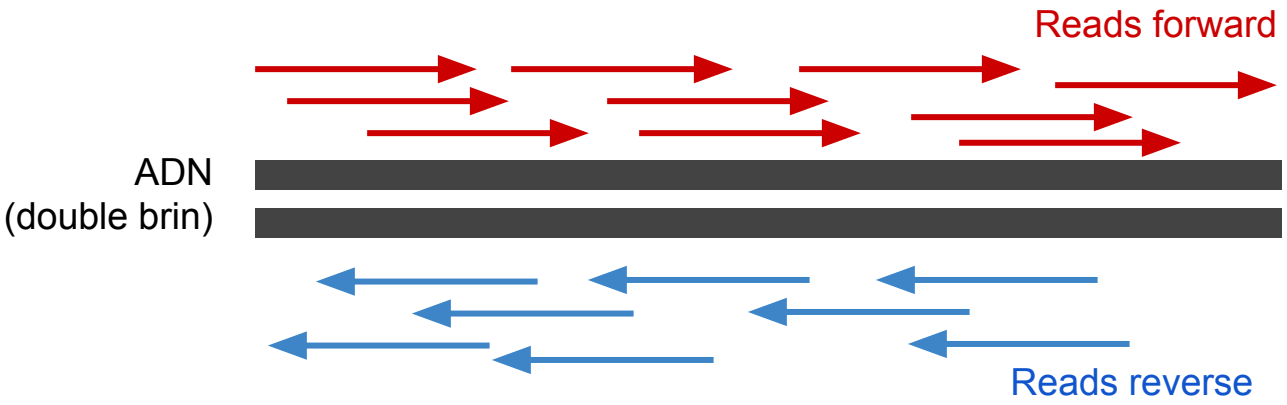
Présentation TP1

Assemblage de séquence



Présentation TP1

Assemblage de séquence

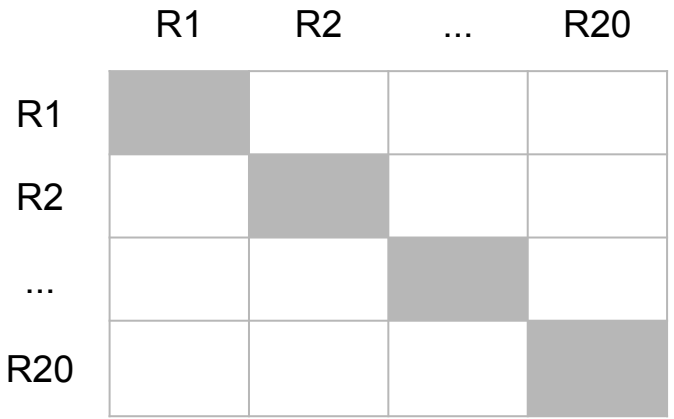
1/ Problème du chevauchement optimal entre deux séquences

