BIN702 - Série d'exercices sur les réarrangements

Manuel Lafond

Exercice 1: Montrez comment simuler, d'un point de vue général, comment simuler une inversion sur génomes signés avec une opération DCJ. Une inversion prend la forme $Xa\underline{bYc}dZ \rightarrow XacYbdZ$, où X, Y, Z sont des sous-chaînes et a, b, c, d sont des extrémités de gènes.
Indice : essayez sur de petits exemples, e.g. $G_1 = a^+b^+c^+d^+$ et $G_2 = a^+c^-b^-d^+$. Ça se généralise ensuite aisément.

Solution. Prenons l'exemple, mais représenté avec la notation des extrémités de gènes. On a

$$G_1 = a_t a_h \underline{b_t b_h c_t c_h} d_t d_h$$
$$G_2 = a_t a_h \underline{c_h c_t b_h b_t} d_t d_h$$

La région affectée par l'inversion est soulignée. En termes d'adjacence, il suffit d'enlever les adjacences $a_h b_t$ et $c_h d_t$ de G_1, et de les remplacer par $a_h c_h$ et $b_t d_t$. Ceci est une opération DCJ.

Plus généralement, on remarque qu'une inversion ne fait qu'affecter deux adjacences. C'est-à-dire, disons que notre génome est $Xa\underline{bYc}dZ$, où X, Y, Z sont des sous-chaînes et a, b, c, d sont des extrémités de gènes, et que l'inversion affecte bYc. En termes d'adjacences, il suffit de retirer $\{a, b\}$ et $\{c, d\}$, puis de les remplacer par $\{a, c\}$ et $\{b, d\}$. Ce qu'on vient de décrire est une opération DCJ. □

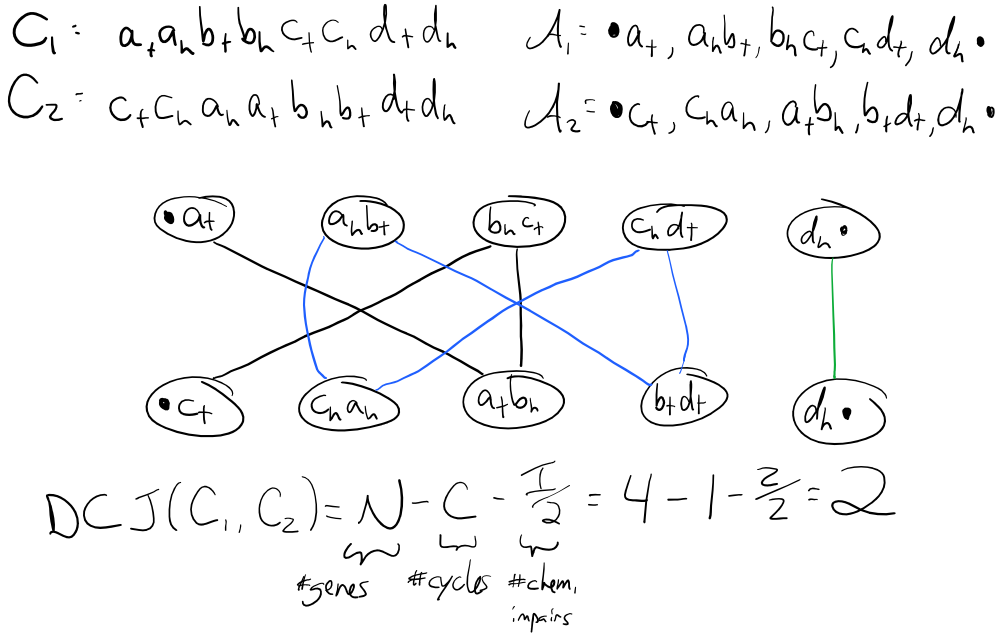
Exercice 2: Considérez les chromosomes

$$C_1 = a^+ b^+ c^+ d^+$$
$$C_2 = c^+ a^- b^- d^+$$

Le but de cet exercice est de calculer la distance DCJ de G_1 à G_2 en construisant le graphe d'adjacences, étape par étape.

a. Réécrivez C_1 et C_2 sous la forme incluant les extrémités (e.g. $a^+ = a_t a_h$) ;
b. Écrivez la liste de toutes les adjacences de C_1 et C_2 (incluant celles avec les télomères dénotés par $\bullet$) ;
c. Dessinez le graphe d'adjacences ;
d. Quelle est la distance DCJ entre C_1 et C_2 ? (n'essayez pas de trouver des opérations DCJ, utilisez la formule).

