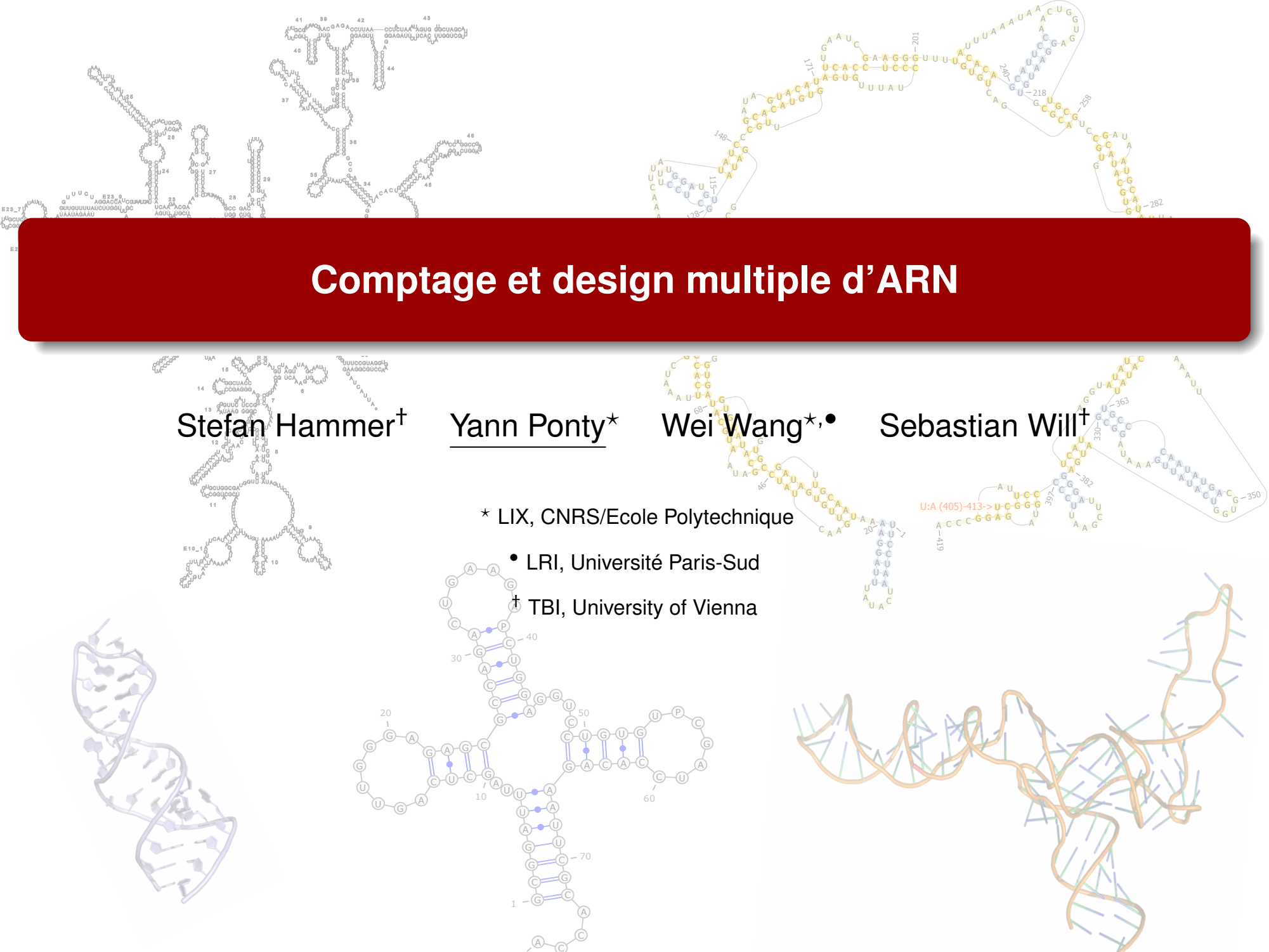
Comptage et design multiple d'ARN

Stefan Hammer† Yann Ponty⋆ Wei Wang⋆,• Sebastian Will†

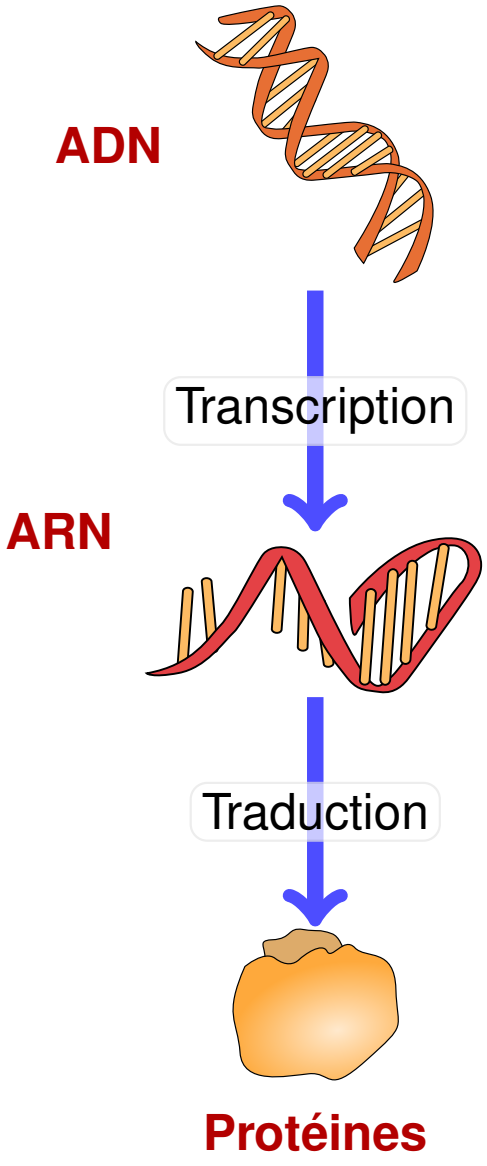
⋆ LIX, CNRS/Ecole Polytechnique

• LRI, Université Paris-Sud

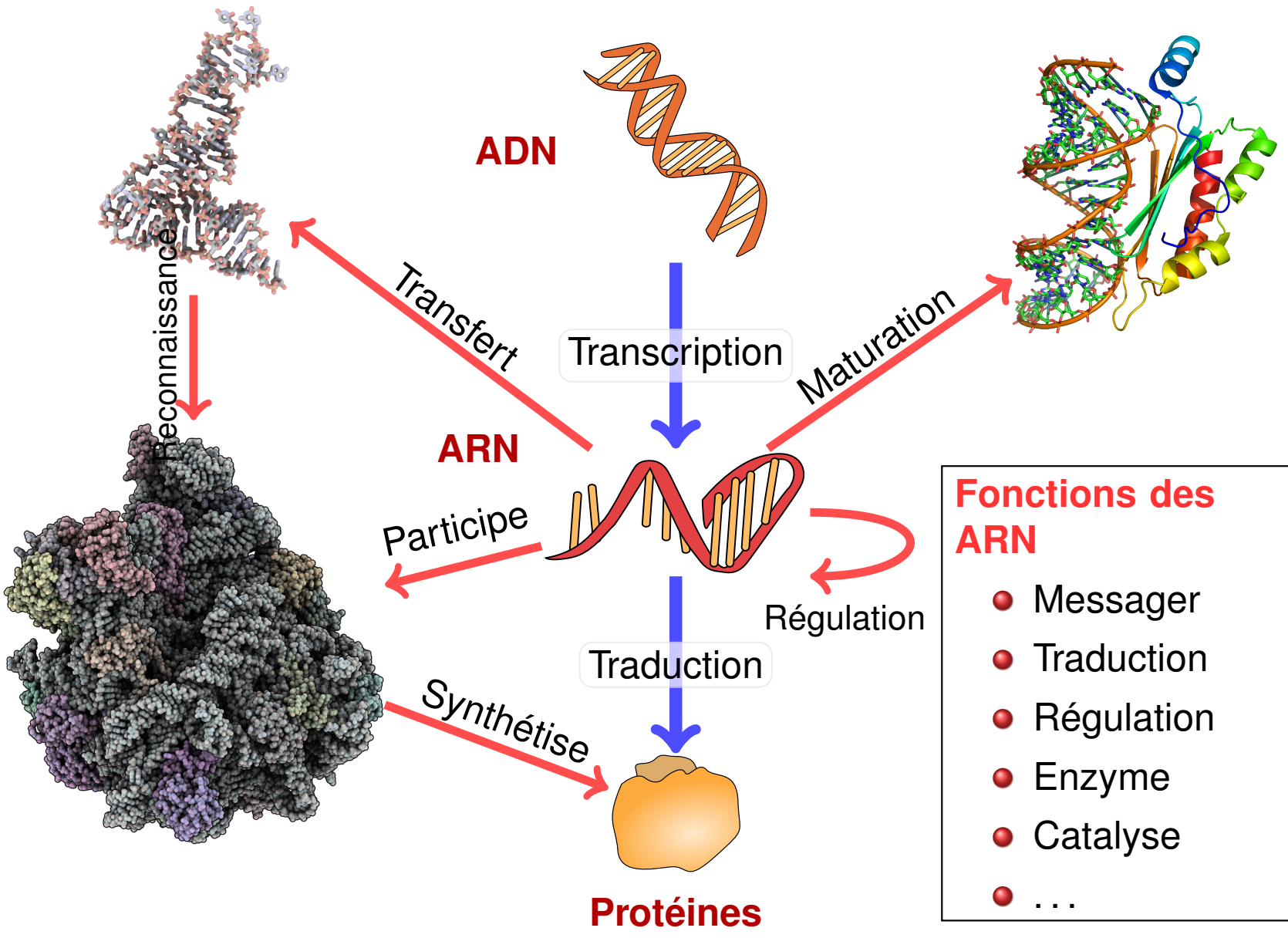
† TBI, University of Vienna



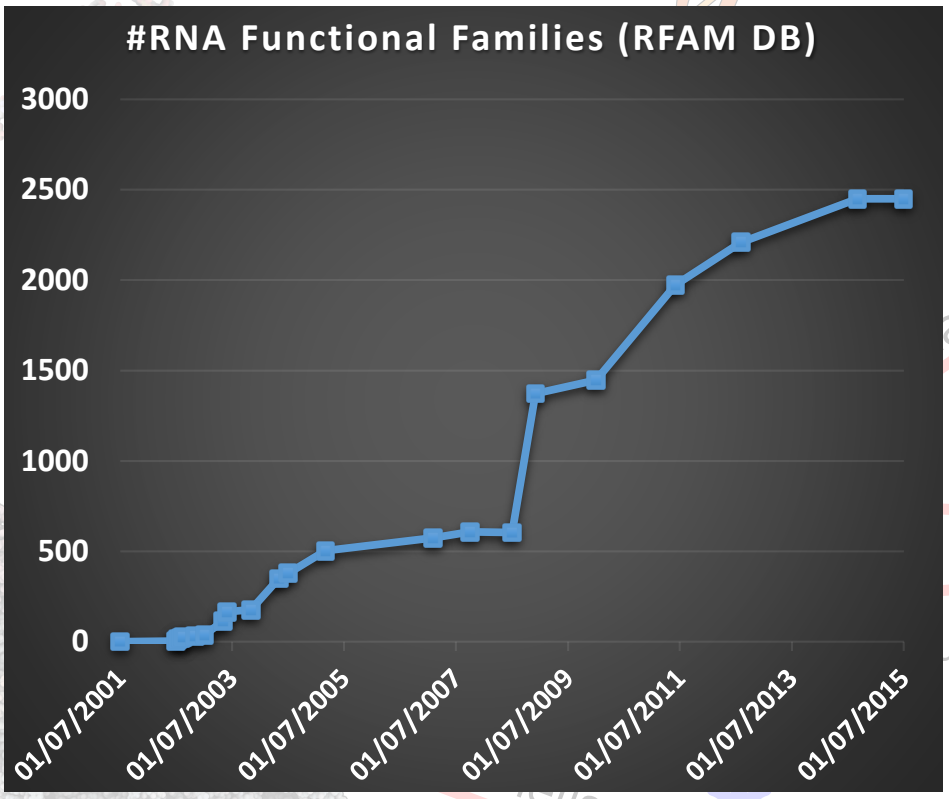
Dogme fondamental de la biologie moléculaire



Dogme fondamental de la biologie moléculaire (v2.0)



Dogme fondamental de la biologie moléculaire



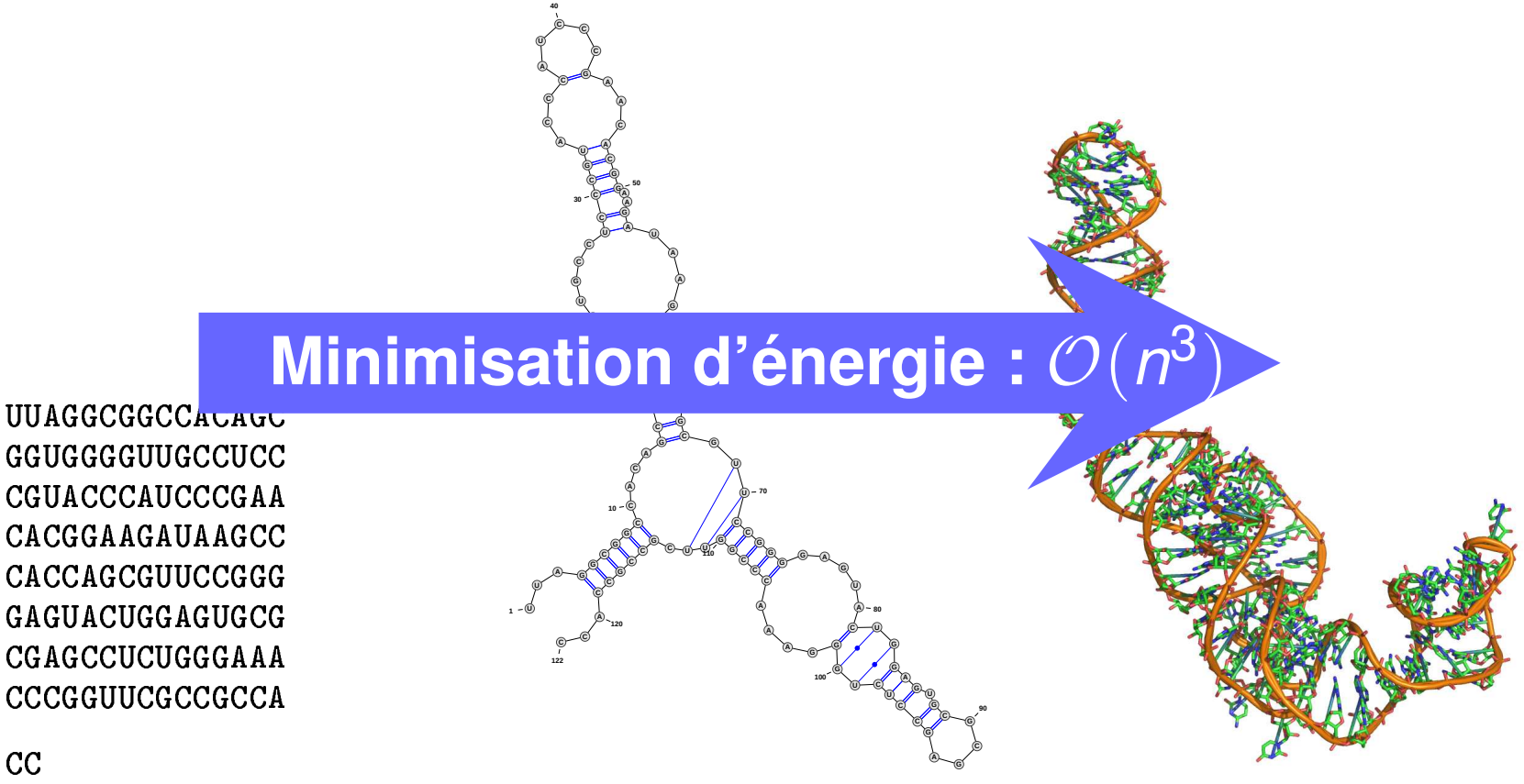
Fonctions des ARN

- Messager
- Traduction
- Régulation
- Enzyme
- Catalyse
- . . .

Protéines

Séquence et structure(s) des ARN

ARN = Polymère linéaire = Séquence sur $\{A, C, G, U\}^\star$

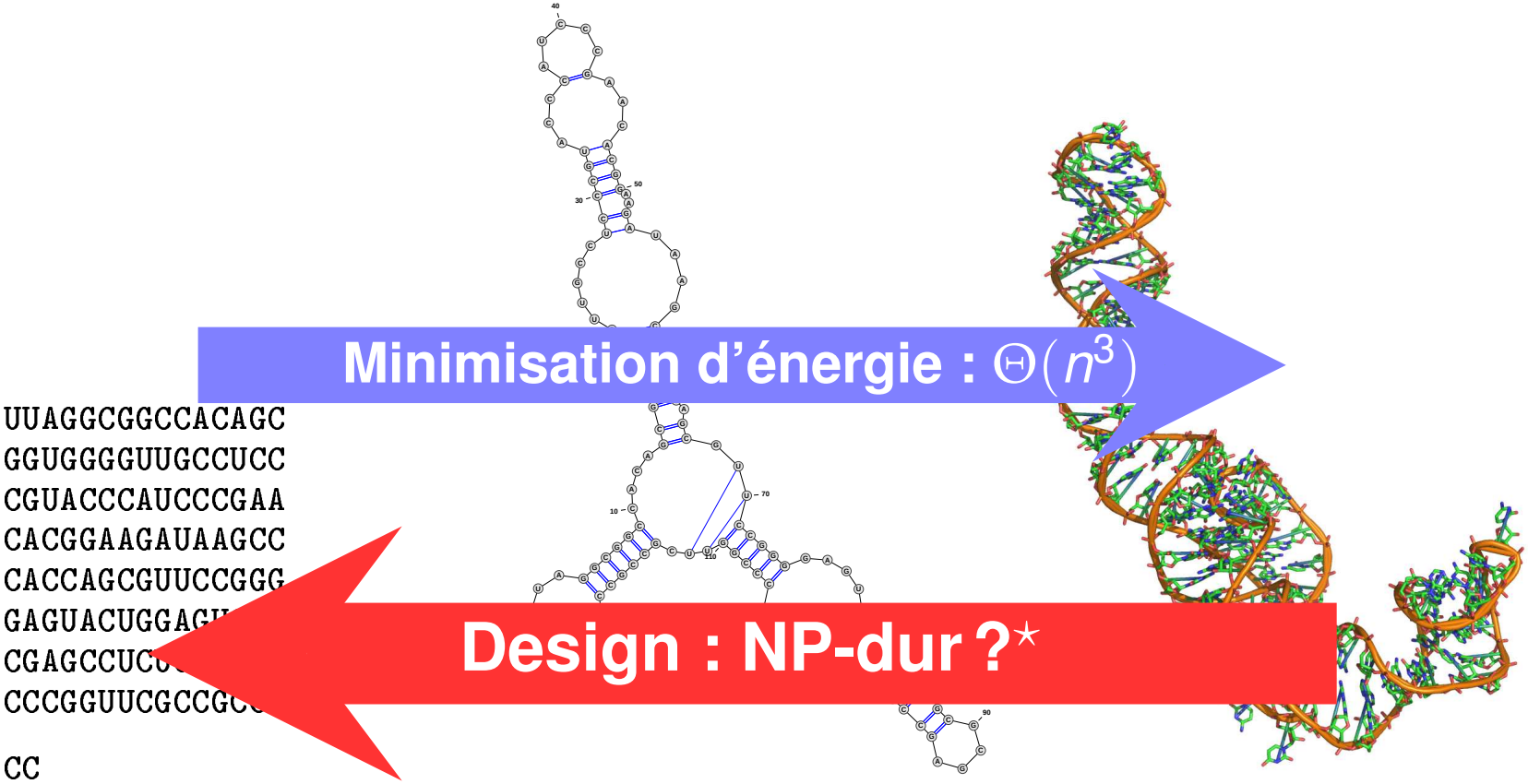


Structure Primaire Structure Secondaire Structure Tertiaire

ARNr 5s (PDBID : 1K73 :B)

Séquence et structure(s) des ARN

ARN = Polymère linéaire = Séquence sur $\{A, C, G, U\}^\star$

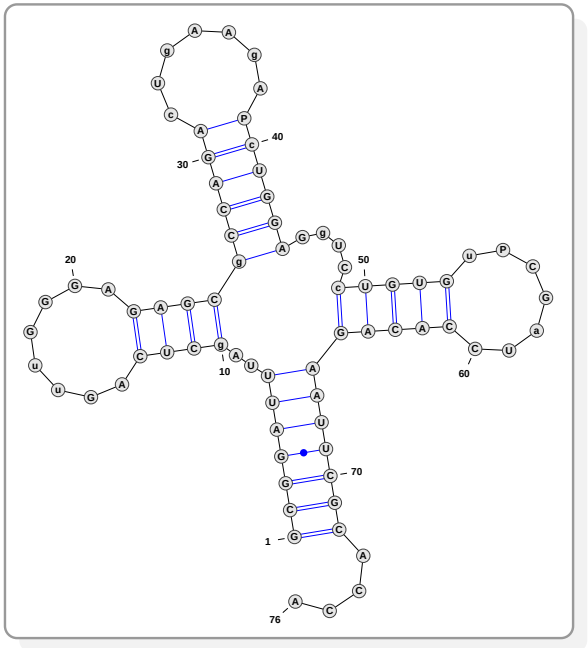


Structure Primaire Structure Secondaire Structure Tertiaire

ARNr 5s (PDBID : 1K73 :B)

*Probablement . . . (cf **[Bonnet/Rzążewski/Sikora, RECOMB'18]**)