Input :
→ Séquence S
→ Séquence T

Output :
→ Score
→ Alignement
→ Longueur du chevauchement

Présentation TP1
Assemblage de séquence

2/ Assemblage des segments

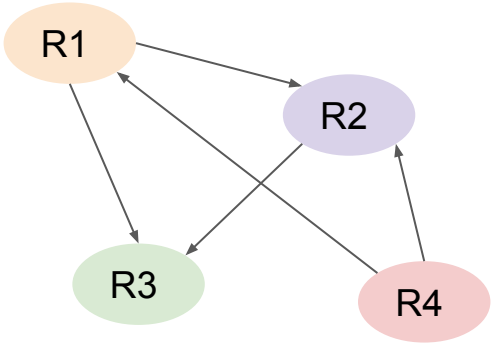


Matrice 20 x 20

Présentation TP1

Assemblage de séquence

2/ Assemblage des segments



Graphe orienté de chevauchement

⇒ Identification des reads forward et reverse

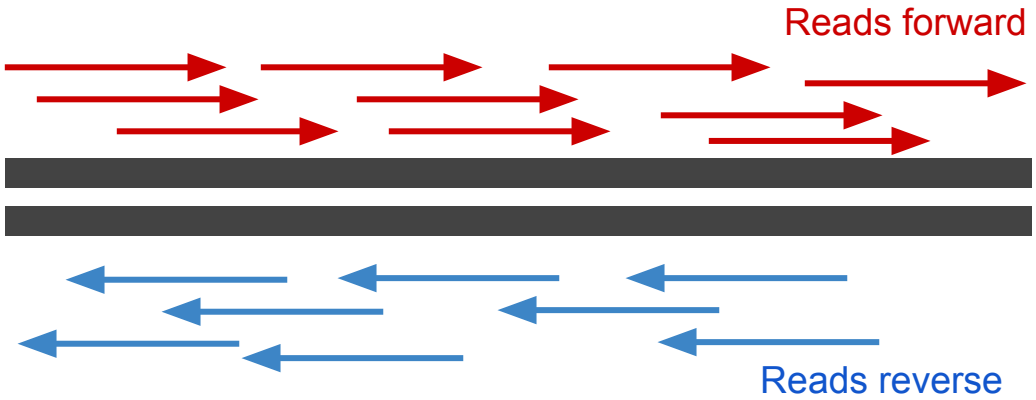
Présentation TP1

Assemblage de séquence

2/ Assemblage des segments

A=T
T=A
G≡C
C≡G

```
— A T G C G C T A —
   ‖ ‖ ||| ||| ||| ||| ‖ ‖
— T A C G C G A T —
```



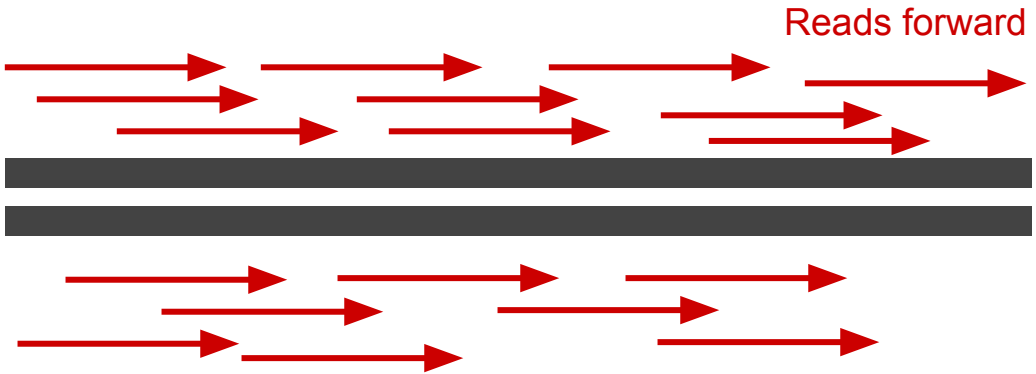
Présentation TP1

Assemblage de séquence

2/ Assemblage des segments

A=T
T=A
G≡C
C≡G

```
— A T G C G C T A —
   ‖ ‖ ||| ||| ||| ||| ‖ ‖
— T A C G C G A T —
```

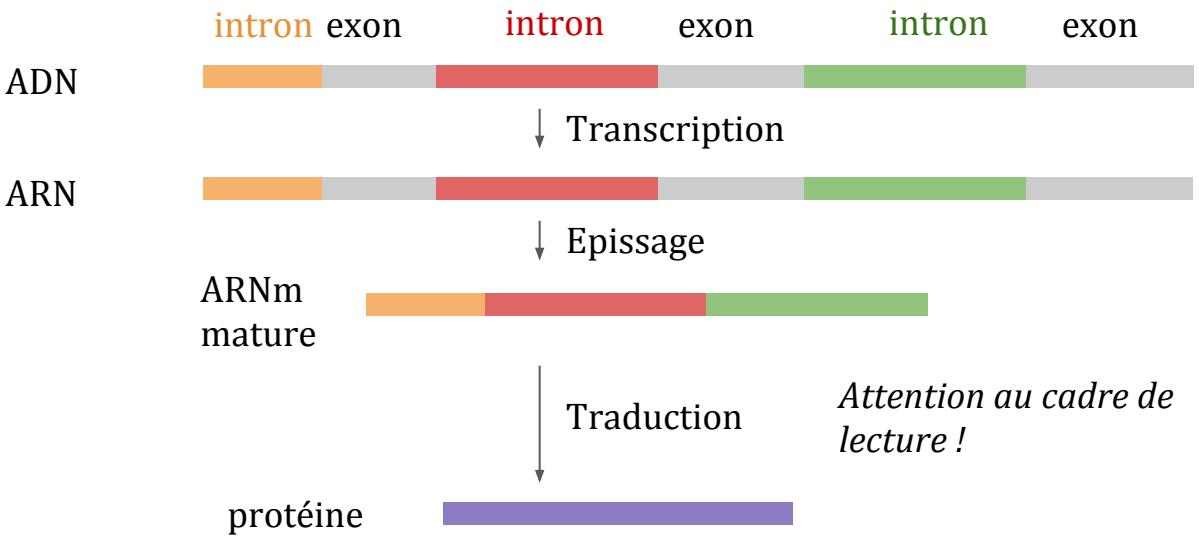


⇒ Retrouver la séquence complète

Présentation TP1

Assemblage de séquence

3/ Recherche d'introns



Programmation dynamique

Alignement pairwise avec pondération affine

Rappel pondération affine :

$$gap(k) = g_{open} + k*g_{ext}$$

- Combien de matrices de programmation dynamique ?
- Conditions initiales et équations de récurrence ?
- Comment retrouver l'alignement ?

