur de $\Big(\mathcal{F}(P) - \mathcal{F}(T[k..k+n-1])\Big) \neq 0$

Soit $U_k = \Big|\mathcal{F}(P) - \mathcal{F}(T[k..k+n-1]\Big|$

$$\mathbb{P}\Big\{q|U_k\Big\} = \frac{\text{nombre de diviseurs}}{K-1}.$$

Rabin-Karp — 5

Meilleure idée pour choisir q : soit q un nombre premier.

Si $|P| < 64$, alors $0 < U_k \leq 2^{63}$. Choisissons $K = 2^{63}$. Le produit des 16 nombres premiers plus petits ($= 2 \cdot 3 \ldots 53$) est plus grand que 2^{64}.

$\Rightarrow U_k$ as moins que 15 diviseurs qui sont des nombres premiers :

$$\mathbb{P}\Big\{q|U_k\Big\} \leq \frac{15}{2^{63}-1} \approx 1.7 \cdot 10^{-18}.$$

Rabin-Karp — 6

On peut même répéter la procédure avec plusieurs [C] choix de q.

$$\mathbb{P}\Big\{q|U_k, q_2|U_k, \ldots, q_C|U_k\Big\} \leq \left(\frac{15}{2^{63}-1}\right)^C.$$

Avec $C = 10$, ça donne cca. 10^{-178} !

Note : il existe des algorithmes efficaces pour tests de primalité

Densité des nombres premiers :
pour toute K, $\pi(K) = c_K \frac{K}{\ln K}$, où $1 \leq c_K < \beta$ et β ne dépend pas de K ;
$c_K \rightsquigarrow 1$.

Rabin-Karp — 7

I. Et si $|P|$ est plus grande que la longueur du mot-machine ? Alors opérations binaires sur les bits $\rightarrow$ tout multiplié par $O(\log q)$.

II. Extension pour un alphabet Σ de taille $|\Sigma > 2$.

Solution 1. Soit $|\Sigma| \leq 2^m$, alors on peut utiliser un encodage binaire sur m bits pour chaque caractère ; match est permis dans toutes les m-ième positions de T seulement.

Solution 2. Refaire tout avec $\mathcal{F}(S) = \sum S[i]|\Sigma|^{|S|-i}$.

Banques de données

NCBI : «National Center for Biotechnology Information» — États-Unis

Interface [Entrez] à plusieurs bases de données :

– séquences d'acides nucléiques
– séquences protéiques
– PubMed : publications
– structures
– taxonomie
– …

GenBank

Séquences d'ADN : GenBank (É-U), DDBJ (Japon), EMBL (Europe)

GenBank «flatfile» : exemple (HUMXIHB :

```
LOCUS       HUMXIHB                  458 bp    mRNA    linear   PRI 14-JAN-1995
DEFINITION  Human zeta hemoglobin mRNA, complete cds.
ACCESSION   M24173
VERSION     M24173.1  GI:340391
KEYWORDS    zeta-globulin.
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 458)
  AUTHORS   Cohen-Solal,M.M., Authier,B., deRiel,J.K., Murnane,M.J. and
            Forget,B.G.
  TITLE     Cloning and nucleotide sequence analysis of human embryonic
            zeta-globin cDNA
  JOURNAL   DNA 1 (4), 355-363 (1982)
  MEDLINE   83182021
   PUBMED   6963223
COMMENT     Original source text: Human erythroleukemia cell line K562, cDNA to
            mRNA, clones 1 (1g7-8), 2 (4p7-7), and 3 (5a3-3).
```

GenBank - champs 1

LOCUS

– 1–10 caractères alphanumériques ; jadis l'identificateur de la séquence (p.e. l'abbreviation du gène), préservée pour compatibilité seulement.
– longueur et type de la séquence (DNA, mRNA, tRNA, rRNA)
– code de la division (p.e. PRI) et date de dernière modification.

DEFINITION «sommaire» de la séquence : espèce et le nom de la séq

ACCESSION nombre d'accession : clé dans la base de donnée. Identificateur unique parmi les BDs. Forme AA999999. L'accno est généré automatiquement lors de la soumission d'une séquence à la BD.