Représentations de la structure secondaire



Graphe outer-planar

Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*

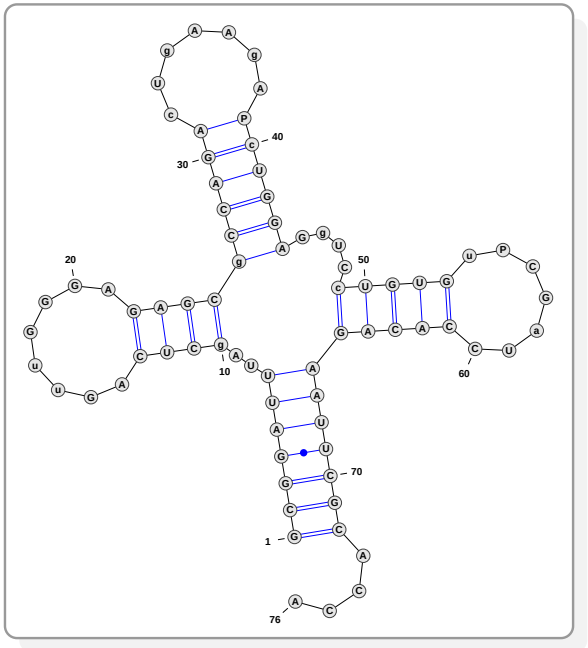
Aide à l'intuition

Différentes représentations

Même objet combinatoire

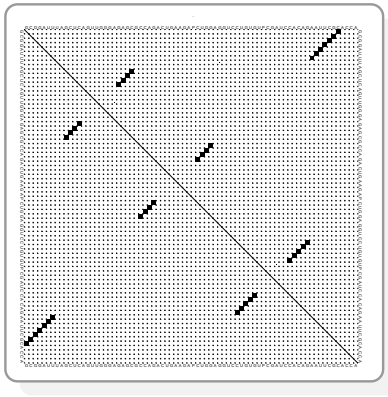
*Contraintes stériques, pseudo-noeuds

Représentations de la structure secondaire



Graphe outer-planar

Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*



Dot plots

Matrice d'adjacence*

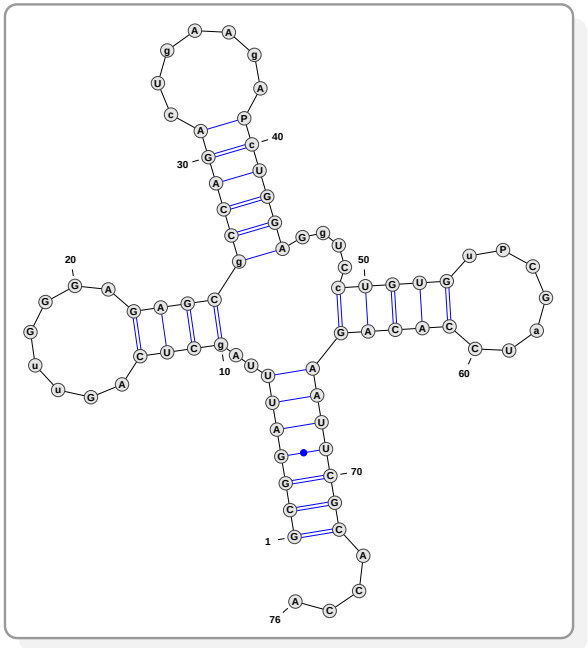
Aide à l'intuition

Différentes représentations

Même objet combinatoire

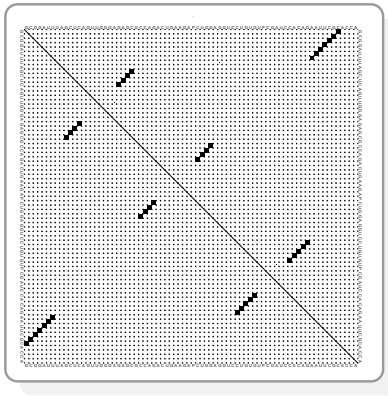
*Contraintes stériques, pseudo-noeuds

Représentations de la structure secondaire



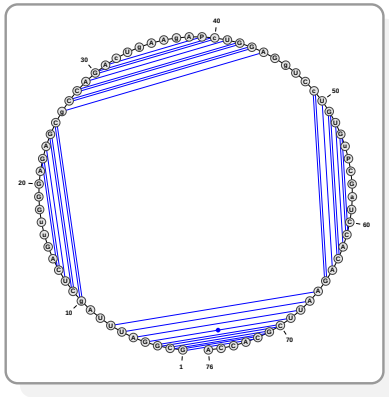
Graphe outer-planar

Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*



Dot plots

Matrice d'adjacence*



Graphe de cordes*

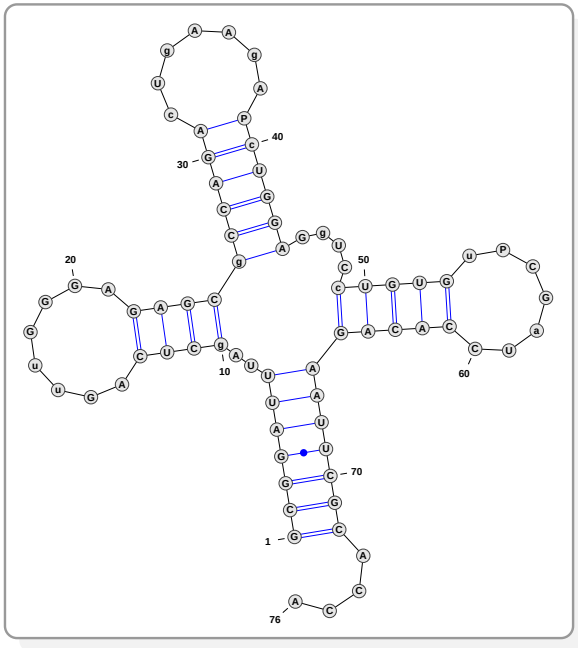
Aide à l'intuition

Différentes représentations

Même objet combinatoire

*Contraintes stériques, pseudo-noeuds

Représentations de la structure secondaire

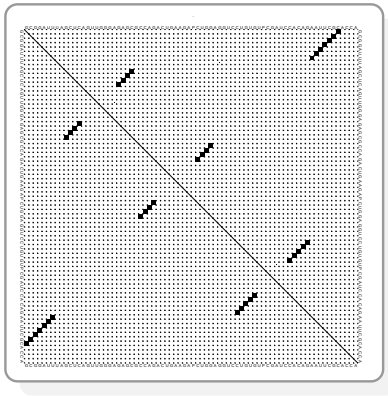


Graphe outer-planar

Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*

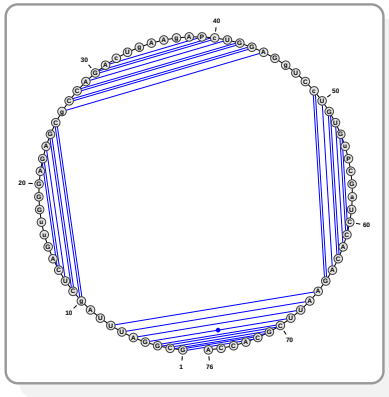
```
(((((((..((((........))))((((((.......))))))....(((((.......)))))))))))....
```

Mots de Motzkin*



Dot plots

Matrice d'adjacence*



Graphe de cordes*

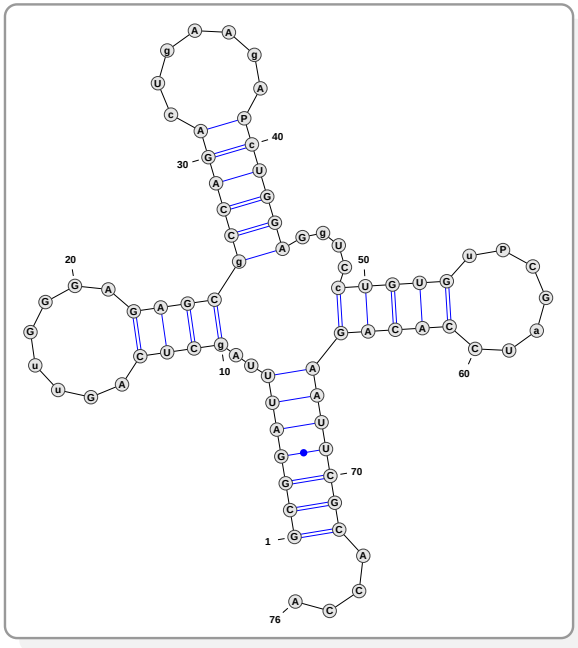
Aide à l'intuition

Différentes représentations

Même objet combinatoire

*Contraintes stériques, pseudo-noeuds

Représentations de la structure secondaire

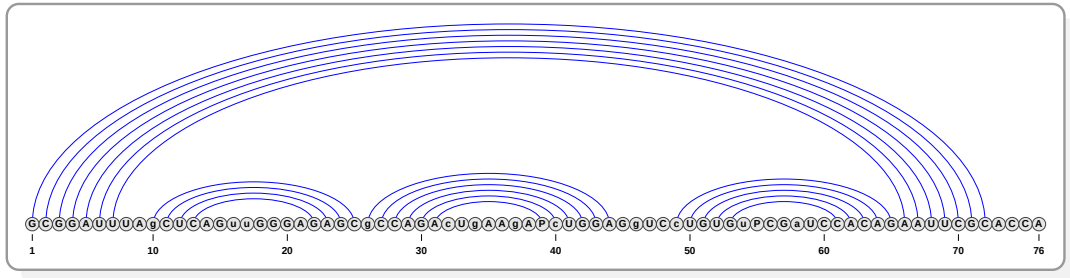


Graphe outer-planar

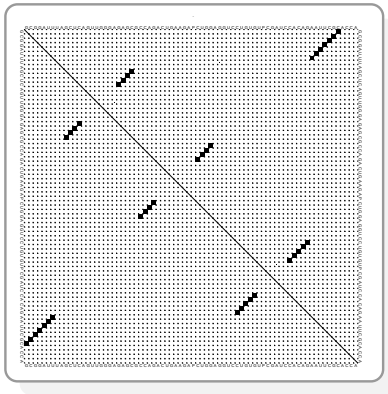
Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*

```
(((((((..((((........))))((((((.......))))))....(((((.......)))))))))))....
```

Mots de Motzkin*

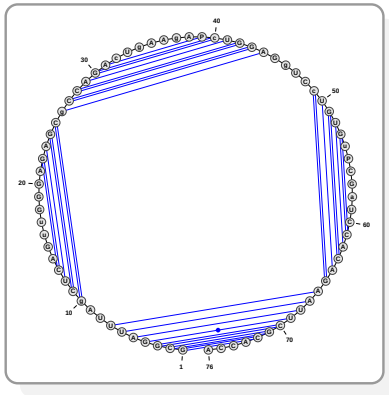


Séquence arc-annotées*



Dot plots

Matrice d'adjacence*



Graphe de cordes*

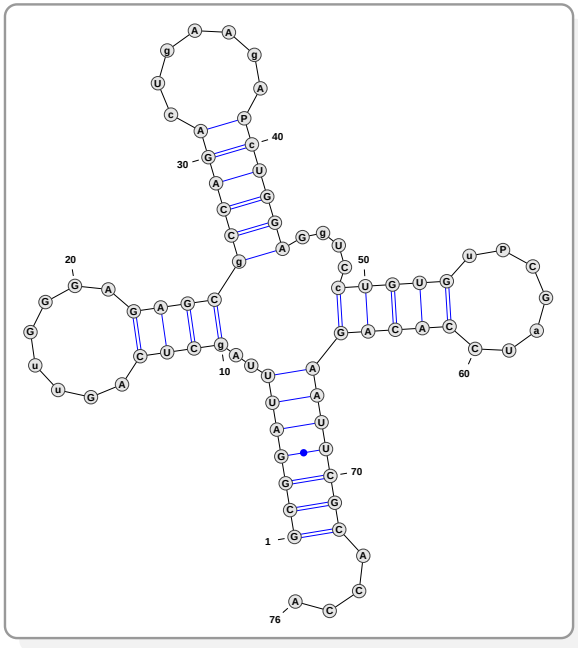
Aide à l'intuition

Différentes représentations

Même objet combinatoire

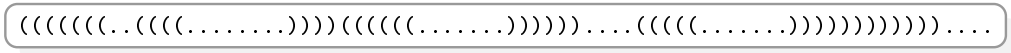
*Contraintes stériques, pseudo-noeuds

Représentations de la structure secondaire

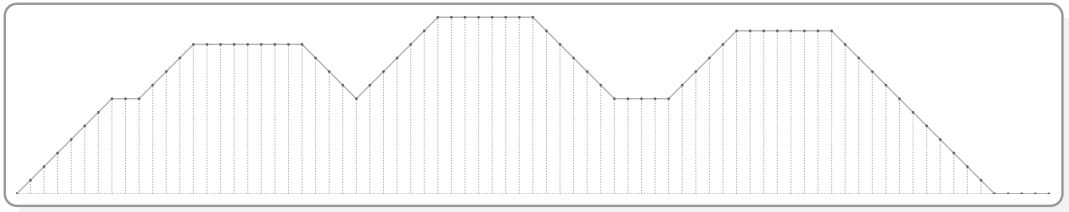


Graphe outer-planar

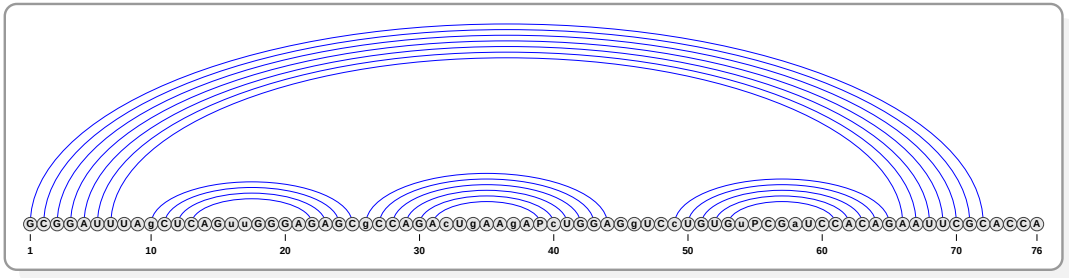
Hamiltonien, $\Delta(G) \leq 3$, 2-connexe*



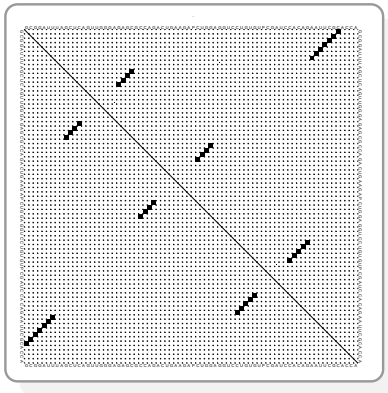
Mots de Motzkin*



Excursions* sur $\mathcal{S} = \{+1, -1, 0\}$

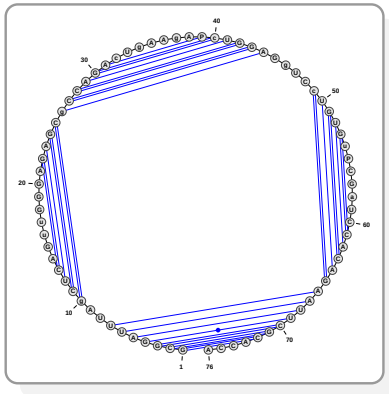


Séquence arc-annotées*



Dot plots

Matrice d'adjacence*



Graphe de cordes*

Aide à l'intuition

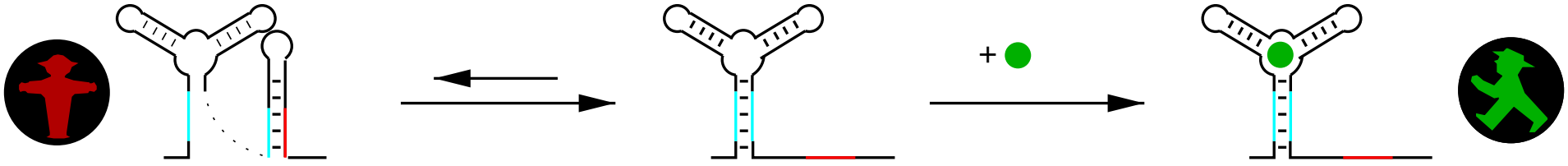
Différentes représentations

Même objet combinatoire

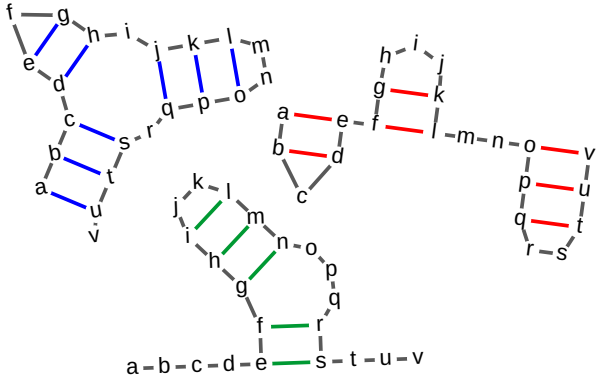
*Contraintes stériques, pseudo-noeuds

Design multiple d'ARN : Motivation

Exemple : Un *riboswitch* pour le contrôle de la traduction



Plusieurs structures cibles → ***Design multiple d'ARN***



```
abcdefghijklmnopqrstuv
((((.)).((..))).)).
(.)((...))..((.))
....(((.(..))...))...
```

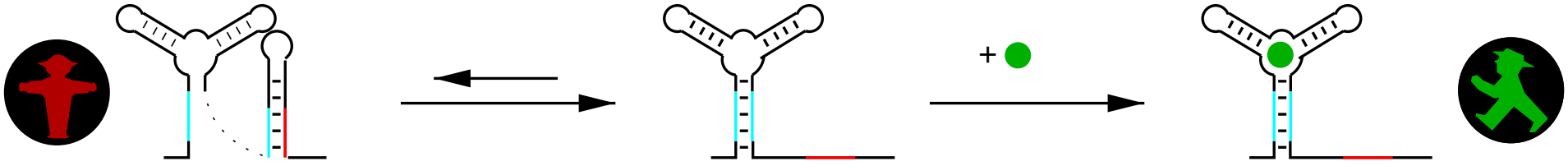
Objectif : Engendrer **aléatoirement** des séquences sous contraintes

1. **Validité** pour toutes les structures cibles
2. **Stabilité** (energie-libre faible, comparable . . .) des structures cibles
3. **Composition contrainte :** (contenu prescrit en GC), motifs exclus/forcés . . .

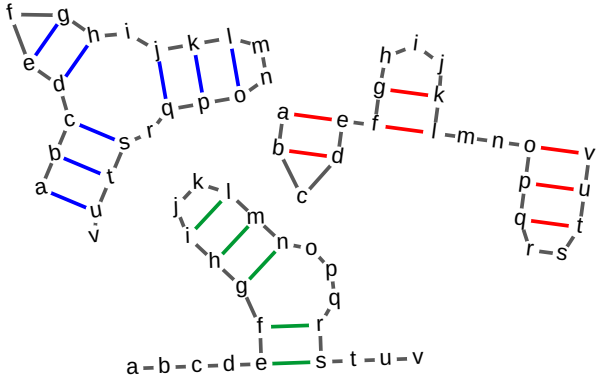
Méthode réc. (prog. dyn.) : Compter puis engendrer séquences **valides** uniformes

Design multiple d'ARN : Motivation

Exemple : Un *riboswitch* pour le contrôle de la traduction



Plusieurs structures cibles $\rightarrow$ ***Design multiple d'ARN***



```
abcdefghijklmnopqrstuv
((((((.)).(((..))).))).
((.))((...))..(((..)))
....((((((..)))...)))...
```

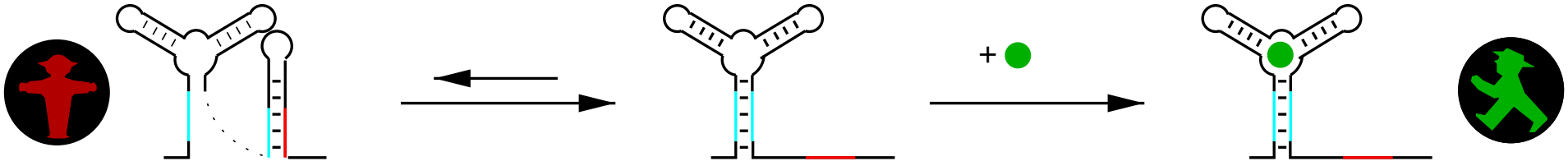
Objectif : Engendrer **aléatoirement** des séquences sous contraintes

1. **Validité** pour toutes les structures cibles
2. **Stabilité** (energie-libre faible, comparable . . .) des structures cibles
3. **Composition contrainte :** (contenu prescrit en GC), motifs exclus/forcés . . .

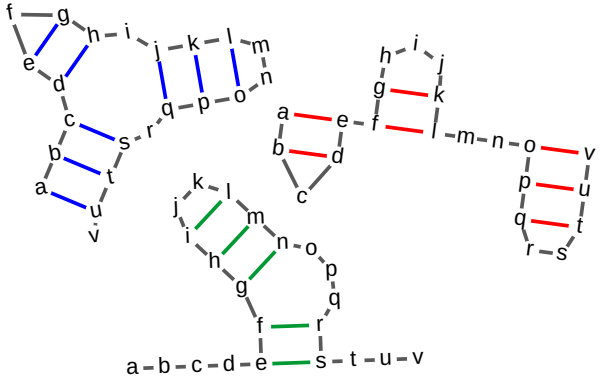
Méthode réc. (prog. dyn.) : Compter puis engendrer séquences **valides** uniformes

Design multiple d'ARN : Motivation

Exemple : Un *riboswitch* pour le contrôle de la traduction



Plusieurs structures cibles $\rightarrow$ ***Design multiple d'ARN***



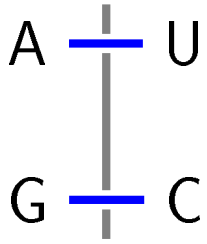
```
abcdefghijklmnopqrstuv
((((((.)).(((..))).))).
((.))((...))..(((..)))
....((((((..)))...)))...
```

Objectif : Engendrer **aléatoirement** des séquences sous contraintes

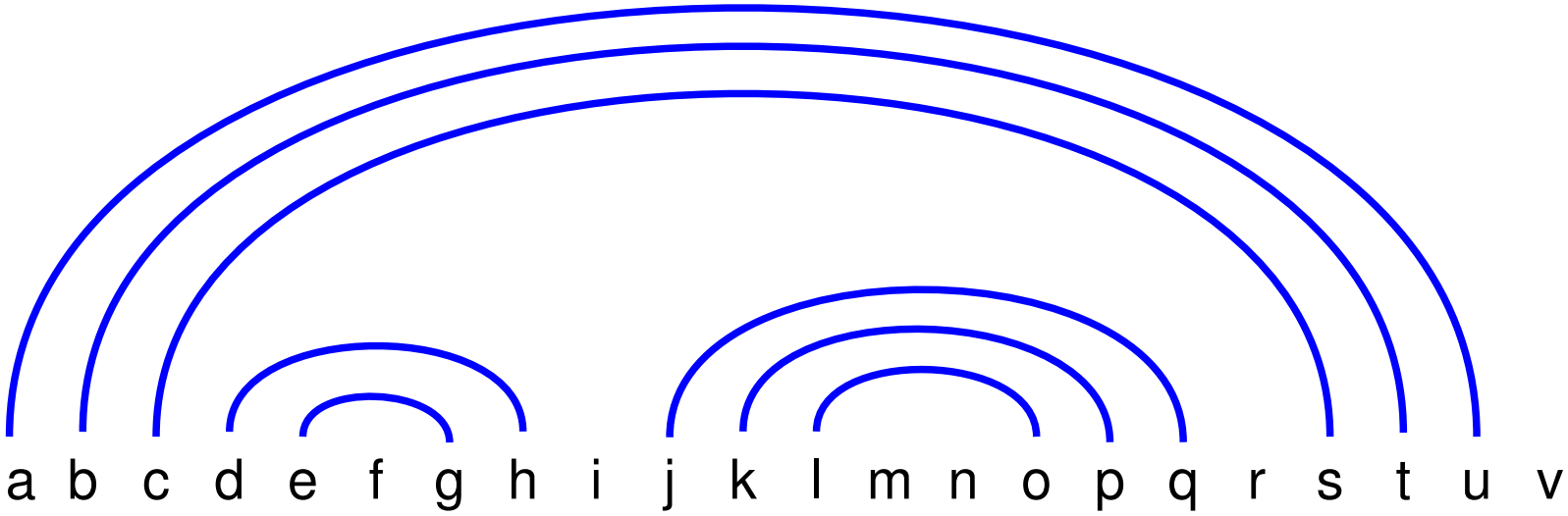
1. **Validité** pour toutes les structures cibles
2. **Stabilité** (energie-libre faible, comparable . . .) des structures cibles
3. **Composition contrainte :** (contenu prescrit en GC), motifs exclus/forcés . . .