Solution.
Voir la Figure.

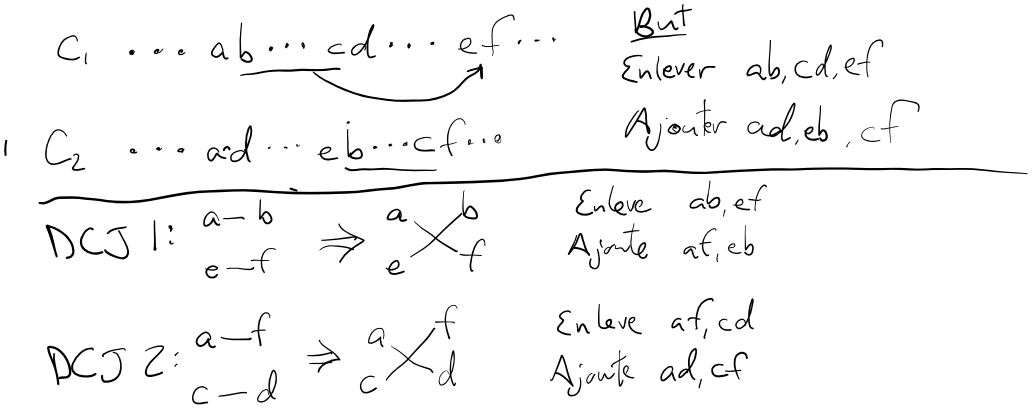


□

Exercice 3: Montrez comment simuler une transposition sur génomes signés avec deux opérations DCJ.

Indice : c'est un peu plus laborieux, mais ça se généralise aussi à partir de petits exemples.

Solution. Je me contente de vous le dessiner. En haut, le avant et l'après d'une transposition. Ici, les a, b, c, d, e, f sont des extrémités de gènes. En bas, les deux DCJ qui ont l'effet d'enlever et d'ajouter exactement les adjacences voulues. Notez que ce n'est pas la peine de redessiner les chromosomes après les DCJ : il est plus simple de rester seulement avce l'interprétation d'un génome comme une liste d'adjacences.



□

Exercice 4: Supposons que nous avons deux chromosomes non-signés A et B (donc, des chaînes de caractères ordinaires). On veut connaître le nombre minimum d'insertions pour transformer A en B (ou ∞ si ce n'est pas possible). Donnez un algorithme qui effectue cette tâche.

Solution. Comme d'habitude, la programmation dynamique peut résoudre ce problème. Une façon de le voir est que c'est comme l'alignement global, mais les deletions et mutations ont un coût infini, et les gaps créés par des insertions ont un coût de 1 pour l'ouverture, et de 0 pour l'extension (révisez les notes de cours sur les coûts arbitraires de gaps !). Les algos vu au début de la session peuvent donc résoudre ce problème.
Pour les détails, on définit $V[i, j]$ comme le nombre minimum d'insertions pour transformer $A[1..i]$ en $B[1..j]$.
Pour $V[i, 0]$ avec $i > 0$, on a $V[i, 0] = \infty$ car il est impossible de supprimer des caractères. Pour $V[0, j]$ avec $j > 0$' on a $V[0, j] = 1$ can en démarrant de la chaîne vide, il suffit d'insérer $B[1..j]$ en une opération.

Pour $i, j > 0$, on a deux cas possibles : 1) $A[i]$ et $B[j]$ sont des caractères correspondants (i.e. $B[j]$ ne requiert aucune insertion), dans quel cas le coût est $V[i-1, j-1]$, ce qui fonctionne seulement si $A[i] = B[j]$; ou 2) il faut insérer $B[j]$. Dans ce cas, on doit chercher l'emplacement de $A[i]$, et on doit l'aligner avec tous les k possibles entre 0 et $j-1$. Ceci en revient à prendre $\min_{0 \leq k < j} V[i,k] + 1$, avec le $+1$ qui sert à compter l'insertion.
Pour résumer, on a la récurrence

$$V[i,j] = \min \begin{cases} V[i-1, j-1] + \delta_{i,j} \\ \min_{0 \leq k \leq j-1} V[i,k] + 1 \end{cases}$$

où $\delta_{i,j} = 0$ si $A[i] = B[j]$, et $\delta_{i,j} = \infty$ sinon.
Il y a $O(|A||B|)$ entrées à calculer, et chacun prend un temps $O(|B|)$, pour un temps de $O(|A||B|^2)$. □