Le sujet comporte deux parties **devant être complétées directement sur le sujet**. **N'oubliez donc pas de rendre le sujet avec votre copie**.

Le barème n'est fourni qu'à titre **indicatif**, et reste sujet à modifications.

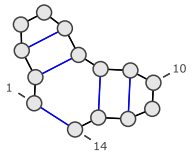
1 Transformée de Burrows-Wheeler

On considère le mot $m =$ `abracadabra` :

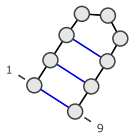
1. [3 pts] Donner la transformée de Burrows-Wheeler de m
2. [3 pts] Reconstruire, à partir de sa transformée, la séquence d'origine.

2 Alignement simplifié de structures d'ARN

Effrayé par la complexité des algorithmes d'alignement de structures secondaires, Ben, étudiant en stage dans un laboratoire, propose d'utiliser un simple algorithme d'alignement de séquence. Après tout, les structures secondaires d'ARN peuvent être représentées sous la forme de séquences bien parenthésées ...



S_1



S_2

3. [2 pts] Donner les notations parenthésées N_1 et N_2 des structure secondaires ci-dessus.

Solution: $N_1 =$ `(((..))((..)))` $N_2 =$ `(((...)))`

Pour réaliser un alignement des deux structures secondaires A et B, Ben adapte donc l'algorithme de Needlemann-Wunsch, obtenant l'équation de programmation dynamique suivante :

$$\begin{aligned} \mathcal{W}(i,0) &= \mathcal{W}(0,i) = i \cdot \alpha \\ \mathcal{W}(i,j) &= \max \left\{ \begin{array}{c} \mathcal{W}(i-1,j) + \alpha \\ \mathcal{W}(i,j-1) + \alpha \\ \mathcal{W}(i-1,j-1) + \beta(A_i, B'_j) \end{array} \right\}, \quad \forall i,j > 0 \end{aligned}$$

où α est la pénalité associée à une insertion/délétion, et β est une fonction de substitution.

4. [2 pts] Soient $n_A = |A|$ et $n_B = |B|$ les longueurs respectives des séquences A et B. Donner les complexités en temps et en mémoire du remplissage de la matrice $\mathcal{W}$.

Solution: Les complexités en temps et en mémoire de la phase de remplissage des matrices sont en $\mathcal{O}(n_A \cdot n_B)$.

5. [2 pts] Expliquer brièvement comment reconstruire l'alignement optimal à partir de la matrice (supposée déjà remplie). Donner la complexité au pire de cette phase.

Solution: Partant de la position $\mathcal{W}(n_A, n_B)$ dans la matrice, on cherche le terme ayant contribué au max, et on réitère sur la position dans la matrice associée à ce terme. Au passage, on ajoute un indel à l'alignement dans les deux premiers cas, ou un match dans le dernier cas. On itère le processus jusqu'à arriver à l'un des cas terminaux $\mathcal{W}(i, 0)$ ou $\mathcal{W}(0, i)$, où l'on complète l'alignement obtenu par i gaps.
On raisonne sur la somme $i + j$ des tailles des deux séquences restant à aligner. A chaque étape, $i + j$ diminue d'au moins 1 unité (cas des indels) et l'algorithme termine quand i ou j est nul, la complexité de la remontée est donc en $\mathcal{O}(n_A + n_B)$. Une famille de scénarios réalisant cette complexité est obtenue, par exemple, en choisissant deux séquences n'ayant aucun caractère en commun, et la complexité est donc en $\Theta(n_A + n_B)$ (optionnel).

On suppose que $\alpha = -1$ et que la fonction β est définie telle que

$$\beta(\boxed{(}, \boxed{(}) = \beta(\boxed{)}, \boxed{)}) = \beta(\boxed{.}, \boxed{.}) = +2 \quad \text{et } \beta(\_, \_) = -2 \text{ pour tout le reste.}$$

6. [3½ pts] Compléter la matrice d'alignement des séquences N_1 et N_2 de la question 9.

		N_1														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N_2	0	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
	1	-1		1	0	-1	-2	-3	-4		-6		-8		-10	-11
	2	-2	1		3		1	0	-1		-3		-5		-7	-8
	3	-3	0			5			2	1	0	-1	-2		-4	-5
	4	-4	-1	2			7	6	5		3	2		0	-1	-2
	5	-5			4			9		7	6		4	3	2	1
	6	-6							7	6	5	8	7		5	4
	7	-7		-1		5				9	8	7	6		8	7
	8	-8		-2	1		7				11		9	8		
	9	-9	-6	-3	0	3	6	9	12	11	10	9	8	11		

Solution:

		N_1														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
N_2	0	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14
	1	-1	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
	2	-2	1	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
	3	-3	0	3	6	5	4	3	2	1	0	-1	-2	-3	-4	-5
	4	-4	-1	2	5	8	7	6	5	4	3	2	1	0	-1	-2
	5	-5	-2	1	4	7	10	9	8	7	6	5	4	3	2	1
	6	-6	-3	0	3	6	9	8	7	6	5	8	7	6	5	4
	7	-7	-4	-1	2	5	8	11	10	9	8	7	6	9	8	7
	8	-8	-5	-2	1	4	7	10	13	12	11	10	9	8	11	10
	9	-9	-6	-3	0	3	6	9	12	11	10	9	8	11	10	13

7. [2 pts] Donner un des alignements optimaux de N_1 et N_2, et représenter le cheminement associé dans la matrice ci-dessus.