Méthode réc. (prog. dyn.) : Compter puis engendrer séquences **valides** uniformes

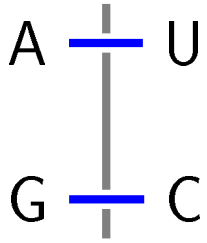
Compter les séquences valides : Watson-Crick + structure unique



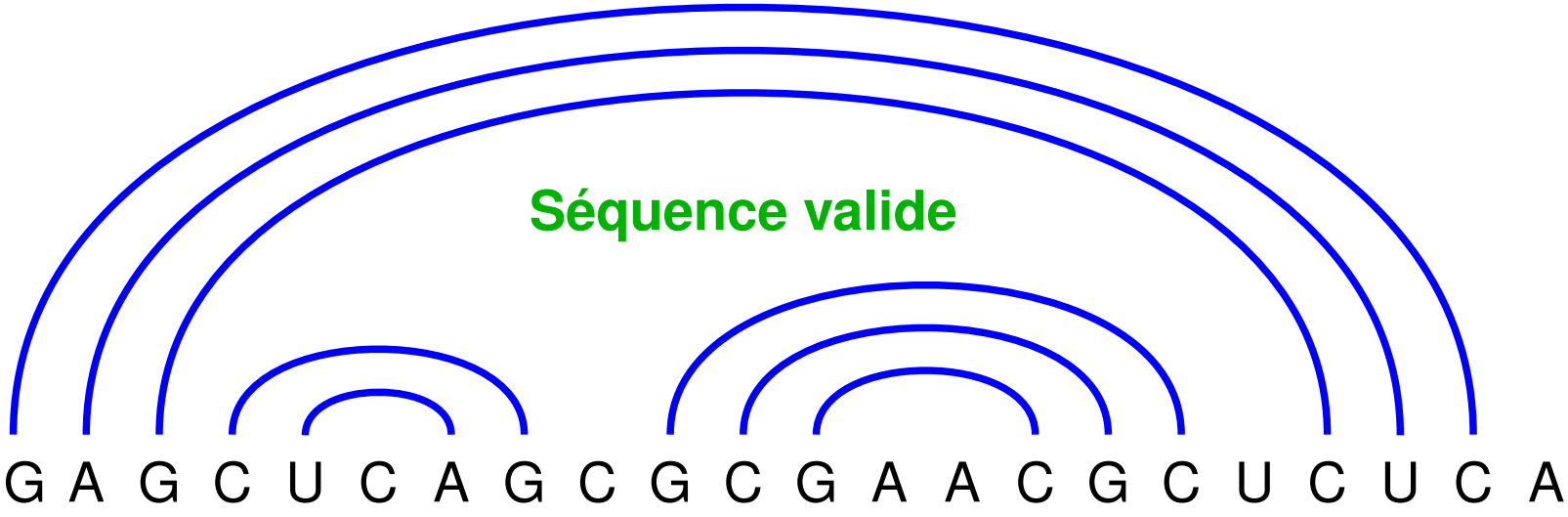
Paires de Bases (PB) valides = **Watson-Crick** seulement



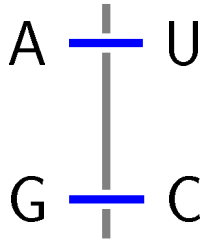
Compter les séquences valides : Watson-Crick + structure unique



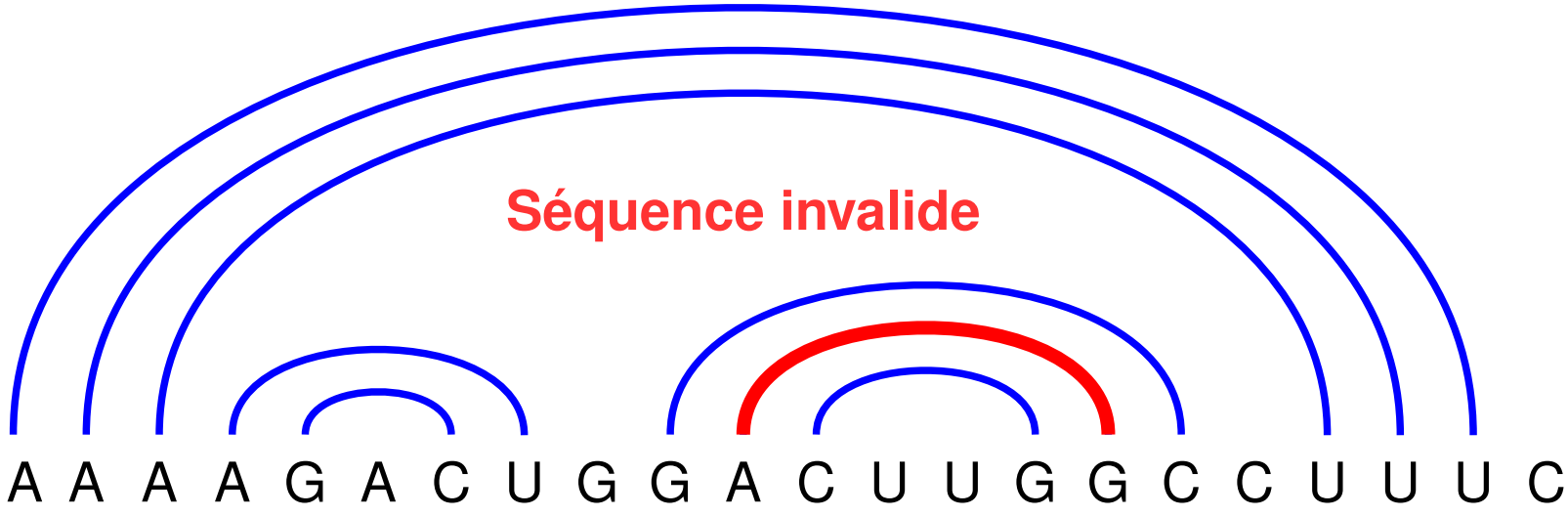
Paires de Bases (PB) valides = **Watson-Crick** seulement



Compter les séquences valides : Watson-Crick + structure unique

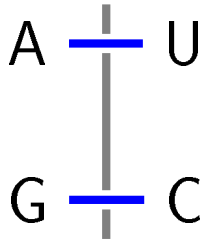


Paires de Bases (PB) valides = **Watson-Crick** seulement

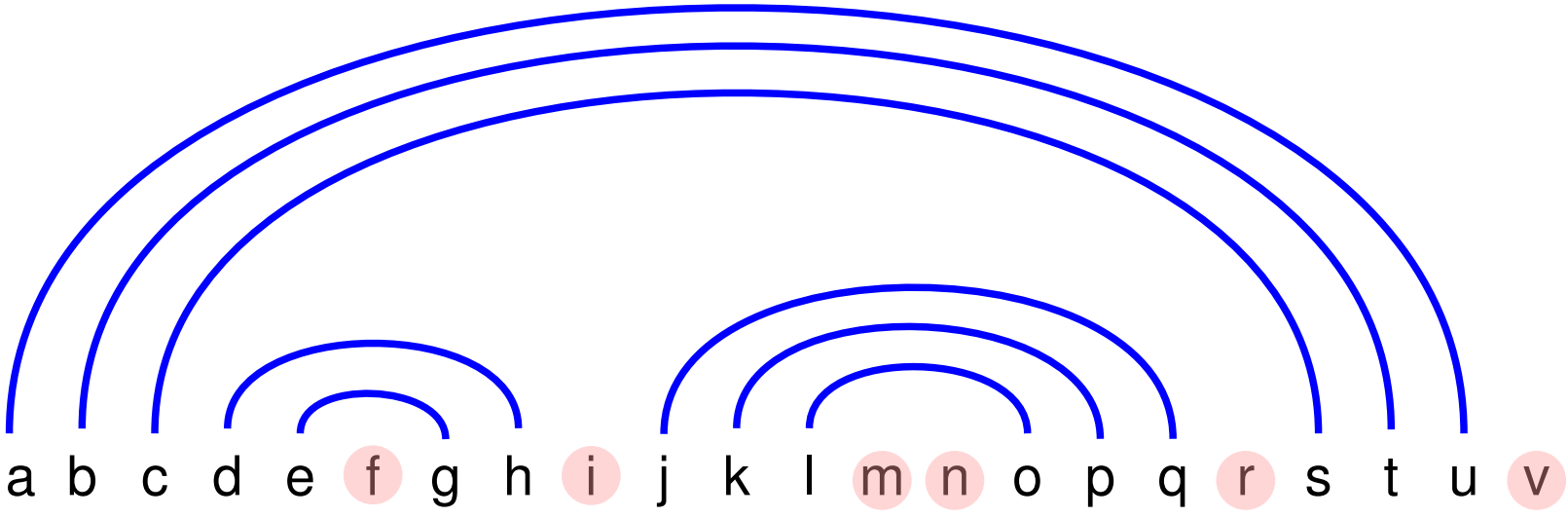


Question : Combien de séquences **valides** ?

Compter les séquences valides : Watson-Crick + structure unique

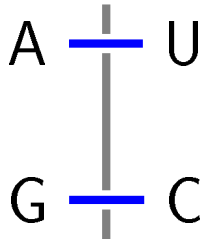


Paires de Bases (PB) valides = **Watson-Crick** seulement

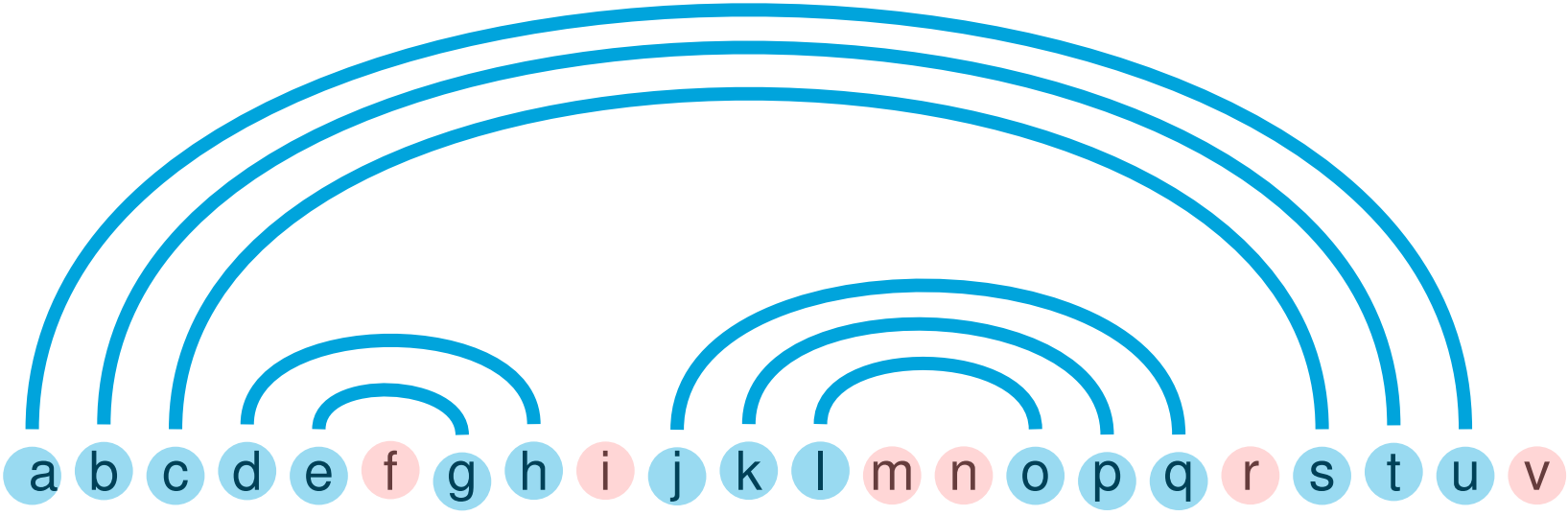


Question : Combien de séquences **valides** ?

Compter les séquences valides : Watson-Crick + structure unique

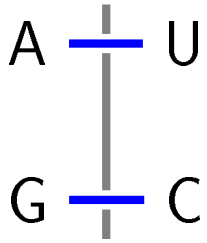


Paires de Bases (PB) valides = **Watson-Crick** seulement

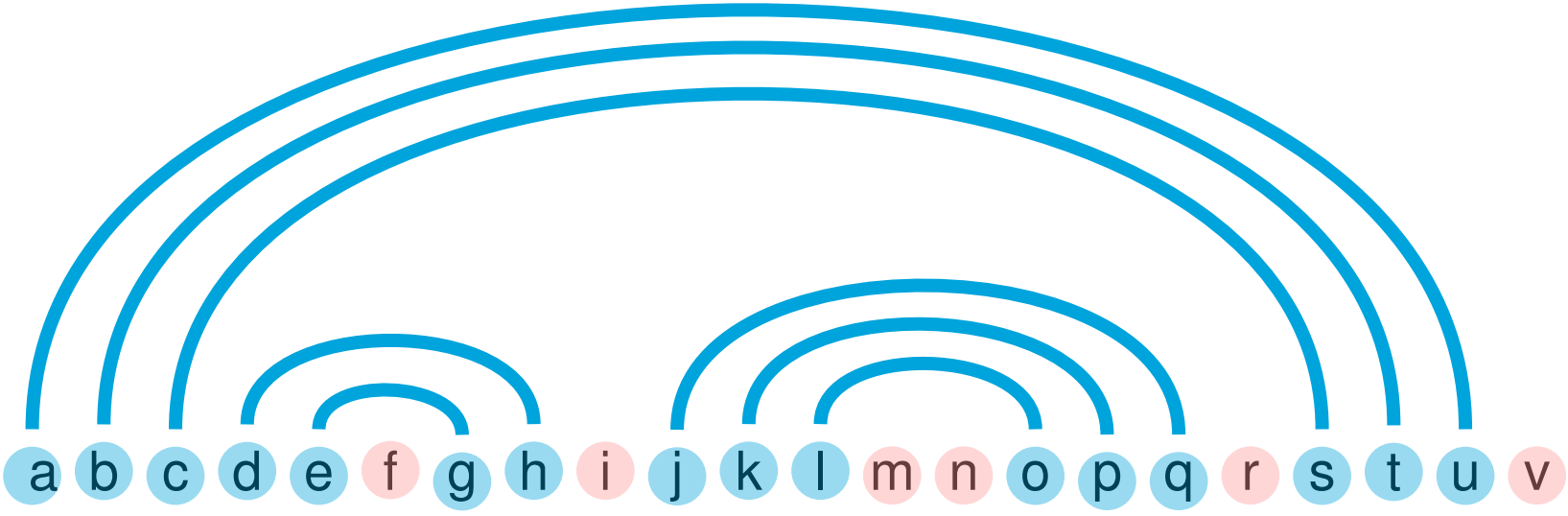


Question : Combien de séquences **valides** ?

Compter les séquences valides : Watson-Crick + structure unique



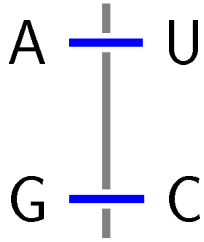
Paires de Bases (PB) valides = **Watson-Crick** seulement



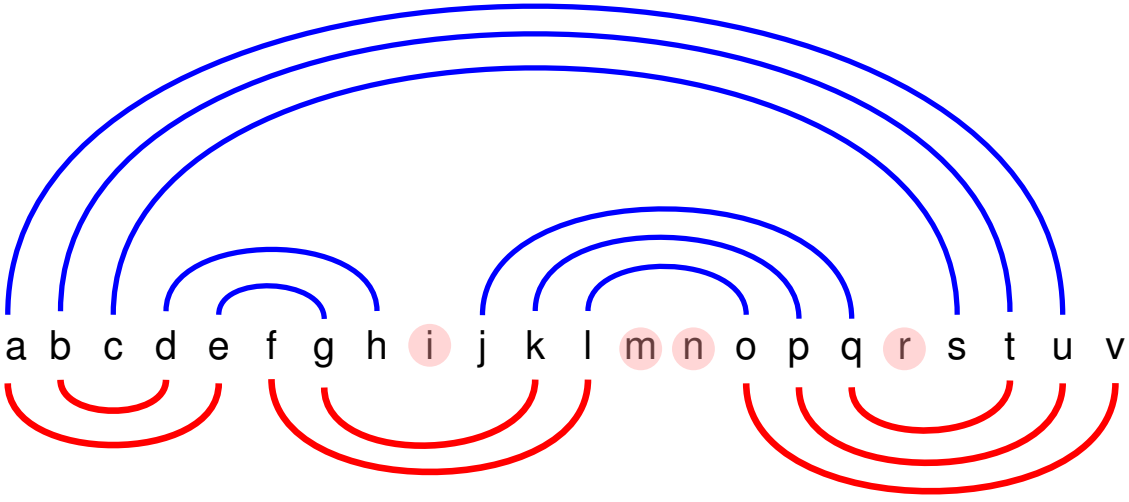
Question : Combien de séquences **valides** ?

Réponse : $4^{\#\text{Paires}} \times 4^{\#\text{Non-appariées}} \rightarrow 268\,435\,456$

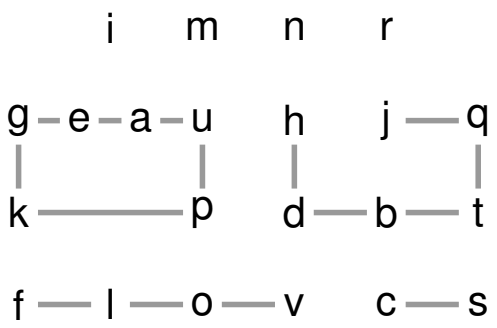
Compter les séquences valides : Watson-Crick + 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



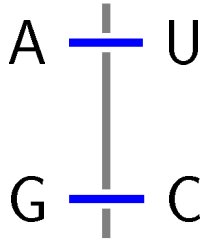
Graphe de dépendance :
Cycles + Chemins



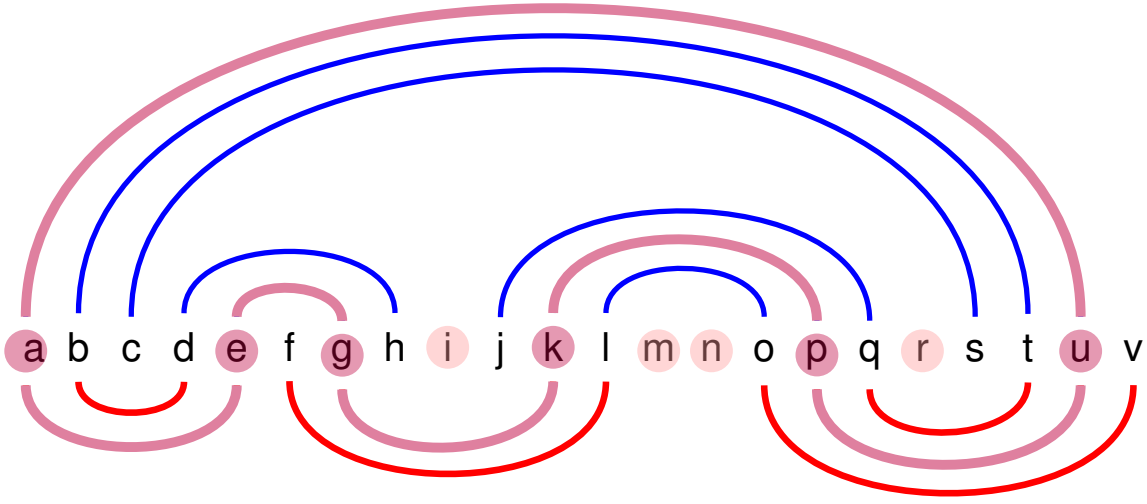
Question : Combien de séquences **valides** ?

Réponse : $\neq \varnothing$ (graphes de dépendance ET des paires valides **bipartis**)

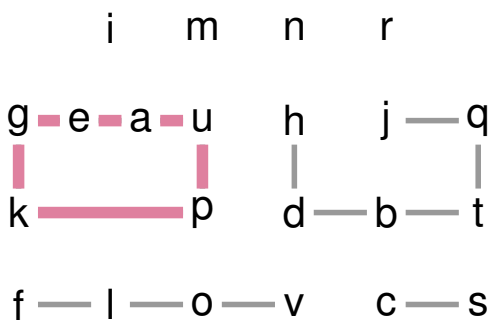
Compter les séquences valides : Watson-Crick + 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



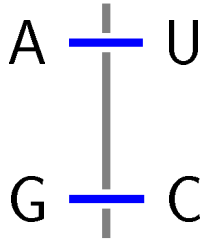
Graphe de dépendance :
Cycles + Chemins



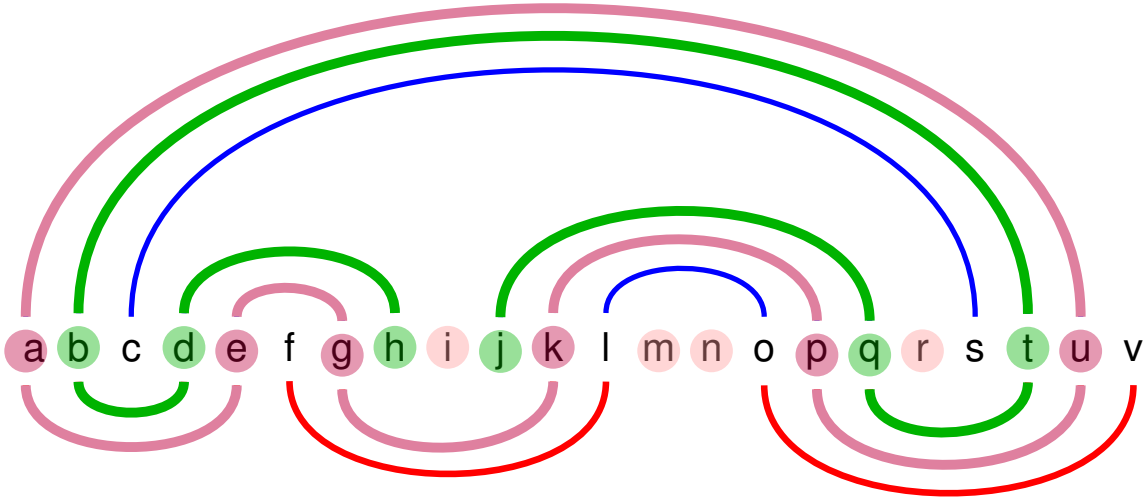
Question : Combien de séquences **valides** ?

