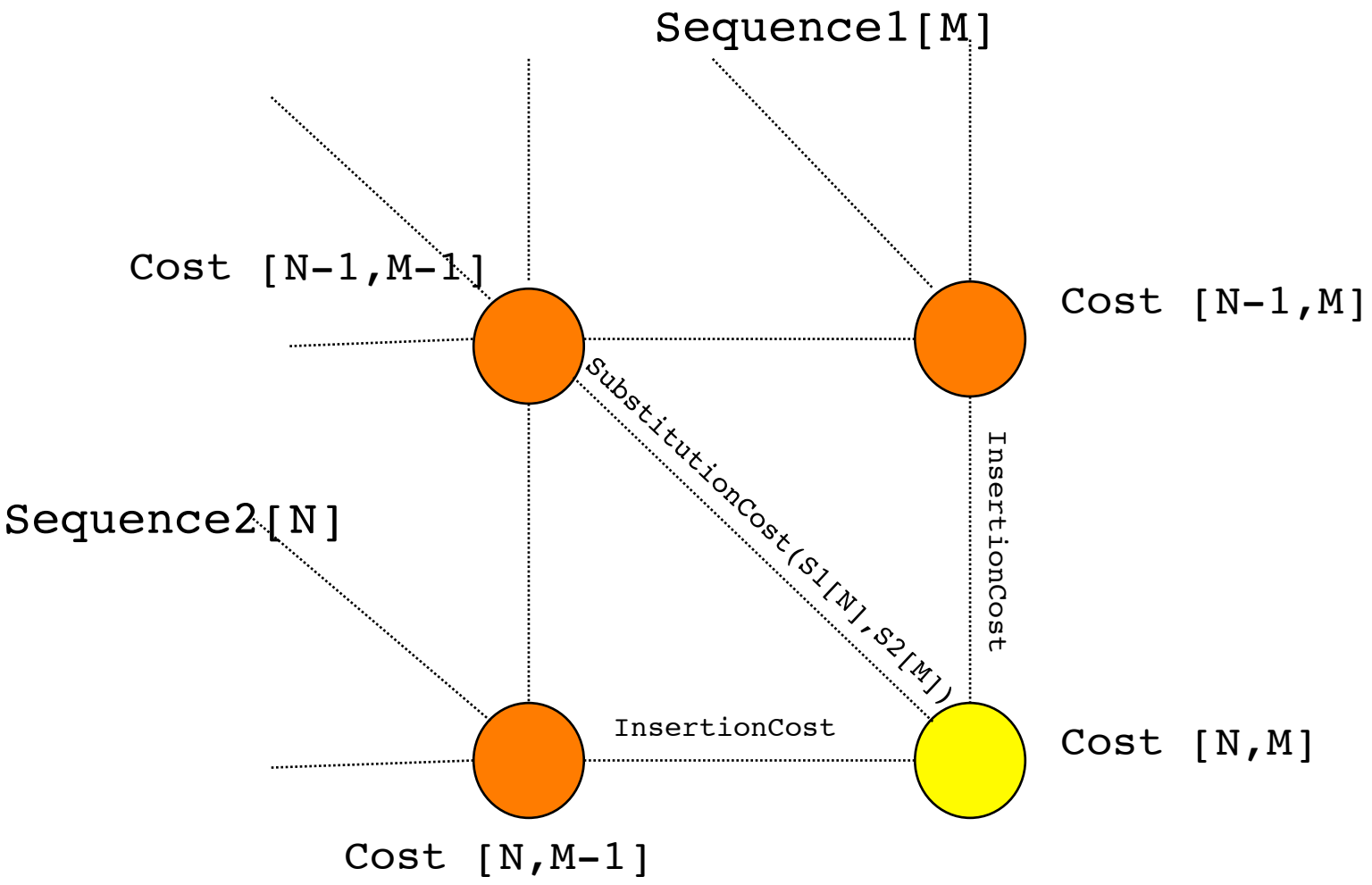
4. Comparaison de séquences

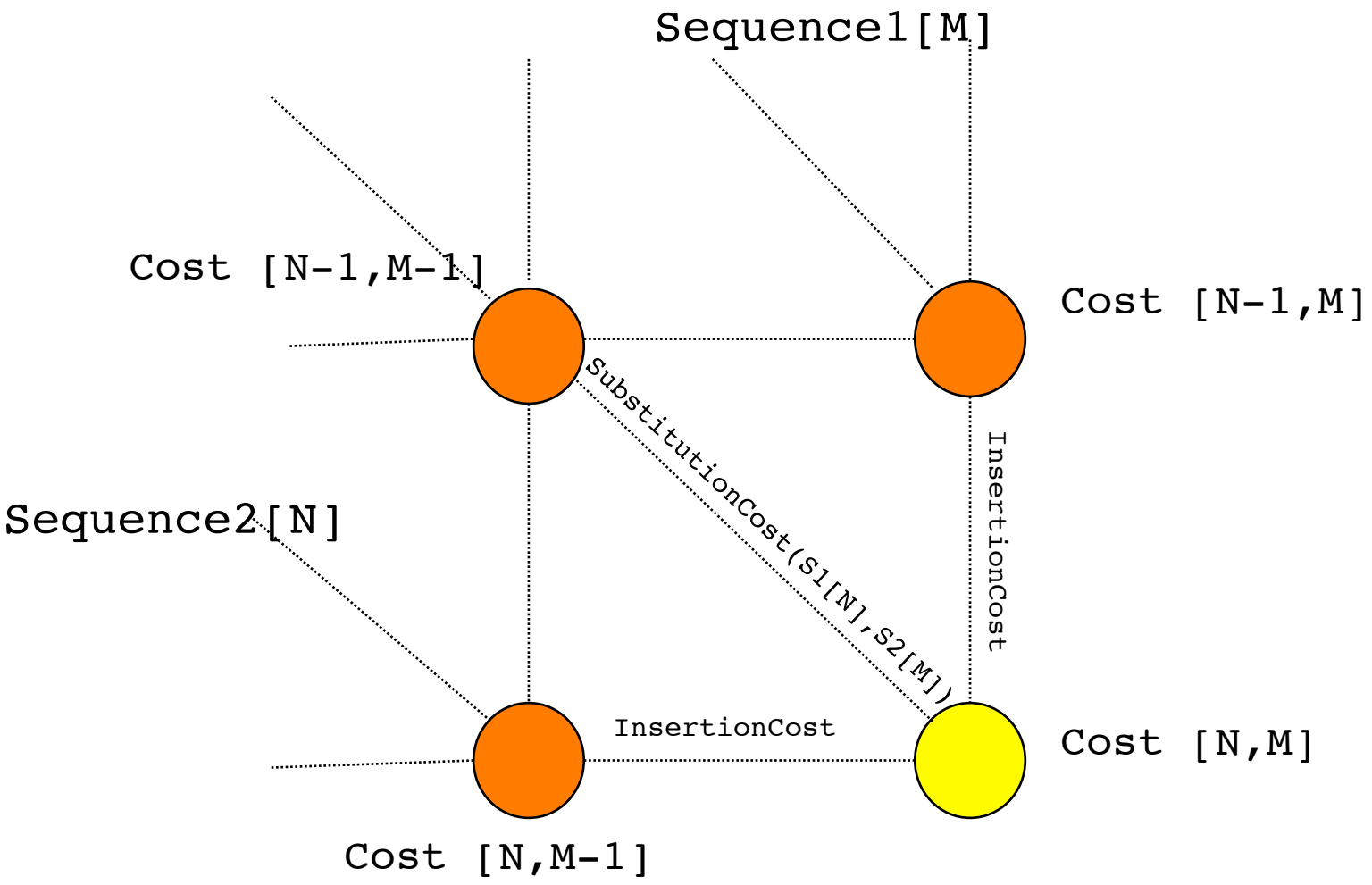
- Comment prédire les fonctions des gènes/protéines ?
- Évolution et similarité de séquences
- Quantifier la similarité de deux séquences
- L'alignement de séquences devient un problème d'optimisation
- Un alignement de séquences vu comme un chemin dans une grille
- Si un chemin est optimal, tous ses chemins partiels sont optimaux
- Coûts et alignement
- **Un algorithme récursif**
- Eviter la récursivité : une version itérative
- Cet algorithme est-il efficace ?