Solution: Il existe plusieurs alignements optimaux, en voici un :

N_1 (((. .)) ((. .)))
N_2 (((. . - - - - - .)))

8. [2½ pts] L'alignement obtenu constitue t-il un alignement valide des structures secondaires S_1 et S_2 ? Pourquoi ?
Donner, sans calcul, un alignement valide (si possible optimal) de S_1 et S_2.

Solution: L'alignement de N_1 et N_2 obtenu est invalide pour S_1 et S_2 car il aligne les deux extrémités de la paire de base $(3, 7)$ de S_2 avec des paires différentes dans S_1 (Parenthèse ouvrante $\rightarrow (3, 6)$ et fermante $\rightarrow (9, 12)$). Idem pour $(2, 8)$.

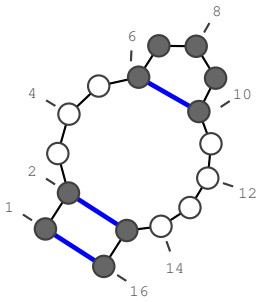
Un meilleur candidat aurait été, par exemple :

N_1 (((. . -)) ((. .)))
N_2 (((. . .)) - - - - - -)

3 Repliement d'ARN sous contraintes

Des méthodes expérimentales, tel le sondage chimique, permettent d'obtenir une information partielle sur la structure secondaire d'un ARN. On peut alors *complèter* cette information par une prédiction algorithmique des portions restées indéterminées. On va pour cela adapter l'algorithme de Nussinov en une version capable de tenir compte de contraintes, modélisées par une **contrainte partielle**, illustrée par le dessin ci-dessous.

`GCAGGGAAUCCCAUGC`
Séquence d'ARN



Contrainte partielle

Dans la contrainte partielle ci-dessus, les bases sombres sont soumise à des contraintes d'appariement (Paires (1,16), (2,15) et (6,10))), ou encore contraintes à rester non-appariées (Bases 7, 8 et 9). Les bases restées blanches sont de statut indéterminé, et sont donc libres de s'apparier (ou pas) entres elles.

9. [2 pts] On se propose d'encoder une contrainte partielle par une expression parenthèsée similaire à celle utilisée pour représenter la structure secondaire. Pour cela, on utilise des symboles :
   - `[` et `]` pour dénoter une paire de base imposée,
   - `*` pour les bases contraintes non-appariées,
   - `.` pour les bases libres.

   Donner l'expression parenthésée C correspondant à la contrainte partielle ci-dessus.

Solution: $C =$ `[[...[***]....]]`

Soit C une contrainte partielle, on définit tout d'abord une fonction α_C permettant de tester si une position rester non-appariée, définie telle que :
- $\alpha_C(i)$ renvoie vrai si et seulement la base de position i peut être laissée non-appariée sans contredire de contrainte de C.

10. [2 pts] Donner, pour la contrainte partielle C déterminée à la question 9, la valeur de vérité Vrai (V) ou Faux (F) de la fonction $\alpha_C(i)$ pour tout position i dans la séquence.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C																
$\alpha_C(i)$																

Solution:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C	`[`	`[`	`.`	`.`	`.`	`[`	`*`	`*`	`*`	`]`	`.`	`.`	`.`	`.`	`]`	`]`
$\alpha_C(i)$	F	F	V	V	V	F	V	V	V	F	V	V	V	V	F	F

On se munit enfin d'une fonction β_C permettant de tester la compatibilité d'une paire de base avec C, et définie telle que :

– $\beta_C(i,j)$ renvoie vrai si la paire (i,j) ne *croise* aucune paire de base (a,b) imposée par C ($i \leq a \leq j \leq b$ ou $a \leq i \leq b \leq j$), et si ni i ni j ne sont des positions contraintes à rester non-appariées.