Réponse : $\neq \varnothing$ (graphes de dépendance ET des paires valides **bipartis**)

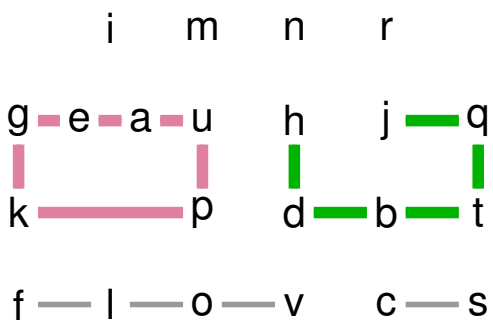
Compter les séquences valides : Watson-Crick + 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



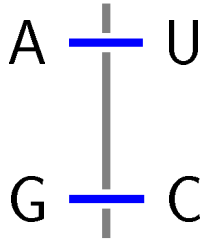
Graphe de dépendance :
Cycles + Chemins



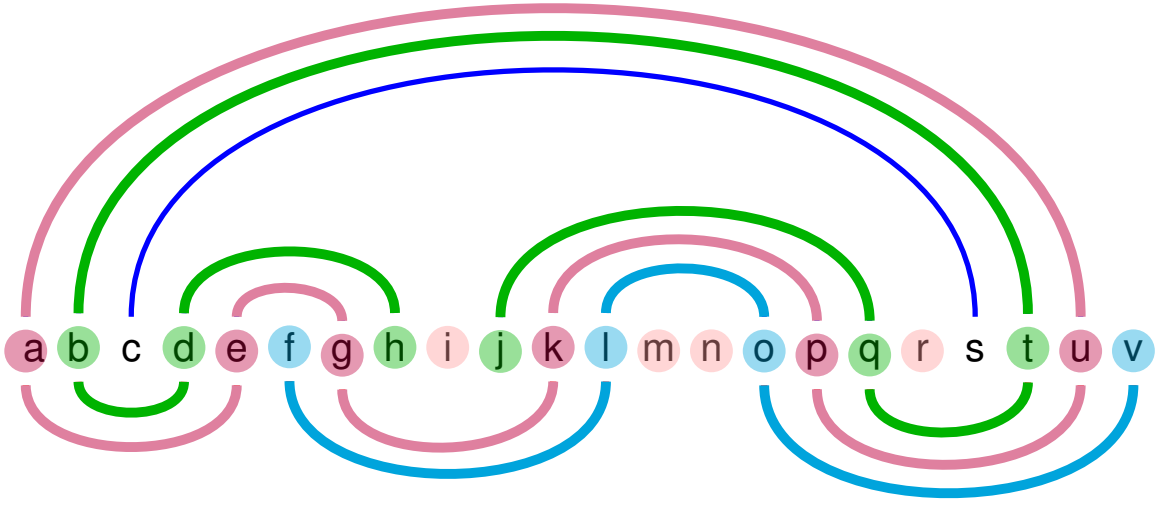
Question : Combien de séquences **valides** ?

Réponse : $\neq \varnothing$ (graphes de dépendance ET des paires valides **bipartis**)

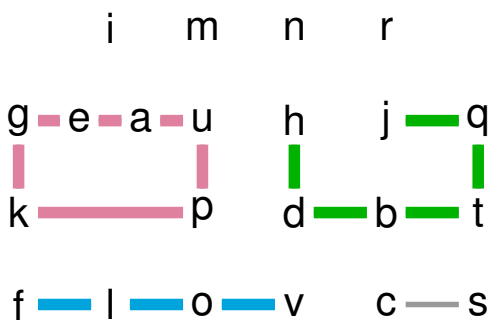
Compter les séquences valides : Watson-Crick + 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



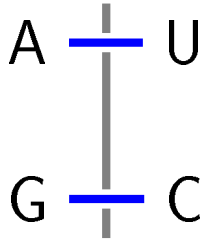
Graphe de dépendance :
Cycles + Chemins



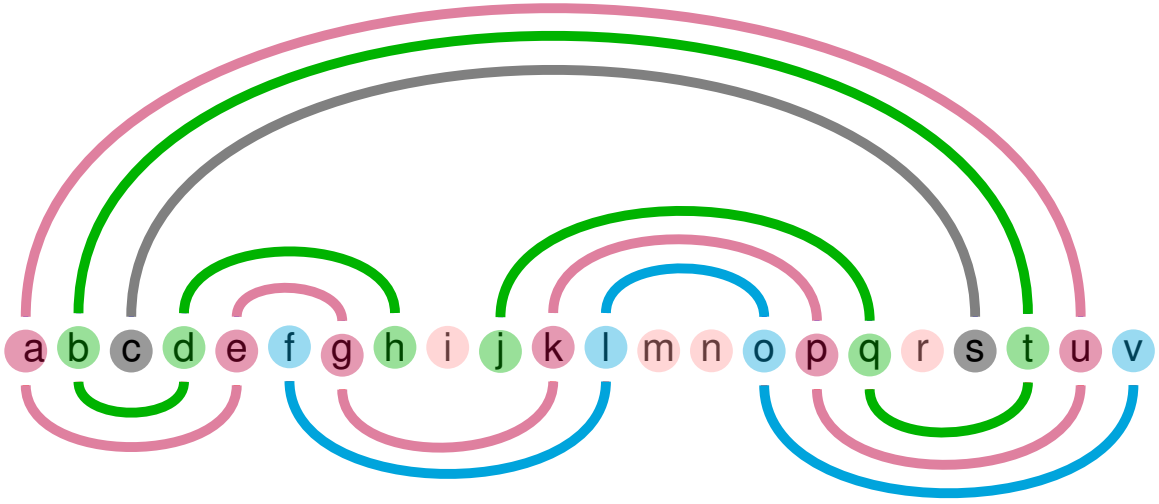
Question : Combien de séquences **valides** ?

Réponse : $\neq \varnothing$ (graphes de dépendance ET des paires valides **bipartis**)

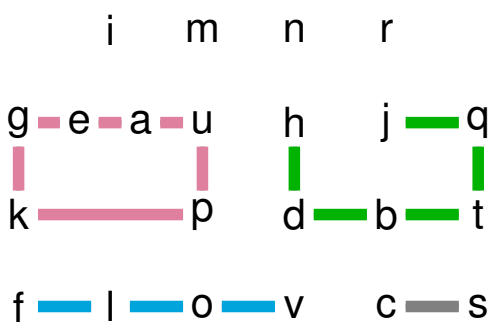
Compter les séquences valides : Watson-Crick + 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



Graphe de dépendance :
Cycles + Chemins

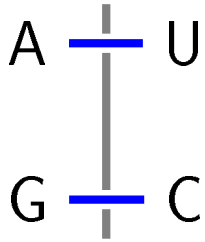


Question : Combien de séquences **valides** ?

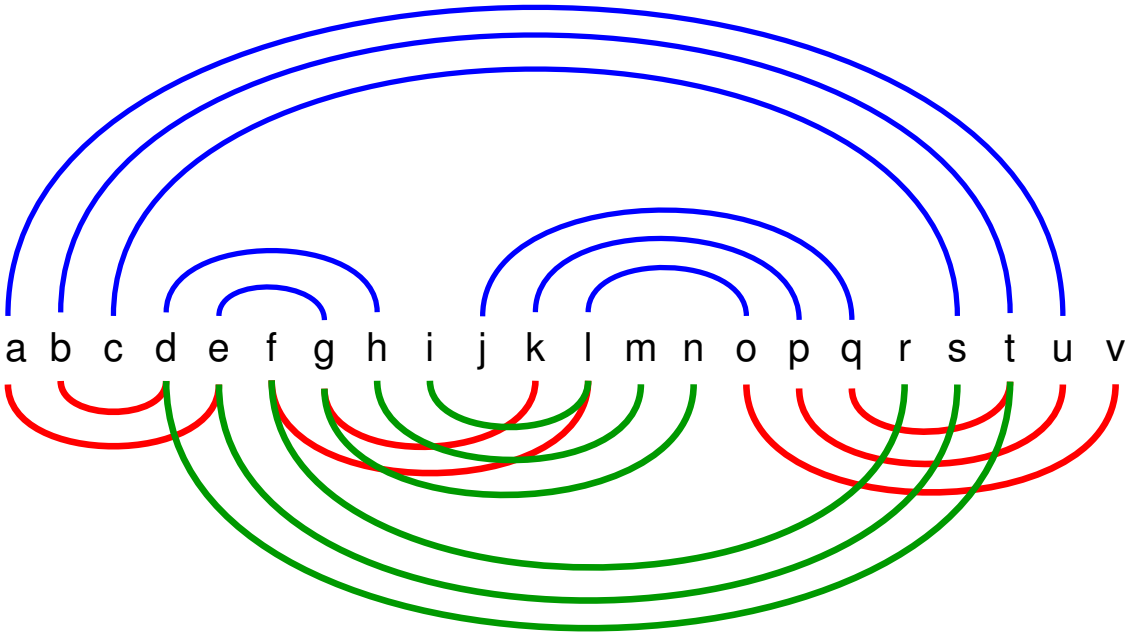
Réponse : $\neq \varnothing$ (graphes de dépendance ET des paires valides **bipartis**)

$4^{\#\text{CCs}} \rightarrow 65\,536$

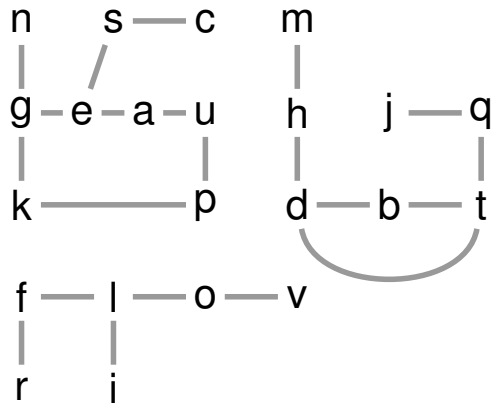
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



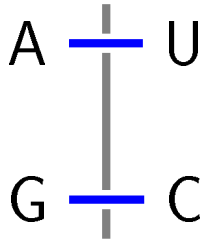
Graphe de dépendance :
Cycles, Chemins, Arbres...



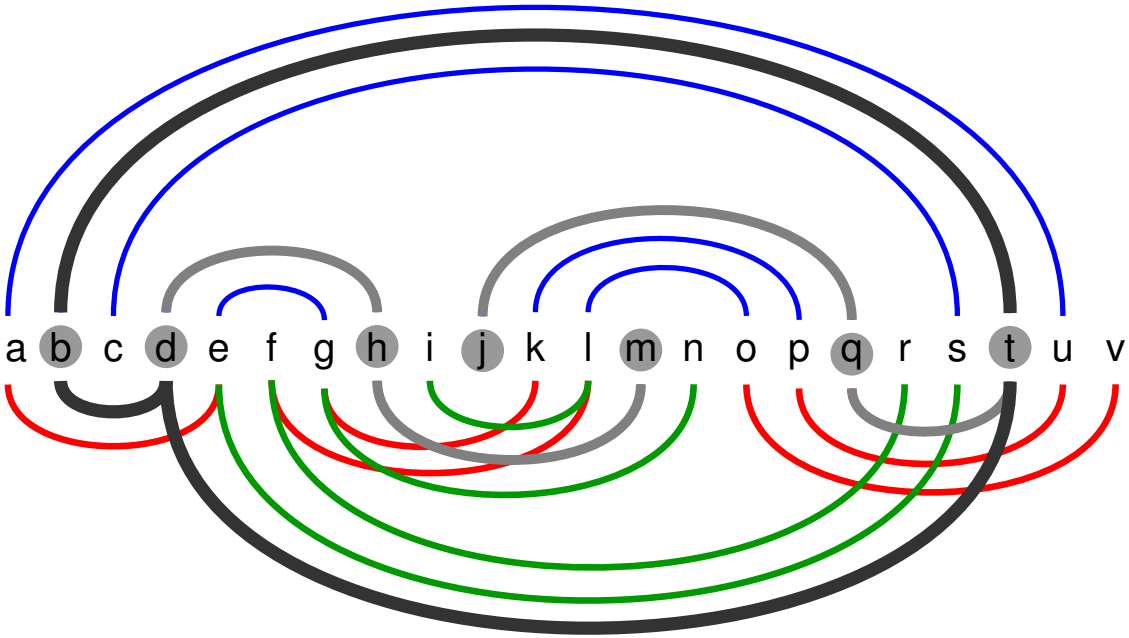
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#CCs} = 64$

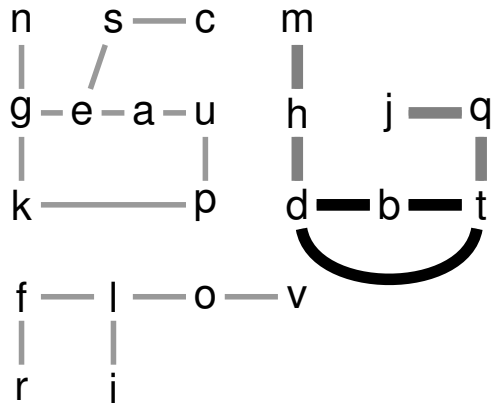
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



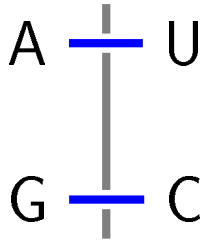
Graphe de dépendance :
Cycles, Chemins, Arbres...



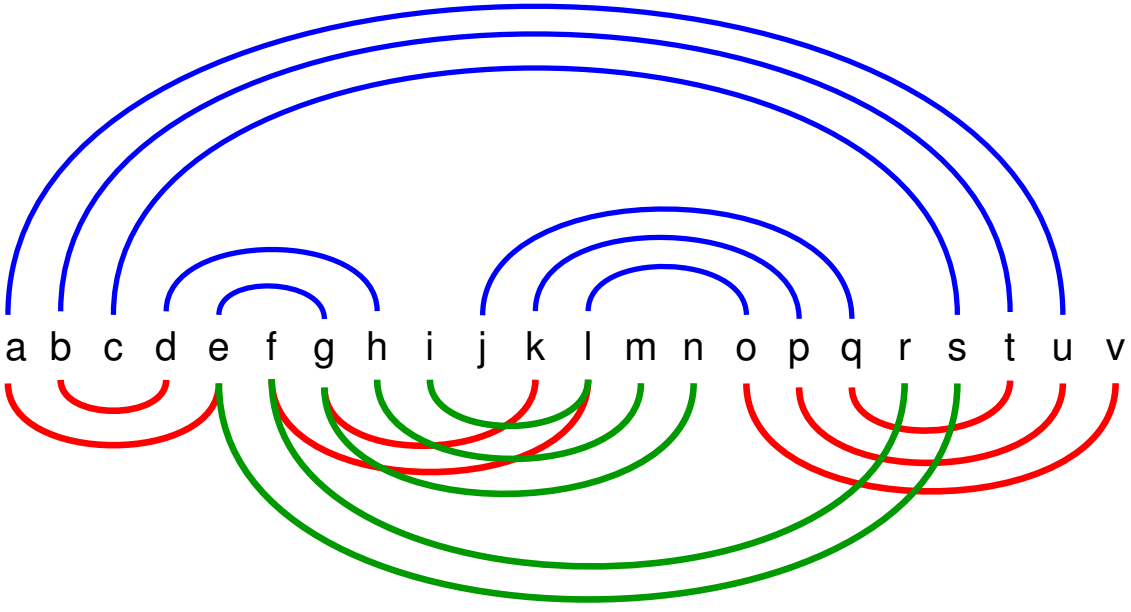
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#CCs} = 64$

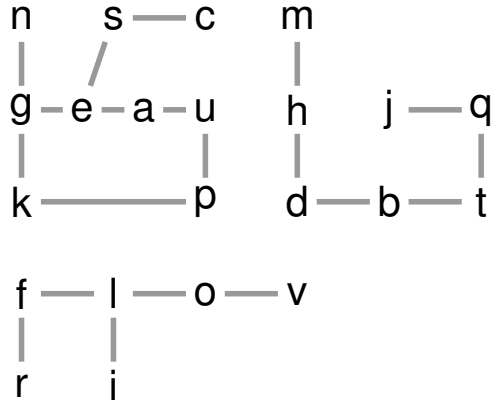
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



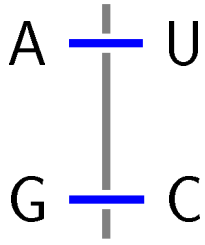
Graphe de dépendance :
Cycles, Chemins, Arbres...



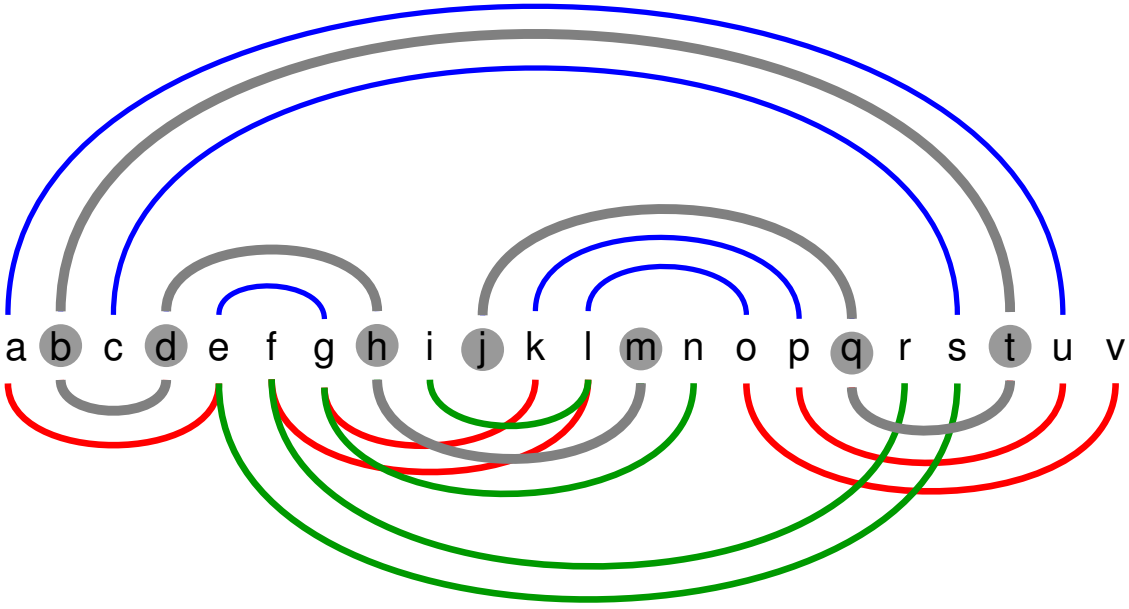
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#CCs} = 64$

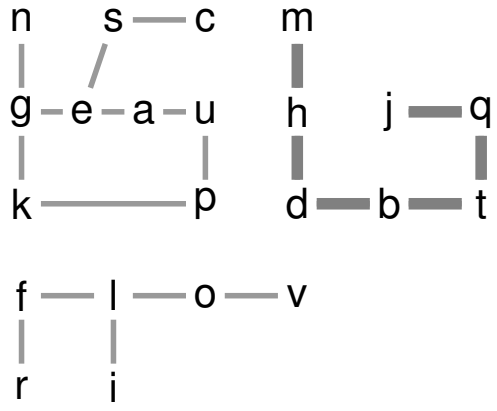
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



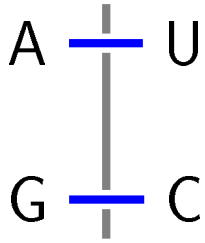
Graphe de dépendance :
Cycles, Chemins, Arbres...



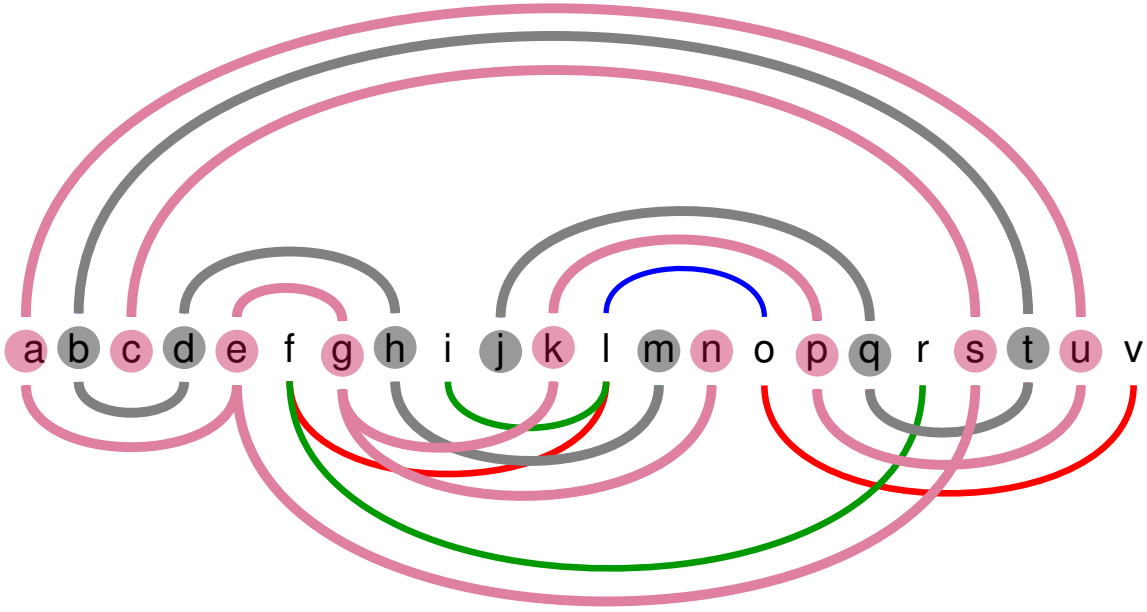
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#CCs} = 64$

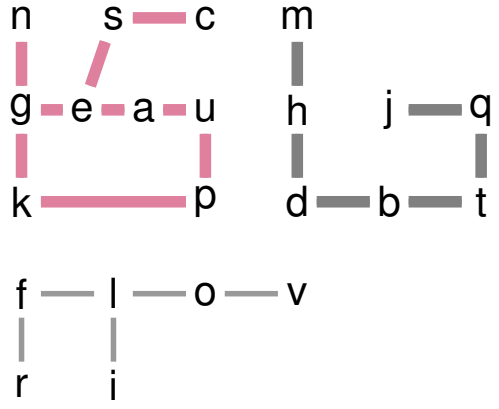
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



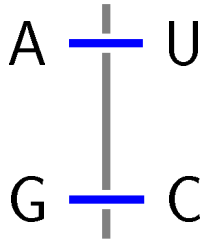
Graphe de dépendance :
Cycles, Chemins, Arbres...