Programmation dynamique

Alignement pairwise avec pondération affine

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + score(x_i,y_j) \\ X(i-1,j-1) + score(x_i,y_j) \\ Y(i-1,j-1) + score(x_i,y_j) \end{cases}$$

Meilleur score sachant que x_i est aligné avec y_j

$$X(i,j) = \max \begin{cases} X(i-1,j) + g_{ext} \\ M(i-1,j) + g_{open} + g_{ext} \end{cases}$$

Meilleur score sachant que x_i est aligné avec un gap

$$Y(i,j) = \max \begin{cases} Y(i,j-1) + g_{ext} \\ M(i,j-1) + g_{open} + g_{ext} \end{cases}$$

Meilleur score sachant que y_j est aligné avec un gap

Programmation dynamique

Alignement pairwise avec pondération affine

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + score(s_i,t_j) \\ X(i-1,j-1) + score(s_i,t_j) \\ Y(i-1,j-1) + score(s_i,t_j) \end{cases}$$

$$X(i,j) = \max \begin{cases} X(i-1,j) + g_{ext} \\ M(i-1,j) + g_{open} + g_{ext} \end{cases}$$

$$Y(i,j) = \max \begin{cases} Y(i,j-1) + g_{ext} \\ M(i,j-1) + g_{open} + g_{ext} \end{cases}$$

- matrice de coûts :

	A	T	C	G
A	+2	−2	−1	−2
T	−2	+2	−2	−1
C	−1	−2	+2	−2
G	−2	−1	−2	+2

- g_{open} : -3
- g_{ext} = -1

$gap(k) = g_{open} + k*g_{ext}$

	-	A	C	A	C	T	G
-	0	−∞	−∞	−∞	−∞	−∞	−∞
	−3	−∞	−∞	−∞	−∞	−∞	−∞
	−3	−4	−5	−6	−7	−8	−9
A	−∞						
	−4						
	−∞						
A	−∞						
	−5						
	−∞						
T	−∞						
	−6						
	−∞						
C	−∞						
	−7						
	−∞						

Optimisation des algorithmes d’alignement

Quelles améliorations possibles de l’algorithme d’alignement de séquences connaissez-vous ?

(principes et complexités)

Optimisation des algorithmes d'alignement

Alignement avec au plus k erreurs

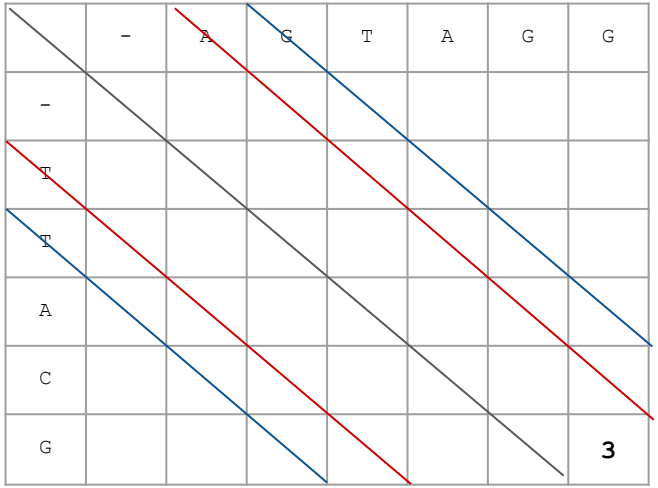
S_1 = AGTAGG

S_2 = TTACG

Alignement avec au plus k = 2 erreurs ? k = 3 ?

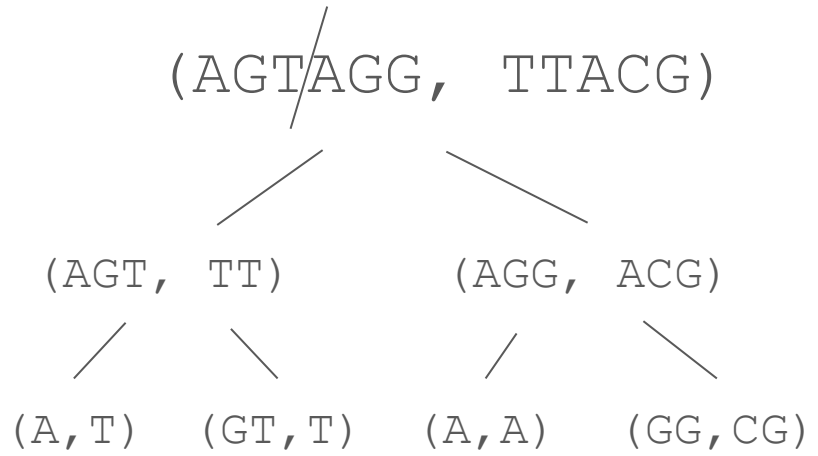
⇒ *Réponse : il existe un alignement avec 3 erreurs mais pas moins*

(matrice de programmation dynamique avec distance d'édition)



Optimisation des algorithmes d'alignement

Algorithme d'Hirshberg



Une récursion *:*

1/ Aligner la moitié de gauche de S (AGT) avec T (TTACG) et reporter la dernière ligne des scores

2/ Aligner l'inverse de la moitié de droite de S (GAA) avec l'inverse de T (GCATT) et reporter la dernière ligne de scores (inversée)

3/ Additionner les scores, le score optimal (minimal si distance d'édition) donne la partition de T (ici : TT/ACG)

Pour retrouver l'alignement :

Il est données par les feuilles de l'arbre des bipartitions