GenBank - champs 2

KEYWORDS et SOURCE : moins d'importance

VERSION donne ⟨accno⟩.⟨version⟩ et gi : identificateur de GenInfo. Ce sont des identificateurs des *séquences* (qui peut changer pour le même accno).

REFERENCE

GenBank — exemple cont.

```
FEATURES             Location/Qualifiers
     source          1..458
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606"
                     /map="16p13.3"
     gene            1..458
                     /gene="HBZ"
     CDS             30..458
                     /gene="HBZ"
                     /note="zeta hemoglobin"
                     /codon_start=1
                     /protein_id="AAA61306.1"
                     /db_xref="GI:340392"
                     /db_xref="GDB:G00-119-302"
                     /translation="MSLTKTERTIIVSMWAKISTQADTIGTETLERLFLSHPQTKTYF
                     PHFDLHPGSAQLRAHGSKVVAAVGDAVKSIDDIGGALSKLSELHAYILRVDPVNFKLL
                     SHCLLVTLAARFPADFTAEAHAAWDKFLSVVSSVLTEKYR"
BASE COUNT       80 a    173 c    127 g     78 t
ORIGIN      79 bp upstream of BglII site.
        1 actccagtgc agctgcccac cctgccgcca tgtctctgac caagactgag aggaccatca
          .......
      421 cggtcgtatc ctctgtcctg accgagaagt accgctga
//
```

GenBank - champs 3

FEATURES annotation de la séquence : un «feature» comprend un mot-clé, sa position, et des qualifieurs

position : sous-mot [p.e., `2..280`], entre deux bases [p.e., `91^92`], `...`,
et opérations : `complement(·)`, `join(·,...,·)`

mots-clé :

- `source` information taxonomique
- `CDS` partie traduite en une séquence protéique
- `exon`, `intron`, `gene`
- `repeat_region`
- `...`

Exemple : U96726

GenBank - entrées virtuelles

Exemple : U00089

```
LOCUS       U00089                816394 bp    DNA     circular CON 06-DEC-2002
DEFINITION  Mycoplasma pneumoniae M129, complete genome.
...
CONTIG      join(AE000016.2:1..19313,AE000015.2:59..17535,AE000014.2:22..12521,
            AE000013.2:53..10328,AE000012.2:59..10228,AE000011.2:59..15387,
            ... [plusieurs lignes]
            AE000019.2:59..10270,AE000018.2:59..11147,AE000017.2:62..15963)
//
```

NCBI

GenBank «flatfile» généré automatiquement à partir des bases de données.

Entrez : interface integré : recherche par identificateurs, mots clés, auteurs, etc.

BLAST : famille d'outils pour trouver des occurrences inexactes d'une séquence P dans le «texte» T

choix de T : `nr`, `est`, `month`, etc.

BLAST

BLAST : recherche par hachage + théorie de probabilités pour alignements locaux

Hachage — idée principale :
pour trouver l'occurrence inexacte de P en T

1. fixer $w > 0$
2. comparer chaque sous-mot de longueur w de P avec ceux de T
3. extension des matches pour obtenir un alignement local entre P et T

$\Rightarrow$ on trouvera rapidement les alignements qui contiennent w matches consécutifs

Hachage

Pour un sous-mot S (séquence ADN) de longueur $|S| = w$, on calcule

$$\mathcal{F}(S) = \sum_{i=1}^{w} e(S[i])4^{w-i},$$

avec e : $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\} \mapsto \{0, 1, 2, 3\}$.

On a donc $0 \leq \mathcal{F}(S) < 4^w$.

On remplit un tableau en calculant $\mathcal{F}(T[k..k + w - 1])$ pour chaque position k. Dans rangée t, se trouve la liste L_t des positions $k_1, k_2, \ldots$ pour lesquelles $\mathcal{F}(T[k_i..k_i + w - 1]) = t$.

Hachage 2

Structure de données pour le tableau (idée de base) :
arbre de recherche pour les rangées+liste liée pour chaque rangée

Implantation facile en Java : on peut utiliser les sous-mots comme clés directement, `Hashtable` calcule une sorte de $\mathcal{F}(\cdot)$ pour nous automatiquement

Hachage 3

- tableau pour T est calculé en avance.