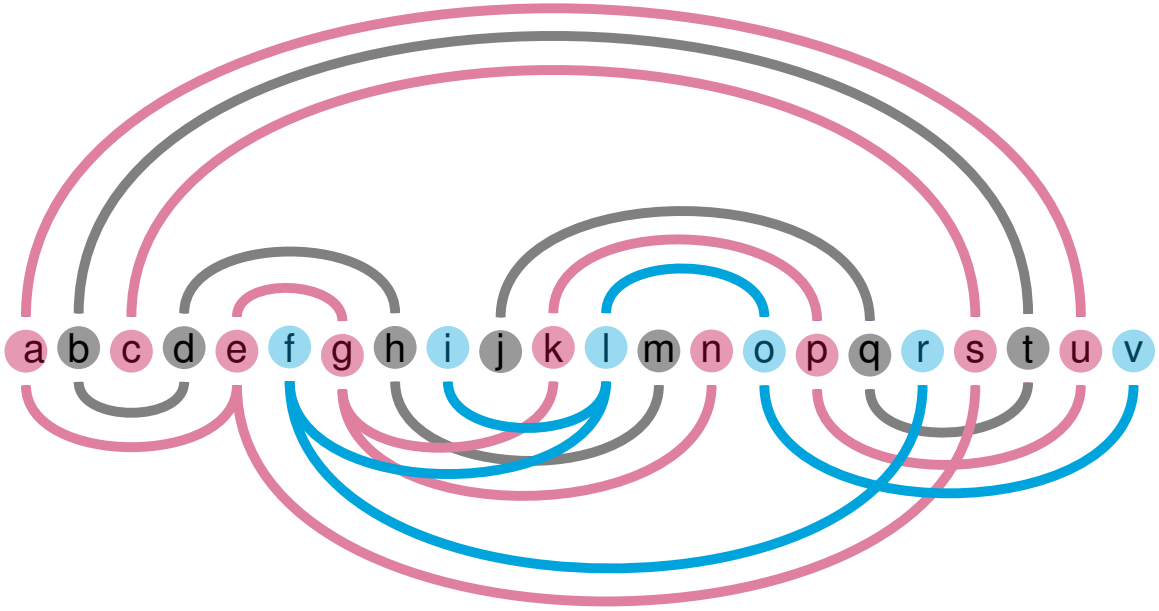
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#CCs} = 64$

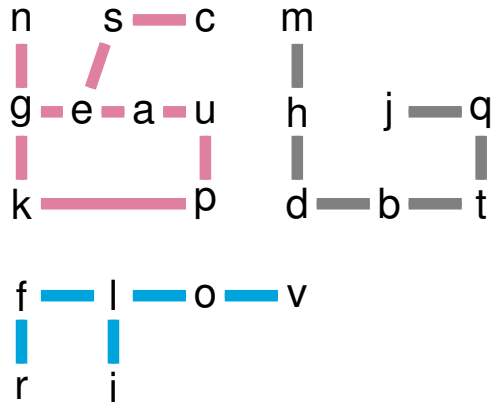
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases (PB) valides = **Watson-Crick** seulement



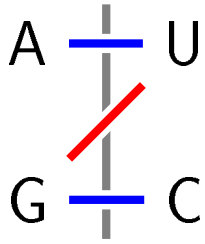
Graphe de dépendance :
Cycles, Chemins, Arbres...



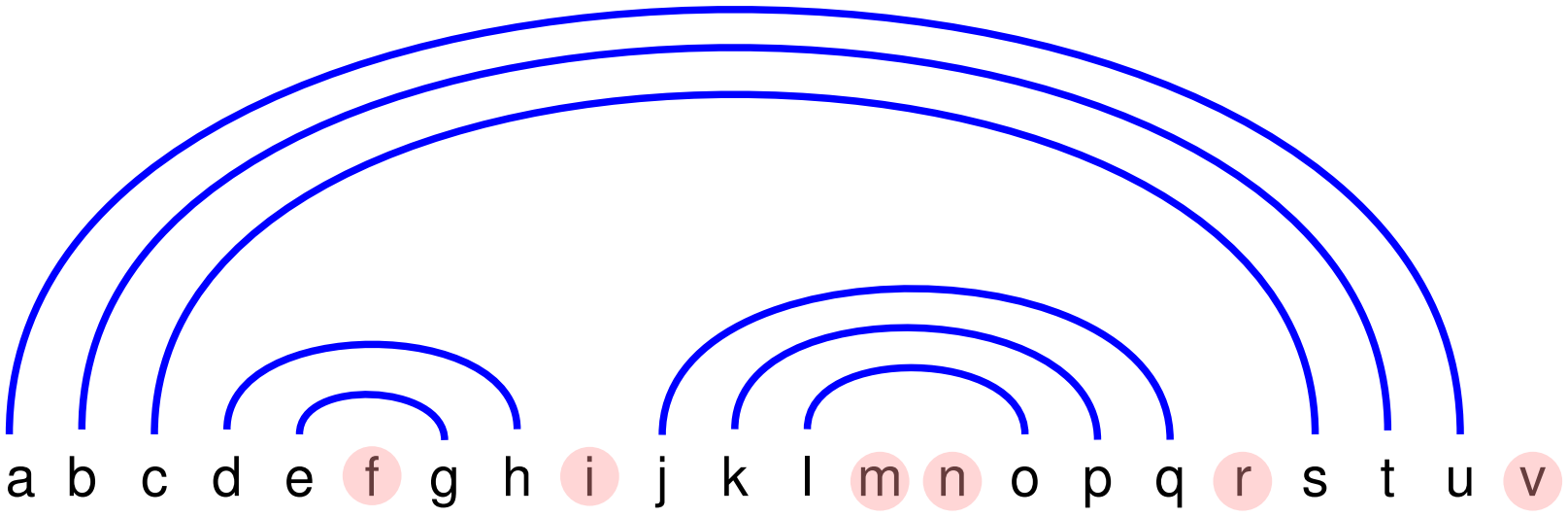
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow 4^{\#\text{CCs}} = 64$

Compter les séquences valides : WC/Wobble + unique structure



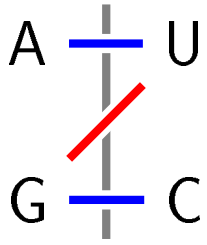
Paires de Bases (PB) valides = autorise les paires **Wobble**



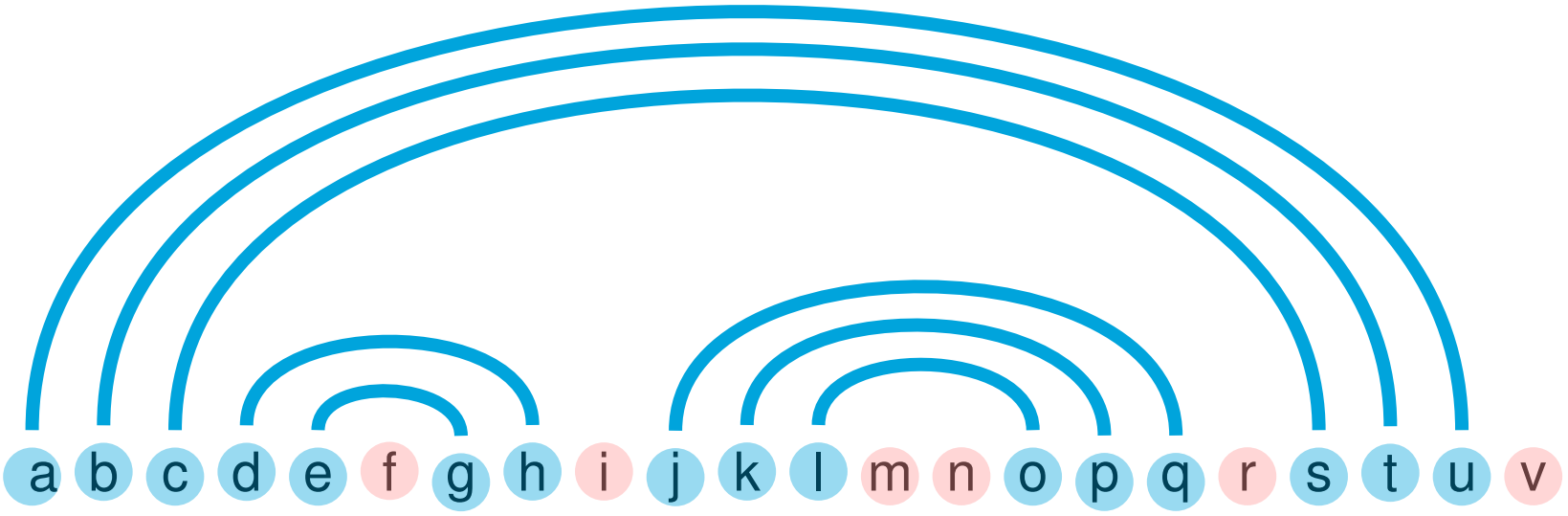
Question : Combien de séquences **valides** ?

Réponse : $4^{\#\text{Non-appariées}} \times 6^{\#\text{PBs}} \rightarrow 6\,879\,707\,136$

Compter les séquences valides : WC/Wobble + unique structure



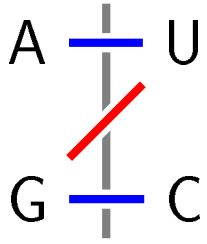
Paires de Bases (PB) valides = autorise les paires **Wobble**



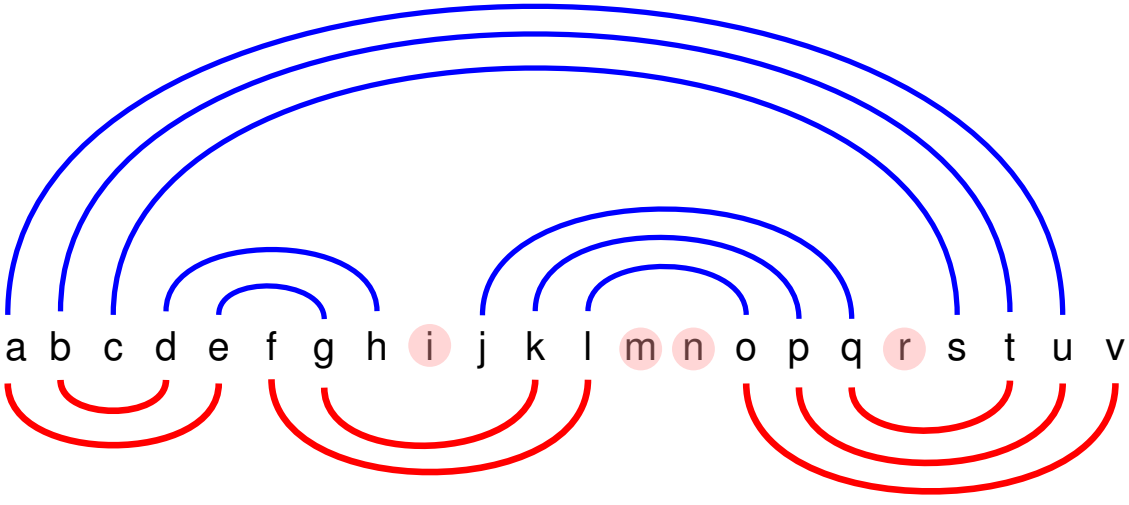
Question : Combien de séquences **valides** ?

Réponse : $4^{\#\text{Non-appariées}} \times 6^{\#\text{PBs}} \rightarrow 6\,879\,707\,136$

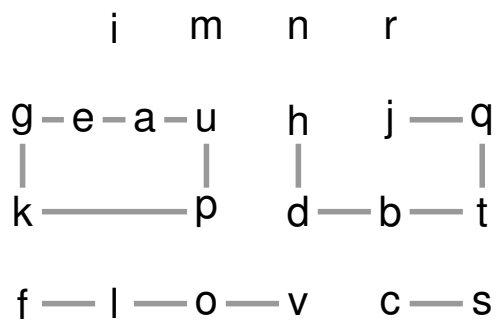
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



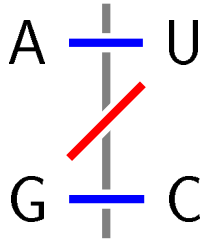
Graphe de dépendance :
Cycles + Chemins



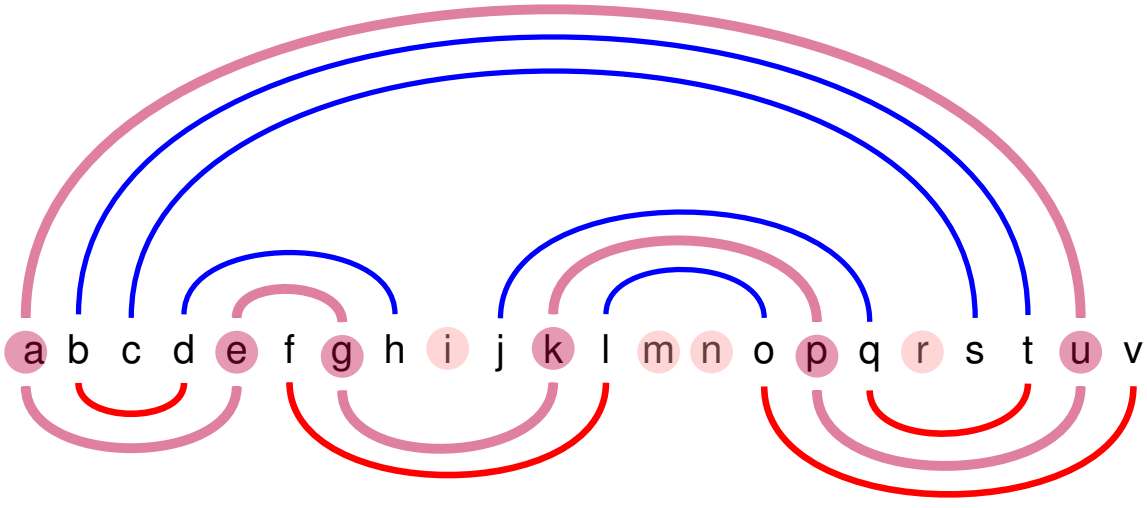
Question : Combien de séquences **valides** ?

Réponses : $\neq \varnothing$! (graphes de dépendance et paires de bases tous deux **bipartis**)

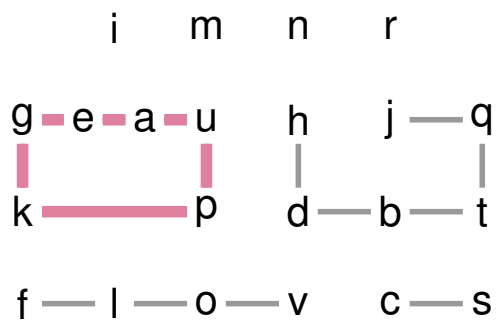
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



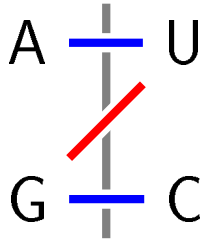
Graphe de dépendance :
Cycles + Chemins



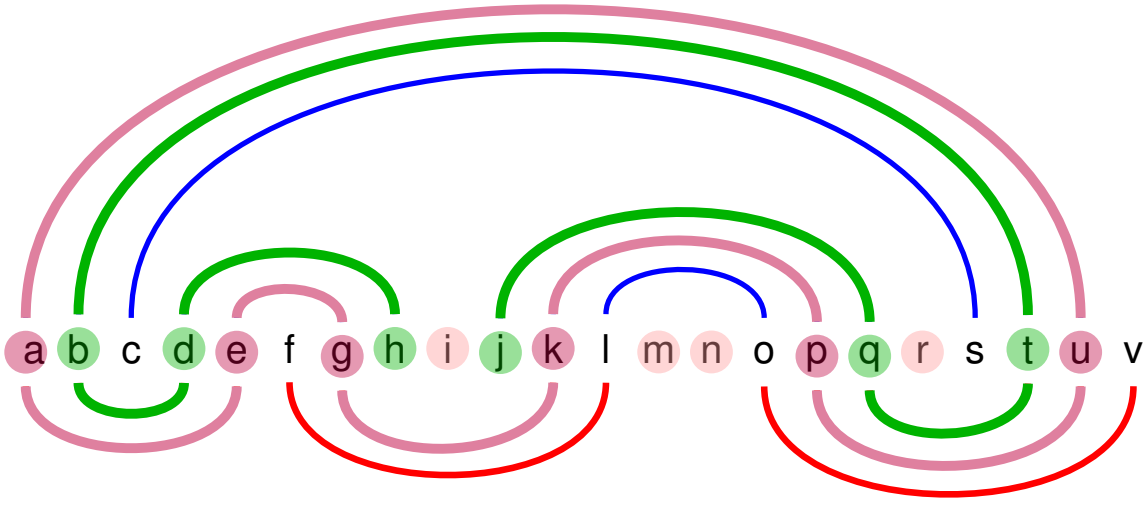
Question : Combien de séquences **valides** ?

Réponses : $\neq \varnothing$! (graphes de dépendance et paires de bases tous deux **bipartis**)

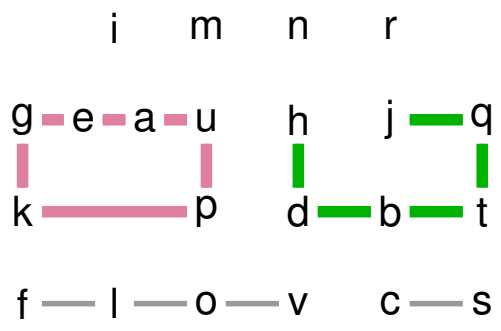
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



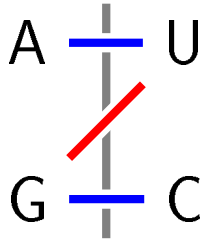
Graphe de dépendance :
Cycles + Chemins



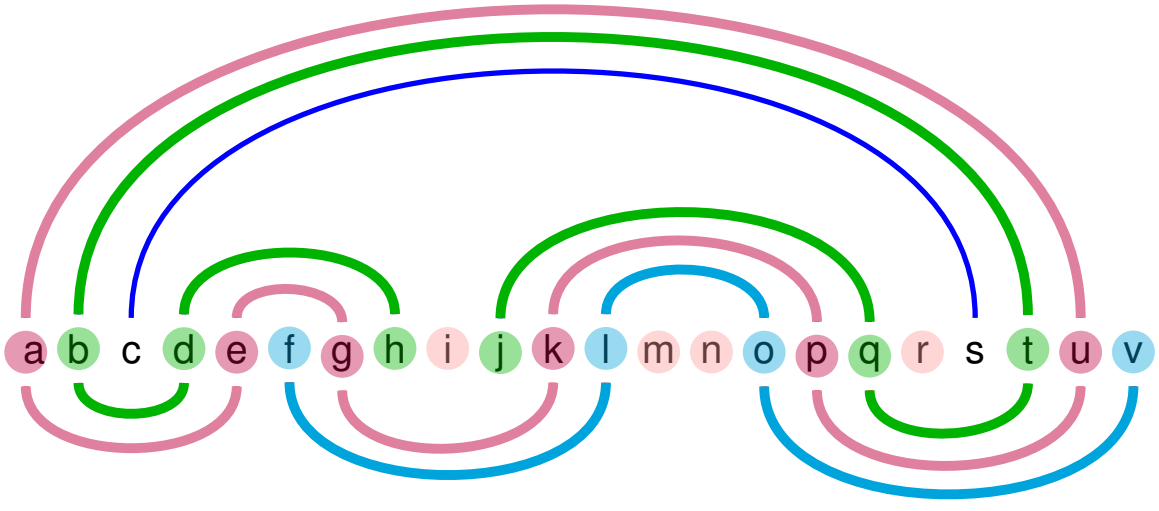
Question : Combien de séquences **valides** ?

Réponses : $\neq \varnothing$! (graphes de dépendance et paires de bases tous deux **bipartis**)

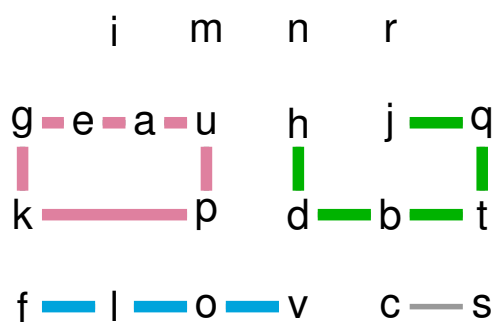
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



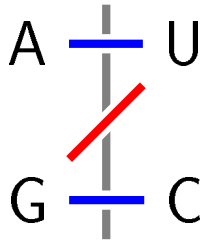
Graphe de dépendance :
Cycles + Chemins



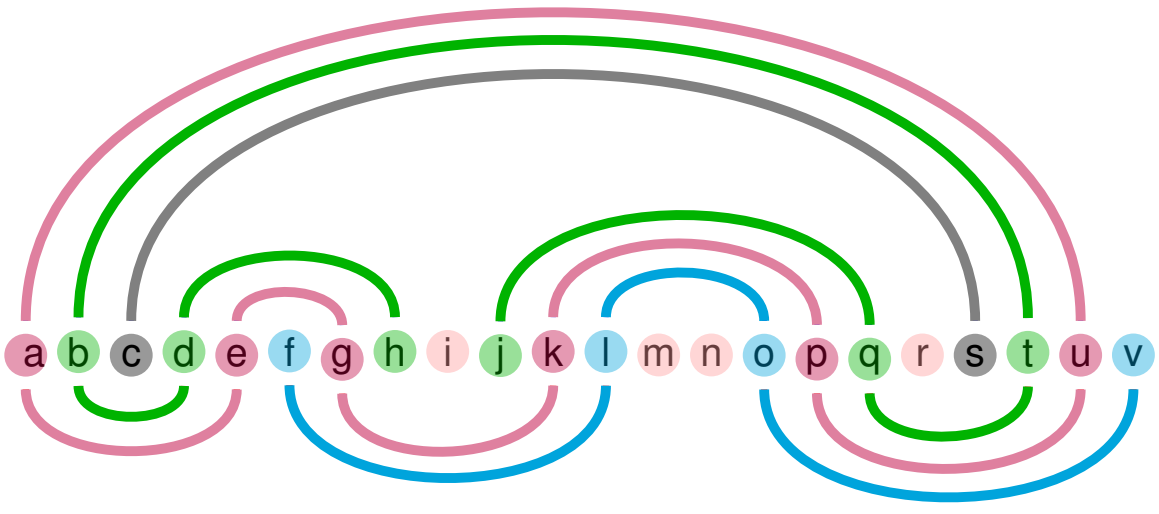
Question : Combien de séquences **valides** ?