11. [2 pts] Compléter le tableau ci-dessous donnant la valeur de la fonction $\beta_C(i,j)$ pour tout couple de positions (i,j).
Conseil : Plutôt que de remplir le tableau case par case, on aura tout intérêt à se focaliser sur les case interdites par les différentes contraintes de C.

		i															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j	1																
	2	∅															
	3	∅	∅														
	4	∅	∅	∅													
	5	∅	∅	∅	∅												
	6	∅	∅	∅	∅	∅											
	7	∅	∅	∅	∅	∅	∅										
	8	∅	∅	∅	∅	∅	∅	∅									
	9	∅	∅	∅	∅	∅	∅	∅	∅								
	10	∅	∅	∅	∅	∅	∅	∅	∅	∅							
	11	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅						
	12	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅					
	13	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅				
	14	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅			
	15	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅		
	16	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	

Solution:

		i															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j	0	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	V
	1	∅	f	f	f	f	f	f	f	f	f	f	f	f	f	V	f
	2	∅	∅	V	V	V	f	f	f	f	f	V	V	V	V	f	f
	3	∅	∅	∅	V	V	f	f	f	f	f	V	V	V	V	f	f
	4	∅	∅	∅	∅	V	f	f	f	f	f	V	V	V	V	f	f
	5	∅	∅	∅	∅	∅	f	f	f	f	V	f	f	f	f	f	f
	6	∅	∅	∅	∅	∅	∅	f	f	f	f	f	f	f	f	f	f
	7	∅	∅	∅	∅	∅	∅	∅	f	f	f	f	f	f	f	f	f
	8	∅	∅	∅	∅	∅	∅	∅	∅	f	f	f	f	f	f	f	f
	9	∅	∅	∅	∅	∅	∅	∅	∅	∅	f	f	f	f	f	f	f
	10	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	V	V	V	V	f	f
	11	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	V	V	V	f	f
	12	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	V	V	f	f
	13	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	V	f	f
	14	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	f	f
	15	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	f

12. [2 pts] Proposez un algorithme pour remplir automatiquement la matrice ci-dessus à partir de deux listes l_p et l_n contenant respectivement les paires de bases appariées et

bases non-appariées imposées. Quelle en est la complexité ?

Solution: Assez libre ici, on s'attend à ce qu'il proposent un algo naïf en $\Theta(n^3)$ au pire : On démarre avec un tableau initialisé à V, puis pour chaque paire de base (a, b) dans la contrainte partielle, on fait $\beta_C(i, j) \leftarrow F$ pour tout $1 \leq i \leq a \leq j \leq b \leq n$ et $1 \leq a \leq i \leq b \leq j \leq n$. Comme il existe à chaque fois $\Theta(n^2)$ couples (i, j) de ce type, et que la contrainte partielle peut contenir $\Theta(n)$ paires de bases, on atteint $\Theta(n^3)$ au total. Une telle complexité peut être optimisée à $\Theta(n^2)$ en organisant correctement les paires de bases de la contrainte (Bonus), mais ça ne vaut pas nécessairement la peine car l'algo qui va suivre est de toute façon en $\Theta(n^3)$.
Je préconise de mettre les points du moment que l'analyse d'algo est convaincante.

Rappels : On se place dans le modèle d'énergie de Nussinov le plus simple, où chaque paire de base contribue pour -1 KCal.mol^{-1} à l'énergie libre. On autorise uniquement l'appariement de bases non directement consécutives, ce qui correspond au cas $\theta = 1$ dans l'algorithme vu en cours, et rappelé ci-dessous :

$$\begin{aligned}
\mathcal{N}(i,i) &= \mathcal{N}(i,i+1) = 0, \quad \forall i \in [1,n] \\
\mathcal{N}(i,j) &= \min \begin{cases} \mathcal{N}(i+1,j) & \\ -1 + \mathcal{N}(i+1,j-1) & \text{Si } \psi(i,j) \\ \min_{k=i+2}^{j-1} -1 + \mathcal{N}(i+1,k-1) + \mathcal{N}(k+1,j) & \text{Si } \psi(i,k) \end{cases}
\end{aligned}$$

ψ est ici une fonction testant la compatibilité des bases, i.e. $\psi(i, k)$ renvoie *Vrai* uniquement quand les bases aux positions (i, k) constituent une paire valide (A/U, G/C ou G/U).

13. [2 pts] Adapter l'algorithme de Nussinov de façon à prendre efficacement en considération une contrainte partielle C dans le repliement. On pourra utiliser les résultats des fonctions α_C et β_C.

Solution:

$$\mathcal{N}(i,i) = \begin{cases} 0 & \text{Si } \alpha_C(i) \\ +\infty & \text{Sinon} \end{cases},$$

$$\mathcal{N}(i,i+1) = \begin{cases} 0 & \text{Si } \alpha_C(i) \text{ et } \alpha_C(i+1) \\ +\infty & \text{Sinon} \end{cases},$$

$$\mathcal{N}(i,j) = \min \begin{cases} \mathcal{N}(i+1,j) & \text{Si } \alpha_C(i) \\ -1 + \mathcal{N}(i+1,j-1) & \text{Si } \psi(i,j) \text{ et } \beta_C(i,j) \\ \min_{k=i+2}^{j-1} -1 + \mathcal{N}(i+1,k-1) + \mathcal{N}(k+1,j) & \text{Si } \psi(i,k) \text{ et } \beta_C(i,k) \end{cases}$$

Question	Points	Score
1	3	
2	3	
3	2	
4	2	
5	2	
6	$3\frac{1}{2}$	
7	2	
8	$2\frac{1}{2}$	
9	2	
10	2	
11	2	
12	2	
13	2	
Total:	30	