- si $|T| \ll 4^w$, la plupart des rangées sont vides

- pour un P donné, on calcule $p_i = \mathcal{F}(P[i..i+w-1])$ pour $i = 1, \ldots, |P| - w + 1$

- pépins («seeds») : membres des listes L_{p_i} — les positions dans T où on a un match

- extension d'un pépin dans les deux extrémités jusqu'à ce que la valeur de l'alignement tombe trop bas (ou simplement < 0)

BLAST - 2

quelques mots de longueur w qui sont trop fréquents sont exclus de la recherche

extension de pépins :
- aucun trou : match/mismatch seulement
- extension jusqu'à un seuil de v sur la valeur de l'alignement : l'extension à valeur maximale est choisie

résultat de la recherche : liste de paires de segments («high-scoring segment pairs» ou HSPs)

probabilités : signification — P-valeur

Signification

Supposons qu'on a trouvé une occurrence de P en T. Est-ce que c'est par chance ou non ?

P-valeur : test de l'hypothèse nulle
H_0 : «P apparaît dans T par chance»
H_1 : «l'occurrence de P dans T correspond à qqch importante»

Probabilité de H_0 donne la P-valeur : si elle est petite, on ne rejette pas l'hypothèse H_1.

$\Rightarrow$ on a besoin d'un modèle probabiliste pour calculer la probabilité de H_0.

Signification - 2

Exemple : T est une séquence de longueur n «au hasard»
au hasard : chaque caractère de T est 0 ou 1 avec probabilités $\frac{1}{2}$-$\frac{1}{2}$.

Quelle est la probabilité d'avoir $P = 00$ ou $P = 01$?

BLAST - 3

P-valeur pour un HSP avec valeur v

H_0 : HSP entre P et T donne un score aussi grand que v

modèle probabiliste : chaque caractère est choisi au hasard avec probabilités $p_{\mathtt{A}}, p_{\mathtt{C}}, p_{\mathtt{G}}, p_{\mathtt{T}}$

pondération de match/mismatch par une matrice $\mathbf{V}$

Thm. L'espérance du nombre de HSPs qui satisfont H_0 est

$$E = K|P||T|e^{-\lambda v},$$

où K est une constante et λ est la solution de

$$\sum_{c,c' \in \Sigma} p_c p_{c'} e^{\lambda \mathbf{V}[c,c']} = 1.$$

BLAST - 4

De E à P-valeur :
nombre η de HSPs est une variable aléatoire : distribution de Poisson avec espérance E :

$$\mathbb{P}\{\eta \geq 1\} = 1 - \mathbb{P}\{\eta = 0\} = 1 - e^{-E} \approx E,$$

si $E \ll 1$.

BLAST - 5

Correction pour une BD de longueur L :

nombre de HSPs dans la BD

$$\mathsf{Expect} = \frac{\left(1 - e^{-E}\right) L}{\ell},$$

où ℓ est la longueur de la séquence qui contient le match.

P-valeur : $\approx 1 - \exp(-\mathsf{Expect})$.

BLAST - 6

```
Sequences producing significant alignments:                      (bits) Value
gi|14336674|gb|AE006462.1|  Homo sapiens 16p13.3 sequence se...   119   4e-25
gi|14523048|ref|NG_000006.1|  Homo sapiens genomic alpha glo...   119   4e-25
gi|20379745|gb|BC027892.1|  Homo sapiens, hemoglobin, zeta, ...   119   4e-25
...
>gi|14336674|gb|AE006462.1|   Homo sapiens 16p13.3 sequence section 1 of 8
          Length = 258002
 Score =  119 bits (60), Expect = 4e-25
 Identities = 60/60 (100%)
 Strand = Plus / Plus
Query: 1      actccagtgcagctgcccaccctgccgccatgtctctgaccaagactgagaggaccatca 60
              ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct: 142880 actccagtgcagctgcccaccctgccgccatgtctctgaccaagactgagaggaccatca 142939
...
Lambda     K      H
    1.37    0.711     1.31

Matrix: blastn matrix:1 -3
```