Réponses : $\neq \varnothing$! (graphes de dépendance et paires de bases tous deux **bipartis**)

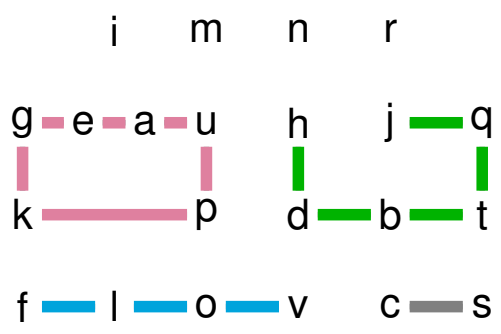
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



Graphe de dépendance :
Cycles + Chemins



Question : Combien de séquences **valides** ?

Réponses : $\neq \varnothing$! (graphes de dépendance et paires de bases tous deux **bipartis**)

$$\#\text{Designs}(G) = \prod_{c \in CC(G)} \#\text{Designs}(cc)$$

Compter les séquences valides pour les chemins et cycles

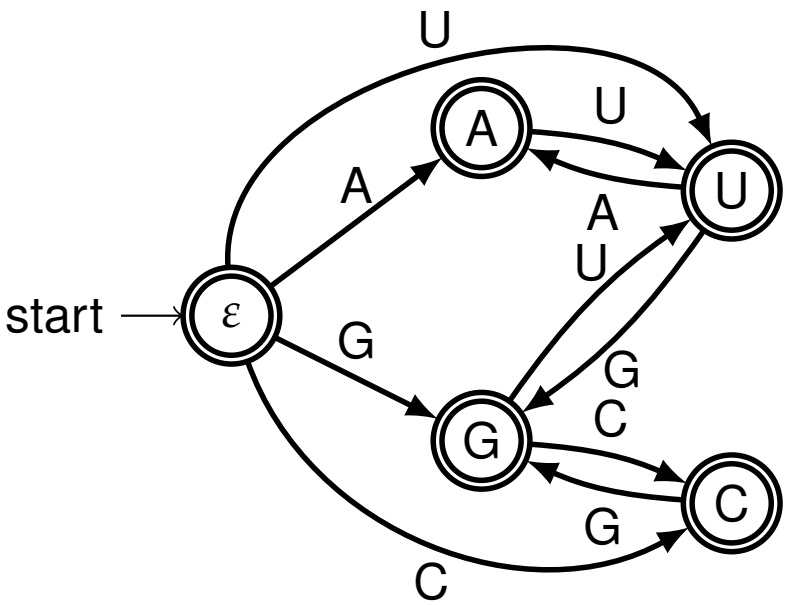
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

Compter les séquences valides pour les chemins et cycles

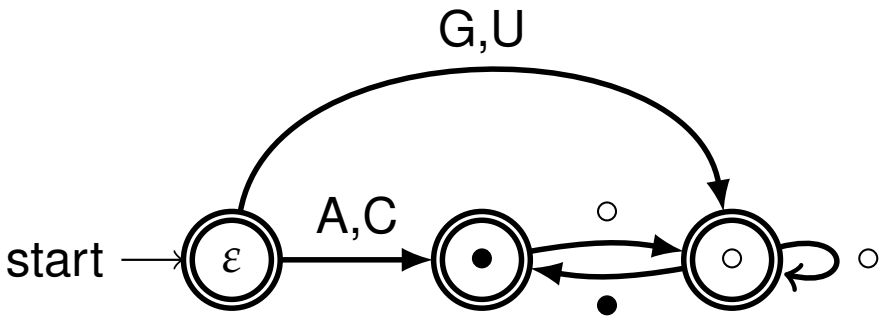
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C / G $\leftrightarrow$ U

Compter les séquences valides pour les chemins et cycles

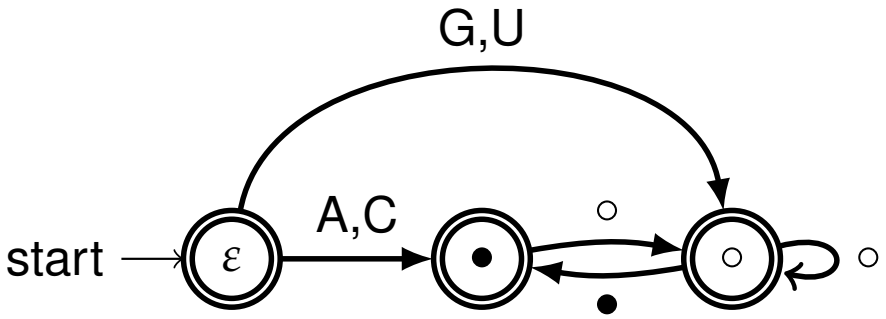
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

$$m_\bullet(n) = m_\circ(n-1)$$

Compter les séquences valides pour les chemins et cycles

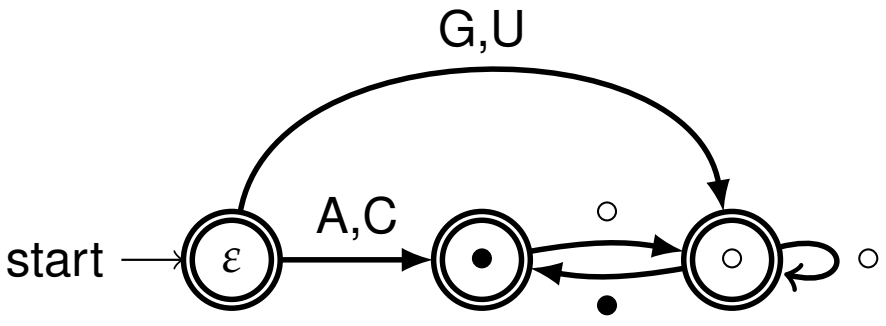
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

$$\begin{aligned} m_\bullet(n) &= m_\circ(n-1) \\ m_\circ(n) &= m_\circ(n-1) + m_\bullet(n-1) \\ &= m_\circ(n-1) + m_\circ(n-2) \\ &= \mathcal{F}(n+2) \end{aligned}$$

Compter les séquences valides pour les chemins et cycles

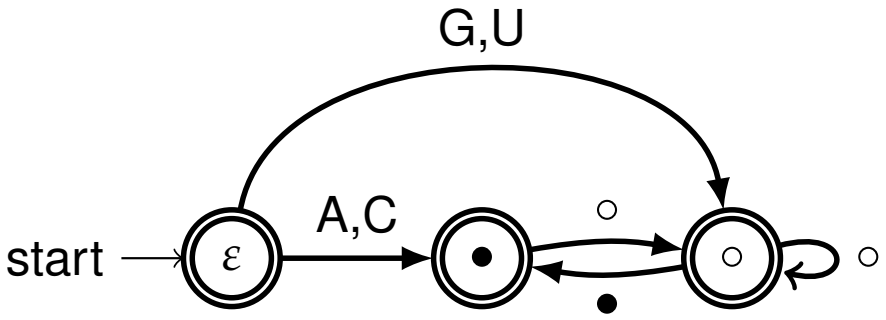
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

$$\begin{aligned} m_\bullet(n) &= m_\circ(n-1) \\ m_\circ(n) &= m_\circ(n-1) + m_\bullet(n-1) \\ &= m_\circ(n-1) + m_\circ(n-2) \\ &= \mathcal{F}(n+2) \end{aligned}$$

(Car $m_\circ(0) = 1$ and $m_\circ(1) = 2$)

Compter les séquences valides pour les chemins et cycles

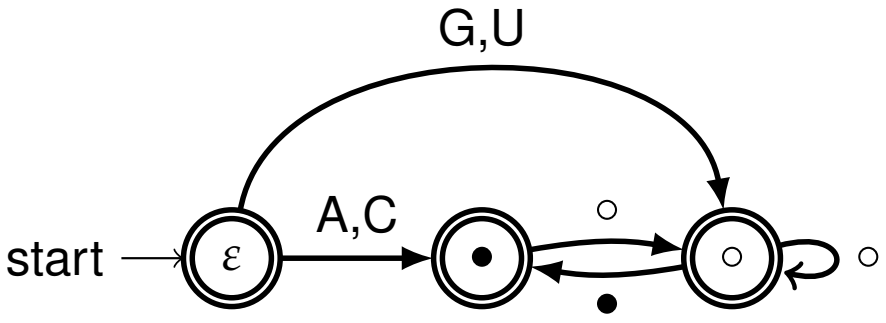
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les chemins : Un simple automate . . .



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

$$\begin{aligned} m_\bullet(n) &= m_\circ(n-1) \\ m_\circ(n) &= m_\circ(n-1) + m_\bullet(n-1) \\ &= m_\circ(n-1) + m_\circ(n-2) \\ &= \mathcal{F}(n+2) \end{aligned}$$

(Car $m_\circ(0) = 1$ and $m_\circ(1) = 2$)

$$p(n) := m_\varepsilon(n) = 2\,m_\bullet(n-1) + 2\,m_\circ(n-1) = 2(\mathcal{F}(n) + \mathcal{F}(n+1)) = 2\mathcal{F}(n+2)$$

Compter les séquences valides pour les chemins et cycles

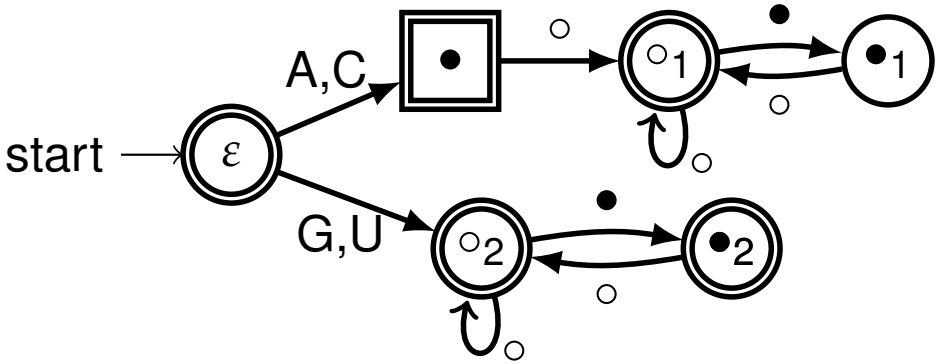
$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

Pour les cycles : *à peine plus compliqué . . .*



Remarque : Symétrie A $\leftrightarrow$ C/G $\leftrightarrow$ U

$$m_{\circ_2}(n) = \mathcal{F}(n+2)$$
$$m_{\circ_1}(n) = \mathcal{F}(n+1)$$

(Car $m_{\circ_1}(0) = 1$ and $m_{\circ_1}(1) = 1$)

$$\begin{aligned} c(n) := m_\varepsilon(n) &= 2\,m_{\circ_1}(n-2) + 2\,m_{\circ_2}(n-1) \\ &= 2(\mathcal{F}(n-1) + \mathcal{F}(n+1)) = 2\,\mathcal{F}(n) + 4\,\mathcal{F}(n-1) \end{aligned}$$

Compter les séquences valides pour deux structures

$p(n)$: Nombre de séquences valides pour un chemin de taille n.
$c(n)$: Nombre de séquences valides pour un cycle de taille n.

Théorème (#Séquences valides pour chemins et cycles)

$$p(n) = 2\,\mathcal{F}_{n+2} \qquad \textit{et} \qquad c(n) = 2\,\mathcal{F}_n + 4\,\mathcal{F}_{n-1}$$

où $\mathcal{F}_n$ est le n-ième nombre de Fibonacci.

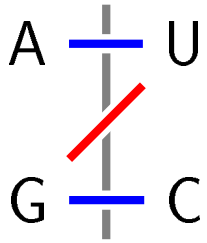
G : Graphe de dépendance issus de 2 structures (degré max ≤ 2).
G décomposé en $\mathcal{P}(G)$ chemins et $\mathcal{C}(G)$ cycles.

Théorème (#Séquences valides pour les graphes de 2-structures)

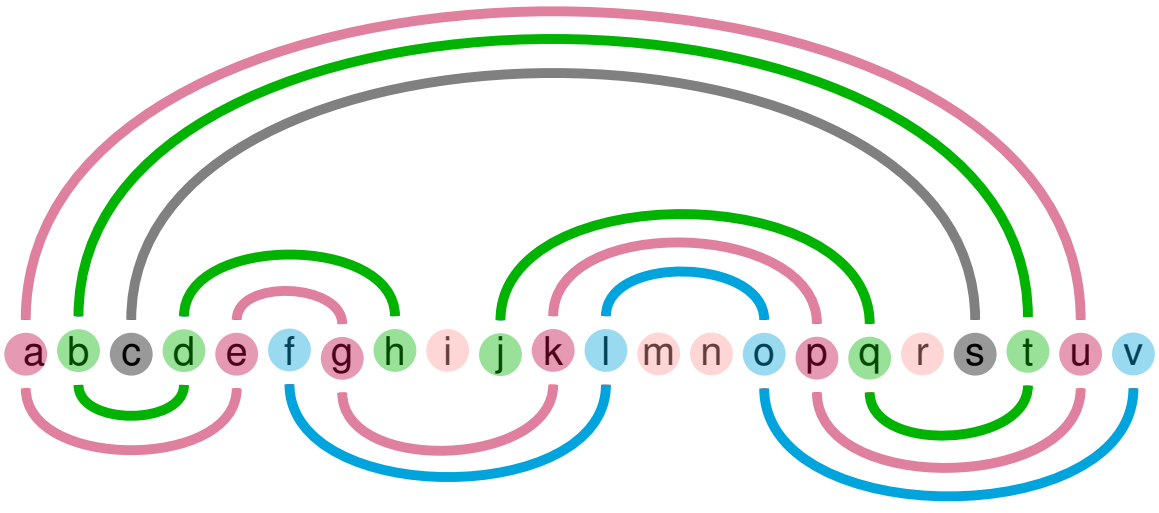
Le nombre $\#\text{Designs}(G)$ de séquences valides pour G est

$$\#\text{Designs}(G) = \prod_{p \in \mathcal{P}(G)} 2\,\mathcal{F}_{|p|+2} \times \prod_{c \in \mathcal{C}(G)} \left(2\,\mathcal{F}_{|c|} + 4\,\mathcal{F}_{|c|-1}\right)$$

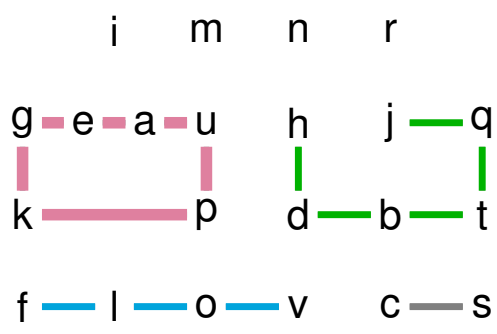
Compter les séquences valides : WC/Wobble + Deux structures



Paires de Bases valides = autorise les paires **Wobble**



Graphe de dépendance :
Cycles + Chemins

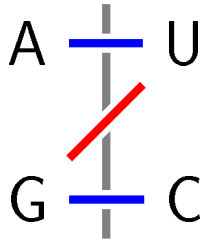


Question : Combien de séquences **valides** ?

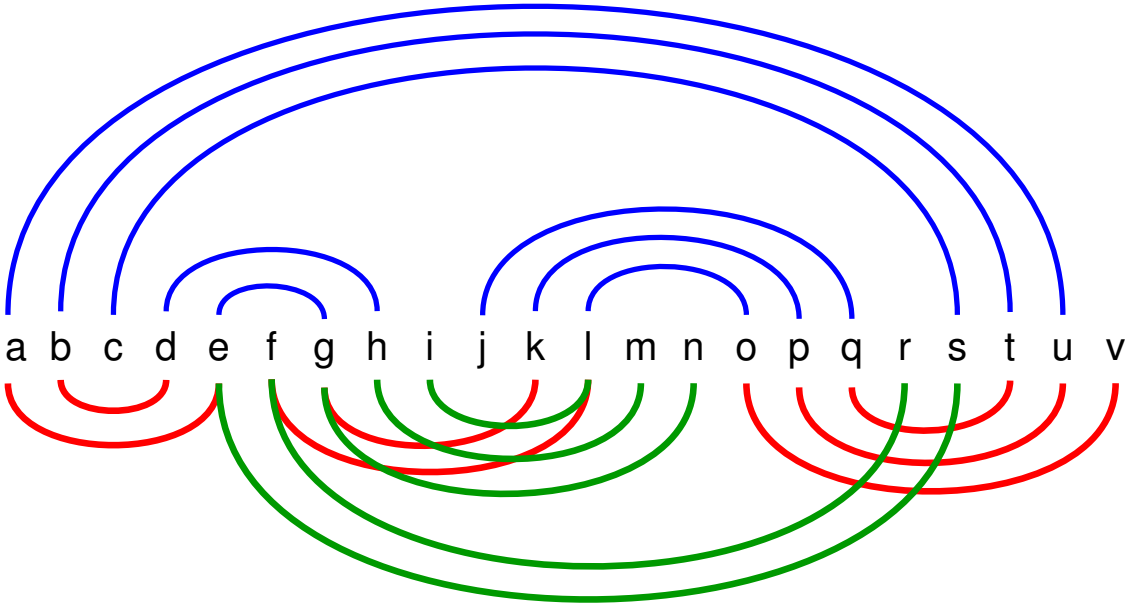
Réponse : $\neq \varnothing$! (graphes PB et dépendance **bipartis**)

$$\#\text{Designs}(G) = \prod_{c \in CC(G)} \#\text{Designs}(cc) = 2\,322\,432$$

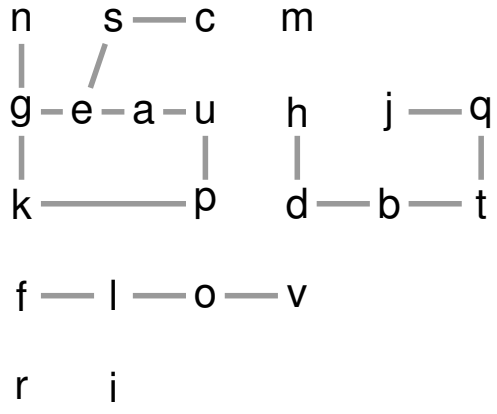
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases valides = autorise les paires **Wobble**



Graphe de dépendance :
Cycles, Chemins, Arbres...



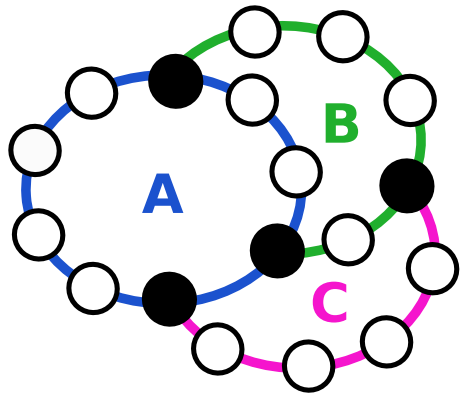
Question : Combien de séquences **valides** ?

Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow$? ? ? ?

Etat de l'art

Approche de Abfalter/Flamm/Stadler 2003 :

- Décomposition *en oreille* du graphe **[Whitney 1932]** *Epluchage du graphe* ;



- **Programmation dynamique :** comptage des #chemins valides pour chaque composante, étant donné des valeurs pour les **ancrages** (noeuds noirs) ;
- **Recombiner** les valeurs pour calculer le nombre de séquences valides.

Complexité : $\Theta(n.4^A)$ où $A =$ #Max. d'ancrages, $A \in \Theta(n)$ dans le cas au pire.

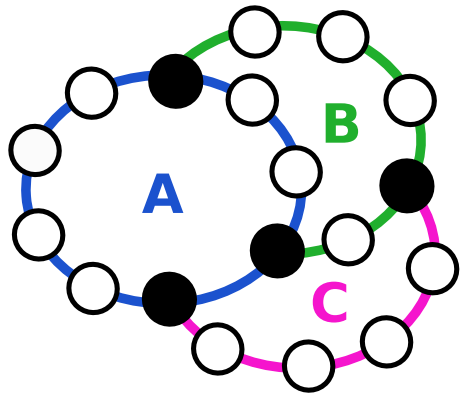
Quelques réactions :

- Est ce bien *optimal* ? Autres algorithmes et paramètres ?
- Quelles extensions possibles ? Génération pondérée (Boltzmann multidim.)
- Est ce bien nécessaire ? **Probablement** car comptage #P-difficile

Etat de l'art

Approche de Abfalter/Flamm/Stadler 2003 :

- Décomposition *en oreille* du graphe **[Whitney 1932]** *Epluchage du graphe* ;



- **Programmation dynamique :** comptage des #chemins valides pour chaque composante, étant donné des valeurs pour les **ancrages** (noeuds noirs) ;
- **Recombiner** les valeurs pour calculer le nombre de séquences valides.

Complexité : $\Theta(n.4^A)$ où $A =$ #Max. d'ancrages, $A \in \Theta(n)$ dans le cas au pire.

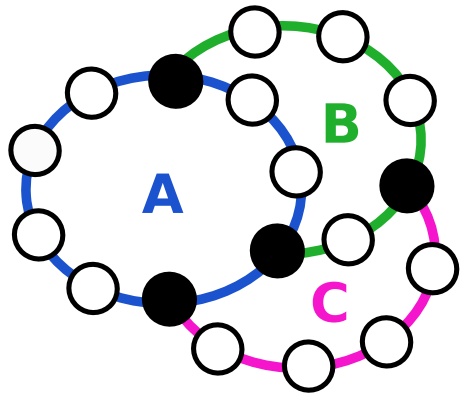
Quelques réactions :

- Est ce bien *optimal* ? Autres algorithmes et paramètres ?
- Quelles extensions possibles ? Génération pondérée (Boltzmann multidim.)
- Est ce bien nécessaire ? **Probablement** car comptage #P-difficile

Etat de l'art

Approche de Abfalter/Flamm/Stadler 2003 :

- Décomposition *en oreille* du graphe **[Whitney 1932]** *Epluchage du graphe* ;



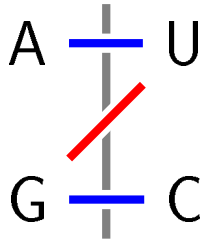
- **Programmation dynamique :** comptage des #chemins valides pour chaque composante, étant donné des valeurs pour les **ancrages** (noeuds noirs) ;
- **Recombiner** les valeurs pour calculer le nombre de séquences valides.

Complexité : $\Theta(n.4^A)$ où $A =$ #Max. d'ancrages, $A \in \Theta(n)$ dans le cas au pire.

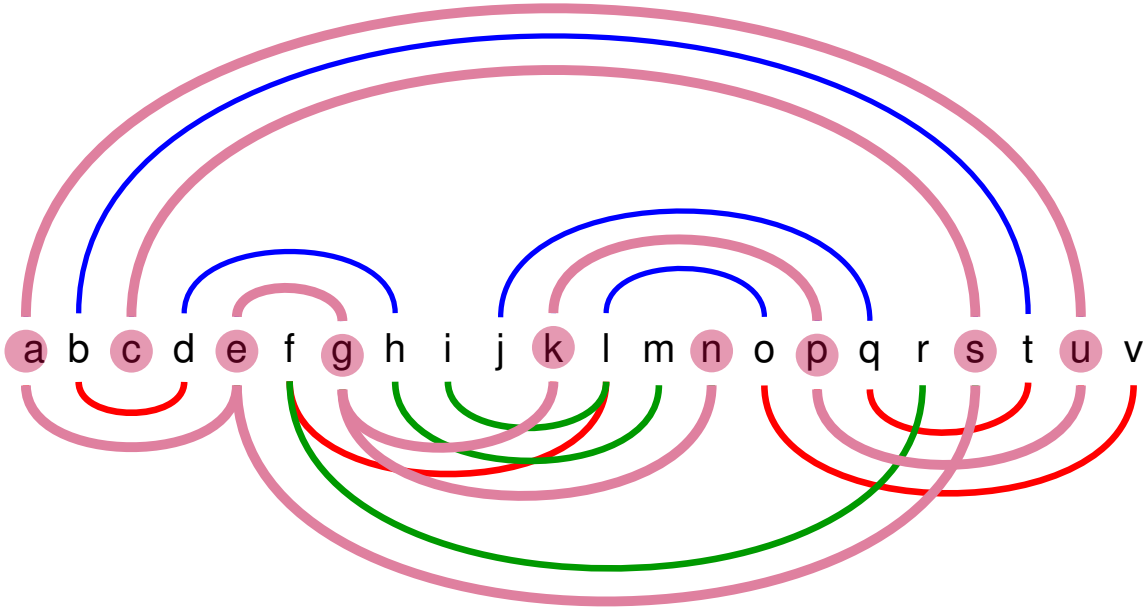
Quelques réactions :

- Est ce bien *optimal* ? Autres algorithmes et paramètres ?
- Quelles extensions possibles ? Génération pondérée (Boltzmann multidim.)
- Est ce bien nécessaire ? **Probablement** car comptage #P-difficile

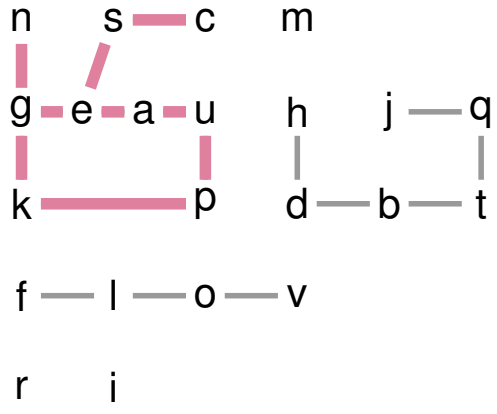
Compter les séquences valides : Watson-Crick + > 2 structures



Paires de Bases valides = autorise les paires **Wobble**



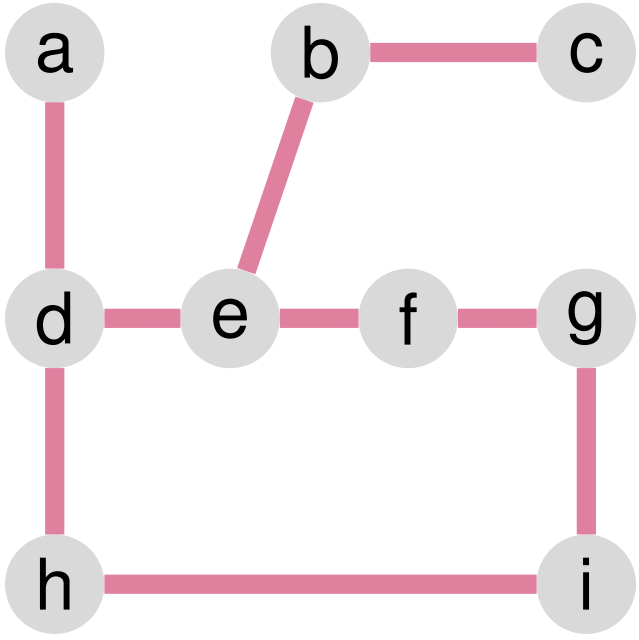
Graphe de dépendance :
Cycles, Chemins, Arbres...

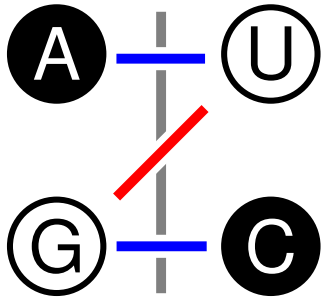


Question : Combien de séquences **valides** ?

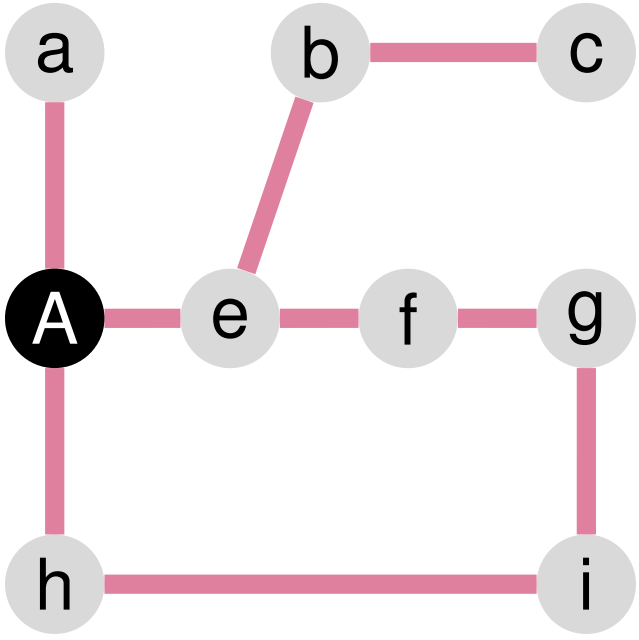
Réponse : Non-biparti $\rightarrow \varnothing$; Biparti $\rightarrow$? ? ? ?

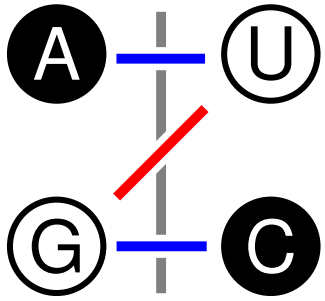
Bijection entre stables d'un graphe biparti et séquences valides



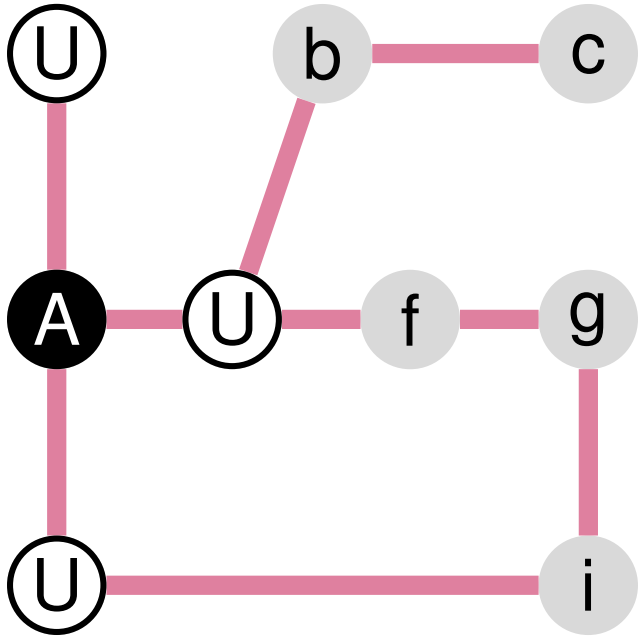


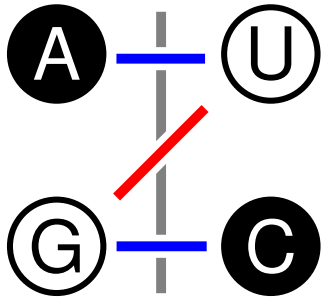
Bijection entre stables d'un graphe biparti et séquences valides



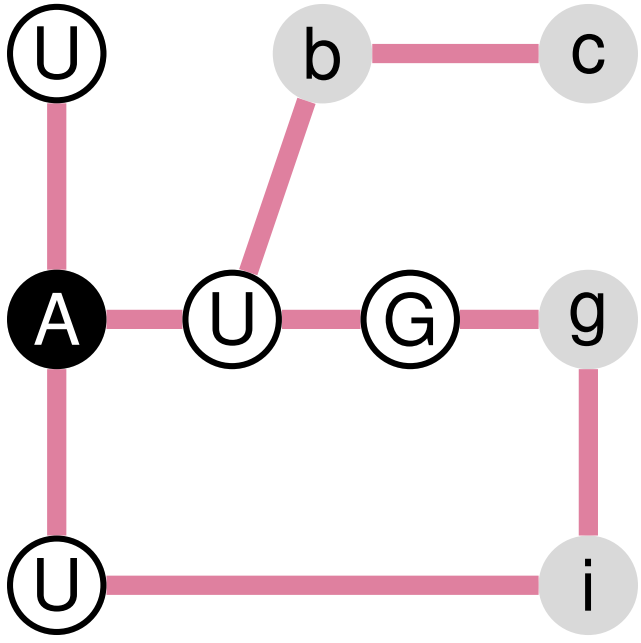


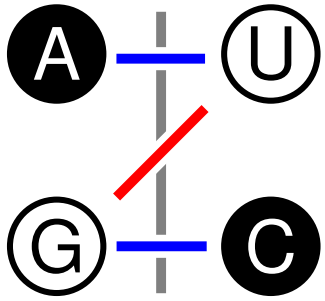
Bijection entre stables d'un graphe biparti et séquences valides



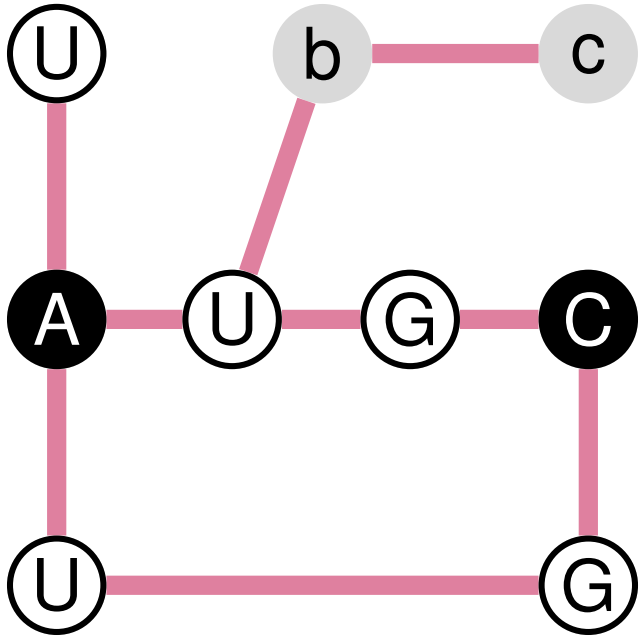


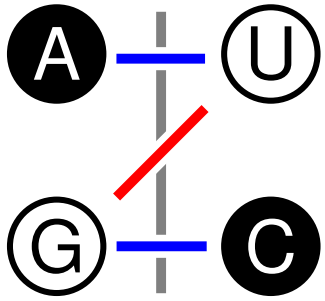
Bijection entre stables d'un graphe biparti et séquences valides



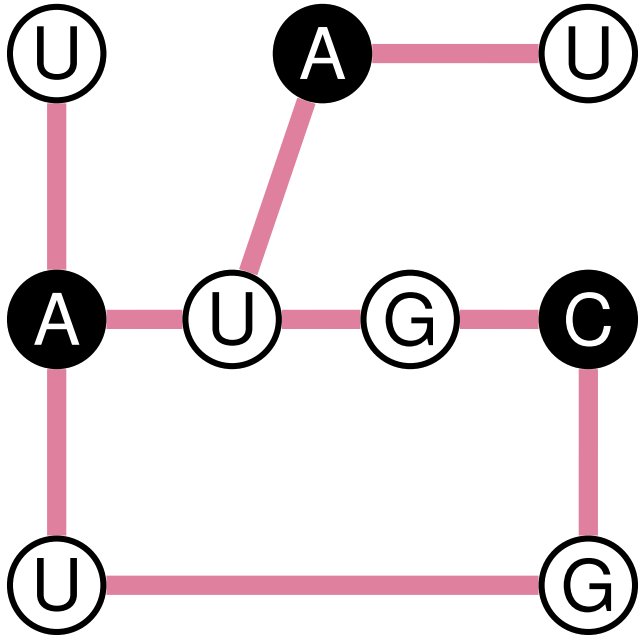


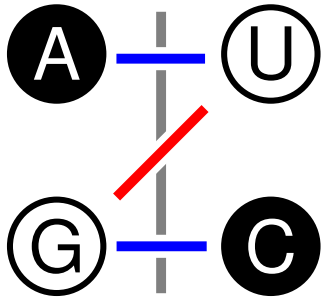
Bijection entre stables d'un graphe biparti et séquences valides



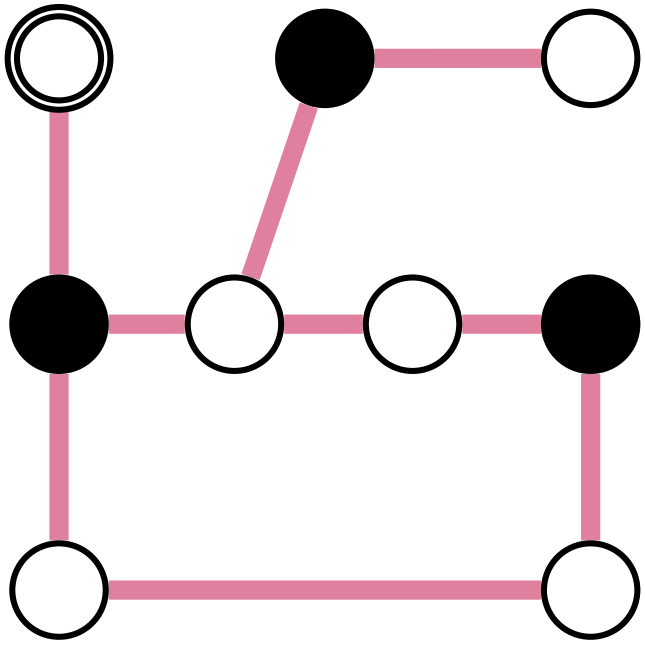


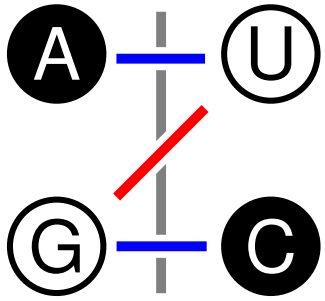
Bijection entre stables d'un graphe biparti et séquences valides





Bijection entre stables d'un graphe biparti et séquences valides



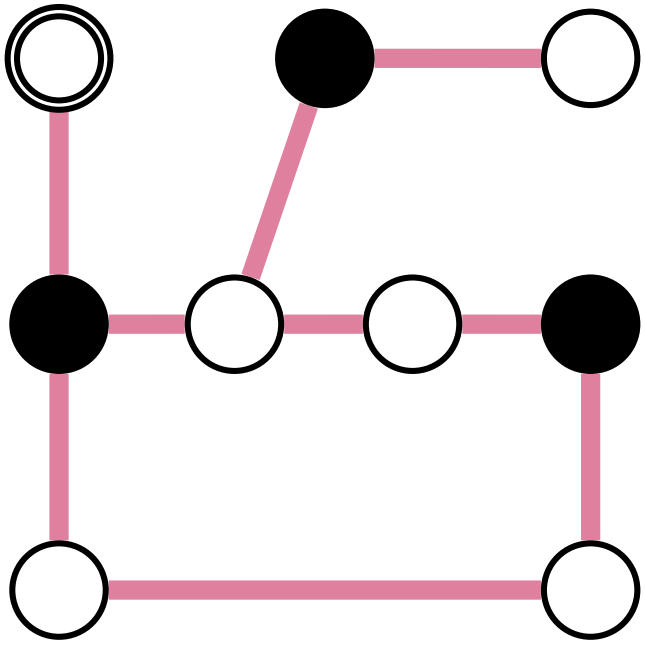


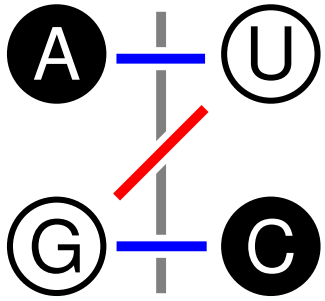
Remarque : Lettres noires non-adjacentes dans séquences valides

À symétrie triviale près* (par ex. position $\nwarrow \in \{\mathrm{U}, \mathrm{C}\}$) :

$$\mathrm{Designs}^\star(\mathrm{cc}) \subseteq \mathrm{Stables}(\mathrm{cc})$$

Bijection entre stables d'un graphe biparti et séquences valides





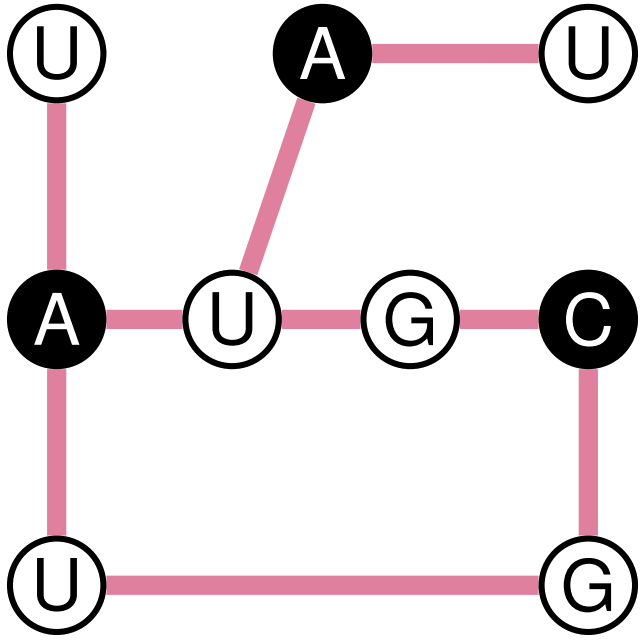
Remarque : Lettres noires non-adjacentes dans séquences valides

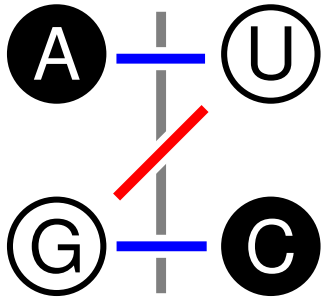
À symétrie triviale près* (par ex. position $\nwarrow \in \{\mathrm{U}, \mathrm{C}\}$) :

$$\mathrm{Designs}^\star(\mathrm{cc}) \subseteq \mathrm{Stables}(\mathrm{cc})$$

Par ailleurs, Stable (en noir) + $\nwarrow$ vert. $\in \{\mathrm{U}, \mathrm{C}\} \Rightarrow$ Séquence valide

Bijection entre stables d'un graphe biparti et séquences valides





Remarque : Lettres noires non-adjacentes dans séquences valides

À symétrie triviale près* (par ex. position $\nwarrow \in \{\mathrm{U}, \mathrm{C}\}$) :

$$\mathrm{Designs}^\star(\mathrm{cc}) \subseteq \mathrm{Stables}(\mathrm{cc})$$

Par ailleurs, Stable (en noir) + $\nwarrow$ vert. $\in \{\mathrm{U}, \mathrm{C}\} \Rightarrow$ Séquence valide

$$\Rightarrow \text{Bijection entre } \mathrm{Designs}^\star(\mathrm{cc}) \text{ et } \mathrm{Stables}(\mathrm{cc}).$$

Séquences valides et stables d'un graphe biparti

Théorème (#Séquences valides et stables dans les graphes connexes bipartis)

Soit G un graphe de dépendance ***biparti et connexe*** *:*

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^{\star}(G) = 2 \times \#\text{Stables}(G)$$

Pour un graphe de dépendance **biparti** G, on a :

$$\#Designs(G) = \prod_{cc \in CC(G)} 2 \times \#Stables(cc) = 2^{|CC(G)|} \times \#Stables(G)$$

Mais $\#Stables(G)$ est **#P-difficile** sur les graphes bipartis ($\#BIS$) **[Bubbley&Dyer'01]**

(+ Tout G est graphe de dépendance de quelque famille de structures)

Donc $\exists$ algo. $\mathcal{A} \in P$ pour $\#Designs(G) \Rightarrow \exists$ algo. $\mathcal{A}' \in P$ pour $\#BIS \ldots$

Théorème

#Designs est #P-hard.

Pas d'algorithme polynomial pour #Designs(G) **sauf si** $\#P = FP$ ($\Rightarrow P = NP$)

Séquences valides et stables d'un graphe biparti

Théorème (#Séquences valides et stables dans les graphes connexes bipartis)

Soit G un graphe de dépendance ***biparti et connexe*** *:*

$$\#\text{Designs}(G) = 2 \times \#\text{Designs}^{\star}(G) = 2 \times \#\text{Stables}(G)$$

Pour un graphe de dépendance **biparti** G, on a :

$$\#Designs(G) = \prod_{cc \in CC(G)} 2 \times \#Stables(cc) = 2^{|CC(G)|} \times \#Stables(G)$$

Mais $\#Stables(G)$ est **#P-difficile** sur les graphes bipartis ($\#BIS$) **[Bubbley&Dyer'01]**

(+ Tout G est graphe de dépendance de quelque famille de structures)

Donc $\exists$ algo. $\mathcal{A} \in P$ pour $\#Designs(G) \Rightarrow \exists$ algo. $\mathcal{A}' \in P$ pour $\#BIS \ldots$

Théorème

#Designs est #P-hard.

Pas d'algorithme polynomial pour #Designs(G) **sauf si** $\#P = FP$ ($\Rightarrow P = NP$)

Séquences valides et stables d'un graphe biparti

Théorème (#Séquences valides et stables dans les graphes connexes bipartis)

Soit G un graphe de dépendance ***biparti et connexe*** *:*

$$\#\text{Designs}(\text{G}) = 2 \times \#\text{Designs}^{\star}(\text{G}) = 2 \times \#\text{Stables}(\text{G})$$

Pour un graphe de dépendance **biparti** G, on a :

$$\#Designs(G) = \prod_{cc \in CC(G)} 2 \times \#Stables(cc) = 2^{|CC(G)|} \times \#Stables(G)$$

Mais $\#Stables(G)$ est **$\#$P-difficile** sur les graphes bipartis ($\#BIS$) **[Bubbley&Dyer'01]**

(+ Tout G est graphe de dépendance de quelque famille de structures)

Donc $\exists$ algo. $\mathcal{A} \in P$ pour $\#Designs(G) \Rightarrow \exists$ algo. $\mathcal{A}' \in P$ pour $\#BIS$...

Théorème

#Designs est #P-hard.

Pas d'algorithme polynomial pour #Designs(G) **sauf si** $\#P = FP$ ($\Rightarrow P = NP$)

Séquences valides et stables d'un graphe biparti

Théorème (#Séquences valides et stables dans les graphes connexes bipartis)

Soit G un graphe de dépendance ***biparti et connexe*** *:*

$$\#\text{Designs}(\text{G}) = 2 \times \#\text{Designs}^{\star}(\text{G}) = 2 \times \#\text{Stables}(\text{G})$$

Pour un graphe de dépendance **biparti** G, on a :

$$\#Designs(G) = \prod_{cc \in CC(G)} 2 \times \#Stables(cc) = 2^{|CC(G)|} \times \#Stables(G)$$

Mais $\#Stables(G)$ est **$\#$P-difficile** sur les graphes bipartis ($\#BIS$) **[Bubbley&Dyer'01]**

(+ Tout G est graphe de dépendance de quelque famille de structures)

Donc $\exists$ algo. $\mathcal{A} \in P$ pour $\#Designs(G) \Rightarrow \exists$ algo. $\mathcal{A}' \in P$ pour $\#BIS\ldots$

Théorème

#Designs est #P-hard.

Pas d'algorithme polynomial pour #Designs(G) **sauf si** $\#P = FP$ ($\Rightarrow P = NP$)

Conséquences

Corollaire (#Approximabilité pour ≤ 5 structures) [Weitz'06]

Pour tout graphe G construit à partir de ≤ 5 structures (avec croisements), #Design(G) peut être approché dans **n'importe quel ratio** en **temps polynomial** (PTAS)

Corollaire ? ? ? (#BIS-difficulté pour > 5 structures) [Cai, Galanis *et al* 16]

À partir de 5 structures (avec croisements), approcher #Design est aussi **aussi difficile que** d'approcher #BIS sans contrainte.

Pourquoi des croisements ? Parce que tout graphe biparti de degré max Δ peut être **décomposé** en Δ couplages en **temps polynomial** (Théorème de Vizing).

Enfin, forte connexion entre **comptage** et **échantillonnage** [Jerrum/Valiant/Vazirani'86].

Conjecture (#BIS-difficulté de l'échantillonnage)

Génération quasi-uniforme aussi difficile qu'approcher #BIS en général

$\Rightarrow$ **Génération aléatoire uniforme** #P difficile ?

Conséquences

Corollaire (#Approximabilité pour ≤ 5 structures) [Weitz'06]

Pour tout graphe G construit à partir de ≤ 5 structures (avec croisements), #Design(G) peut être approché dans **n'importe quel ratio** en **temps polynomial** (PTAS)

Corollaire ? ? ? (#BIS-difficulté pour > 5 structures) [Cai, Galanis *et al*'16]

À partir de 5 structures (avec croisements), approcher #Design est aussi **aussi difficile que** d'approcher #BIS sans contrainte.

Pourquoi des croisements ? Parce que tout graphe biparti de degré max Δ peut être **décomposé** en Δ couplages en **temps polynomial** (Théorème de Vizing).

Enfin, forte connexion entre **comptage** et **échantillonnage** [Jerrum/Valiant/Vazirani'86].

Conjecture (#BIS-difficulté de l'échantillonnage)

Génération quasi-uniforme aussi difficile qu'approcher #BIS en général

$\Rightarrow$ **Génération aléatoire uniforme** #P difficile ?

Conséquences

Corollaire (#Approximabilité pour ≤ 5 structures) [Weitz'06]

Pour tout graphe G construit à partir de ≤ 5 structures (avec croisements), #Design(G) peut être approché dans **n'importe quel ratio** en **temps polynomial** (PTAS)

Corollaire ? ? ? (#BIS-difficulté pour > 5 structures) [Cai, Galanis *et al*'16]

À partir de 5 structures (avec croisements), approcher #Design est aussi **aussi difficile que** d'approcher #BIS sans contrainte.

Pourquoi des croisements ? Parce que tout graphe biparti de degré max Δ peut être **décomposé** en Δ couplages en **temps polynomial** (Théorème de Vizing).

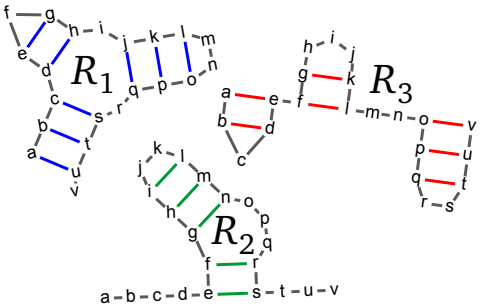
Enfin, forte connexion entre **comptage** et **échantillonnage** [Jerrum/Valiant/Vazirani'86].

Conjecture (#BIS-difficulté de l'échantillonnage)

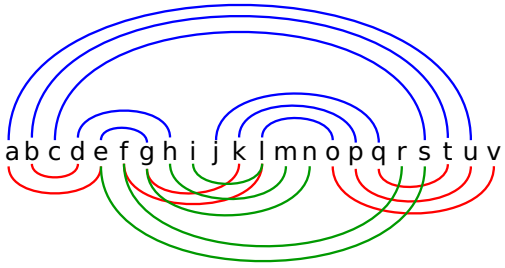
Génération quasi-uniforme aussi difficile qu'approcher #BIS en général

$\Rightarrow$ **Génération aléatoire uniforme** #P difficile ?

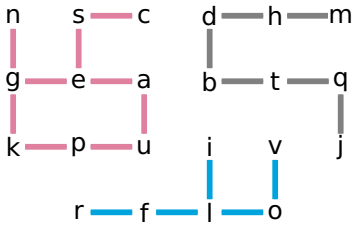
Décomposition arborescente et génération de Boltzmann



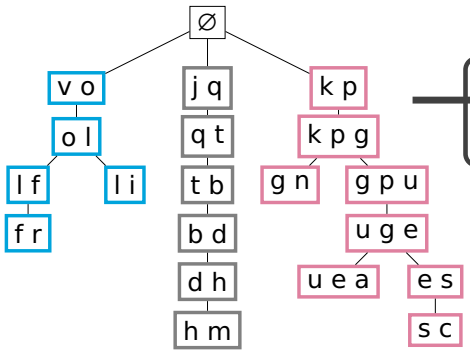
i) Input Structures



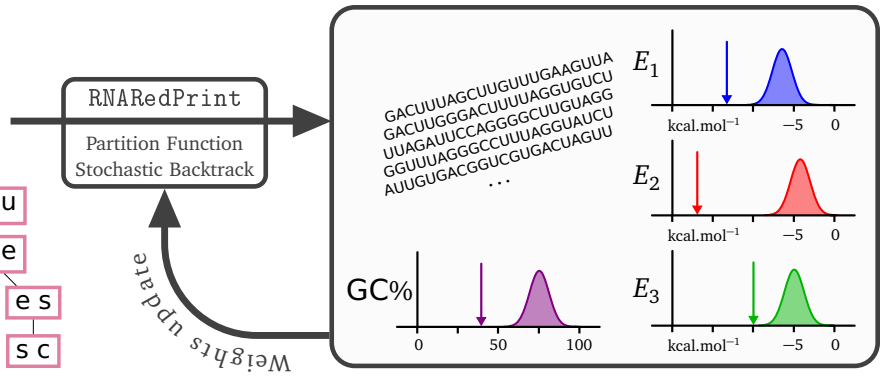
ii) Merged Base-Pairs



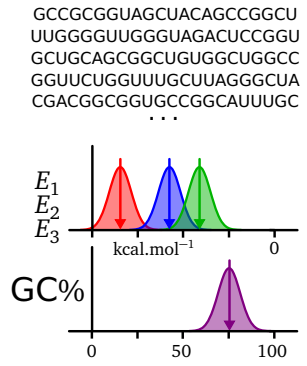
iii) Compatibility Graph



iv) Tree Decomposition



v) Weight Optimization (Adaptive Sampling)



vi) Final Designs

[Hammer/P/Wang/Will, RECOMB'18]

- **Algorithme** de complexité paramétrée sur la **largeur arborescente**
- Génération de Boltzmann multidimensionelle pour contrôler l'énergie, GC%. . .

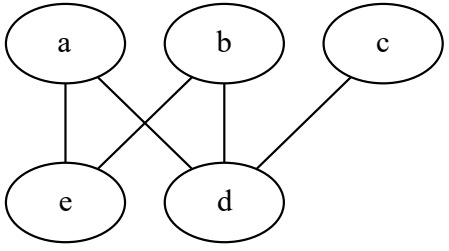
Décomposition arborescente et comptage/génération

Décomposition arborescente T pour graphe $G = (V, E)$ =

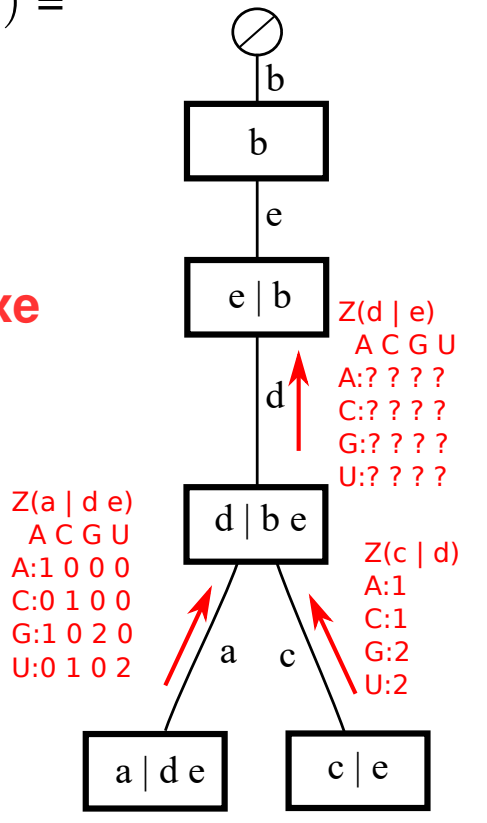
1. Noeuds B tels que $\forall b \in B,\ b \subseteq V$
2. $\forall v \in V, \exists b \in B$ tel que $v \in b$
3. $\forall (v, v') \in E, \exists b \in B$ tel que $\{v, v'\} \subseteq B$
4. Arètes telles que sous arbre induit par $v \in V$ **connexe**

a	b	c	d	e
(	.	.	)	.
.	(	(	)	)
(	(	.	)	)

Structures cibles



Graphe de dépendance



Décomp. arborescente

$b = \{b_1, b_2 \ldots\}$: noeud de D
T_b : sous-arbre enraciné en b
w : **Largeur** de la décomposition D

$$\mathcal{Z}(T_b \mid b_2 \leftarrow v_2 \ldots) = \sum_{\substack{b_1 \leftarrow v_1 \\ v_1 \in \{A,C,G,U\}}} \prod_{c \text{ fils de } b} \mathcal{Z}(T_c \mid b_1 \leftarrow v_1, b_2 \leftarrow v_2 \ldots)$$

Complexité $\Theta(n\,m\,k + n\,k\,2^w)$ pour **génération uniforme** de m séq. (k struct.)

Décomposition arborescente et comptage/génération

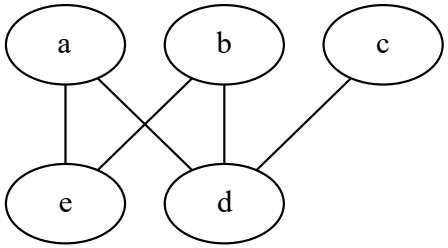
Décomposition arborescente T pour graphe $G = (V, E)$ =

1. Noeuds B tels que $\forall b \in B,\ b \subseteq V$
2. $\forall v \in V, \exists b \in B$ tel que $v \in b$
3. $\forall (v, v') \in E, \exists b \in B$ tel que $\{v, v'\} \subseteq B$
4. Arètes telles que sous arbre induit par $v \in V$ **connexe**

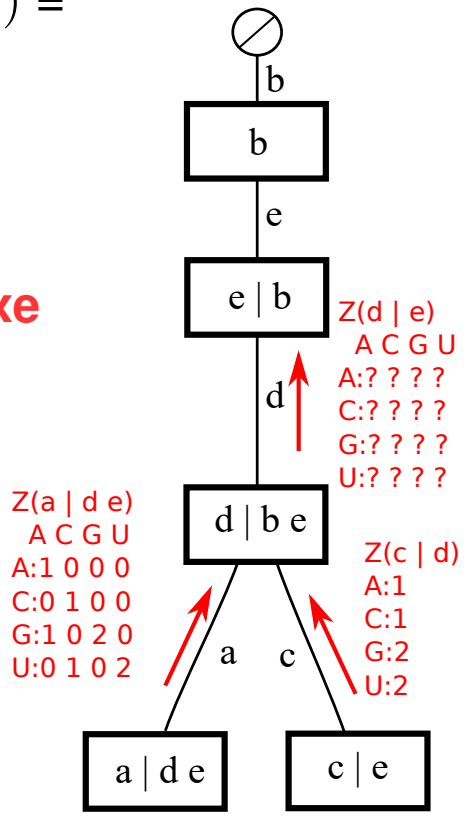
```
a   b   c   d   e

(   .   .   )   .
.   (   (   )   )
(   (   .   )   )
```

Structures cibles



Graphe de dépendance



Décomp. arborescente

$b = \{b_1, b_2 \ldots\}$: noeud de D
T_b : sous-arbre enraciné en b
w : **Largeur** de la décomposition D

$$\mathcal{Z}(T_b \mid b_2 \leftarrow v_2 \ldots) = \sum_{\substack{b_1 \leftarrow v_1 \\ v_1 \in \{A,C,G,U\}}} \prod_{c \text{ fils de } b} \mathcal{Z}(T_c \mid b_1 \leftarrow v_1, b_2 \leftarrow v_2 \ldots)$$

Complexité $\Theta\left(n\,m\,k + n\,k\,2^{w+\#CC}\right)$ pour **génération uniforme** de m séq. (k struct.)

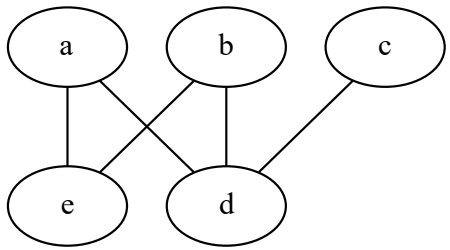
Décomposition arborescente et comptage/génération

Décomposition arborescente T pour graphe $G = (V, E)$ =

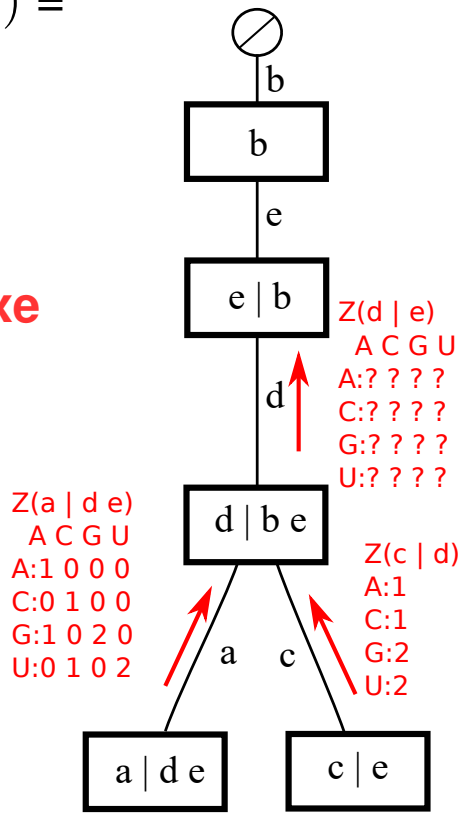
1. Noeuds B tels que $\forall b \in B,\ b \subseteq V$
2. $\forall v \in V, \exists b \in B$ tel que $v \in b$
3. $\forall (v, v') \in E, \exists b \in B$ tel que $\{v, v'\} \subseteq B$
4. Arètes telles que sous arbre induit par $v \in V$ **connexe**

```
a   b   c   d   e
(   .   .   )   .
.   (   (   )   )
(   (   .   )   )
```

Structures cibles



Graphe de dépendance



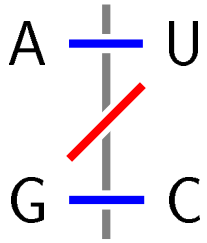
Décomp. arborescente

$b = \{b_1, b_2 \ldots\}$: noeud de D
T_b : sous-arbre enraciné en b
w : **Largeur** de la décomposition D

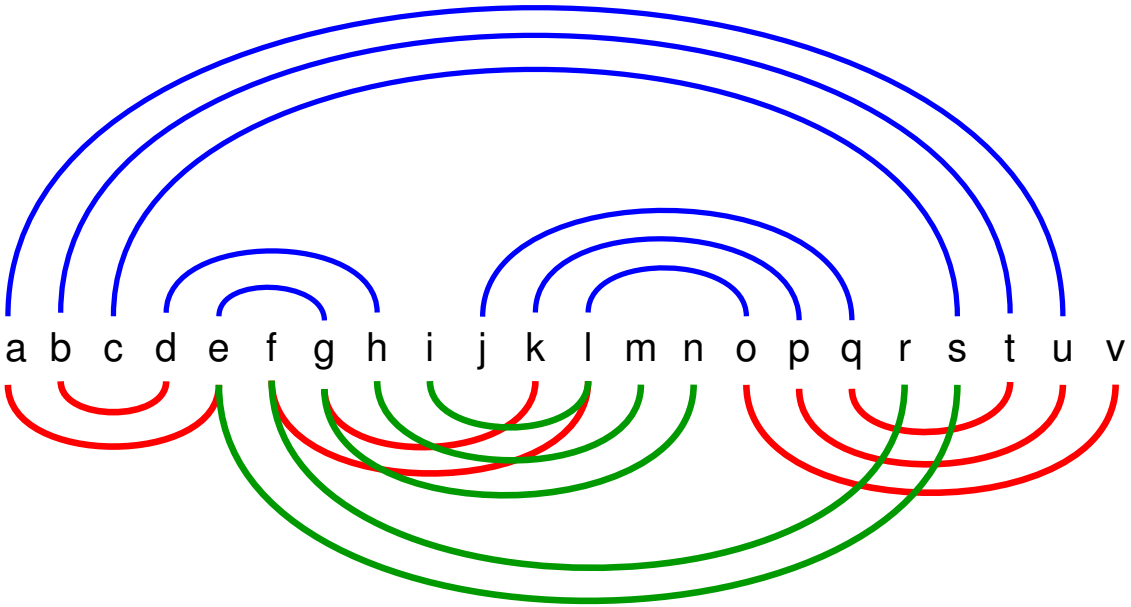
$$\mathcal{Z}(T_b \mid b_2 \leftarrow v_2 \ldots) = \sum_{\substack{b_1 \leftarrow v_1 \\ v_1 \in \{A,C,G,U\}}} \prod_{i=1}^{k} x_i^{E(b, v_1, \ldots)} \prod_{c \text{ fils de } b} \mathcal{Z}(T_c \mid b_1 \leftarrow v_1, b_2 \leftarrow v_2 \ldots)$$

Complexité $\Theta(n\,m\,k + n\,k\,2^w)$ pour **génération pondérée** de m séq. (k struct.)

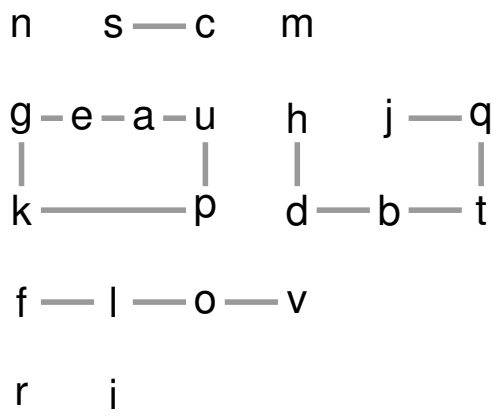
Compter les séquence valides : Watson-Crick + > 2 structures



Paires de Bases valides = autorise les paires **Wobble**



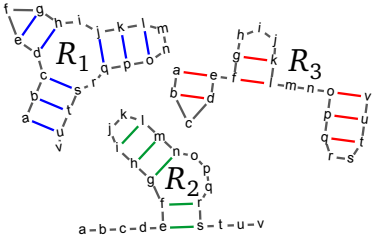
Graphe de dépendance :
Cycles, Chemins, Arbres...



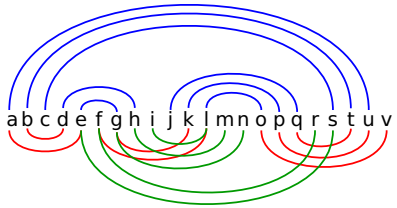
Question : Combien de séquences **valides** ?

Réponse : Biparti $\rightarrow$ $\prod_{cc \in CC(G)} 2 \times \#Stables(cc) = 496\,672$

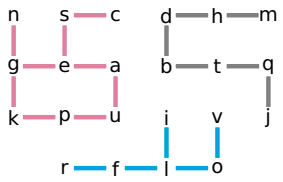
Décomposition arborescente et génération de Boltzmann



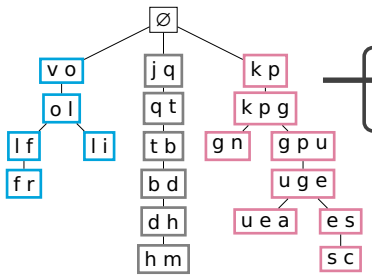
i) Input Structures

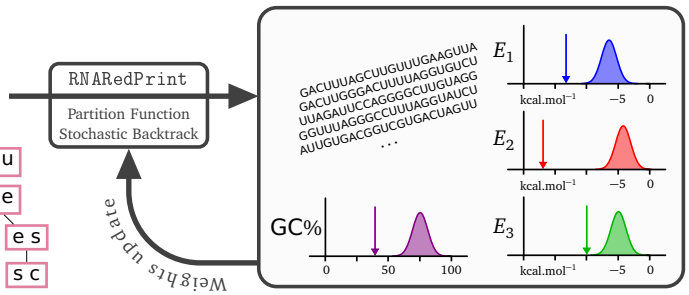


ii) Merged Base-Pairs



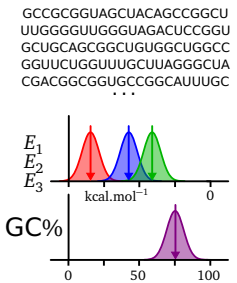
iii) Compatibility Graph





iv) Tree Decomposition

v) Weight Optimization (Adaptive Sampling)



vi) Final Designs

[Hammer/P/Wang/Will, RECOMB'18]

- **Algorithme** de complexité paramétrée sur la **largeur arborescente**
- **Génération de Boltzmann multidim.** **[Bodini/P. 2011]** pour contrôler l'énergie ...

Entrée : Energie visées $(E_\ell^\star)_{\ell=1}^k$, poids $(x_\ell)_{\ell=1}^k$ tels que $\mathbb{E}(E(w, S_\ell)) = E_\ell^\star, \forall \ell$:

$$\mathbb{P}(w \mid x_1 \cdots x_k) \sim \prod_{\ell=1}^{k} x_\ell^{E(w,S_\ell)}$$ **+ Rejet efficace**

A faire : Itération numérique pour déterminer $(x_\ell)_{\ell=1}^k$ (cf exposé Sergey)

Merci !