Sequence1[M]
Cost [N-1,M-1]
Cost [N-1,M]
SubstitutionCost(S1[N],S2[M])
InsertionCost
Sequence2[N]
InsertionCost
Cost [N,M]
Cost [N,M-1]

Fonctions récursives

- Une **fonction récursive fait appel à elle-même** au cours même de son exécution

- Deux points à considérer lors de l'écriture d'une telle fonction
  1. Une **condition d'arrêt** doit être testée dès le début de la fonction
  2. Les appels internes au sein de la fonction doivent s'**appliquer sur un problème plus « petit »** que le problème traité par l'exécution courante


Sequence1[M]
Cost [N-1,M-1]
Cost [N-1,M]
SubstitutionCost(S1[N],S2[M])
InsertionCost
Sequence2[N]
InsertionCost
Cost [N,M]
Cost [N,M-1]

Calcul récursif des coûts

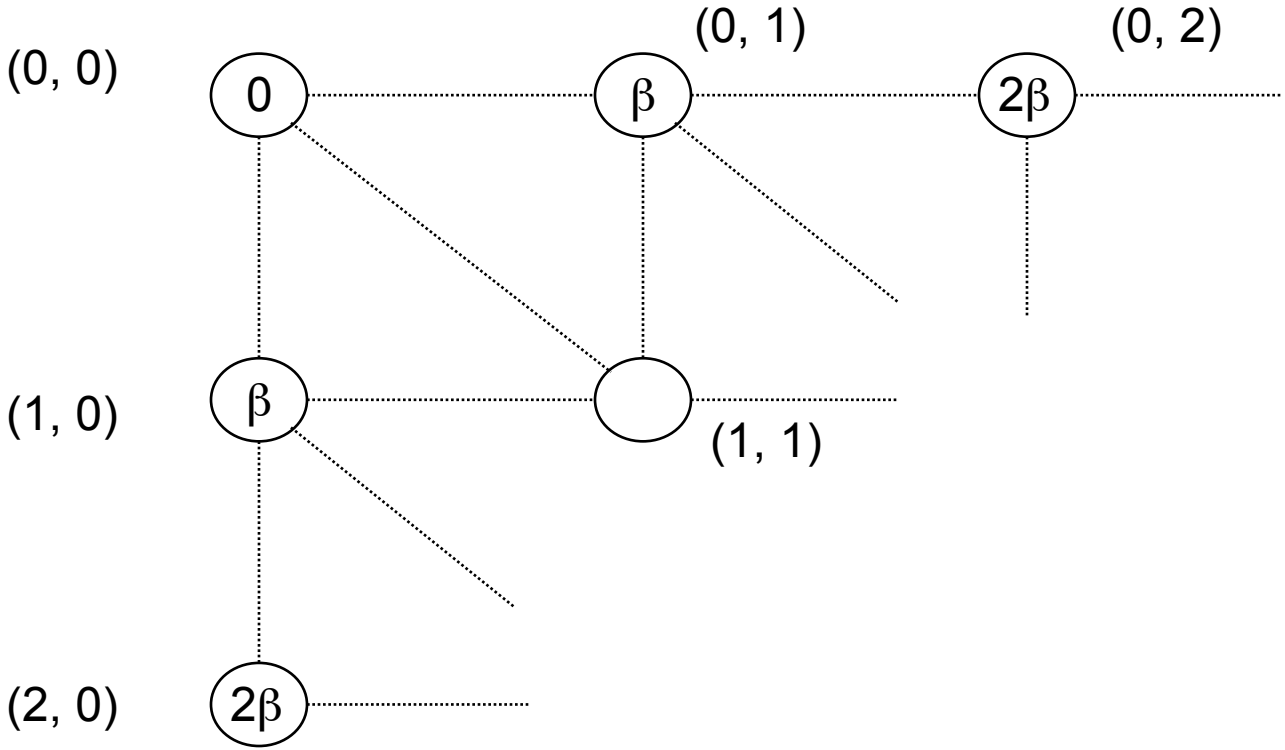
```
function ComputeCost (I, J: integer) returns integer
```

```
ComputeCost(I-1, J-1) +
  substitutionCost(Sequence1[I], Sequence2[J])
ComputeCost(I, J-1) + InsertionCost
ComputeCost(I-1, J) + InsertionCost
```

Calcul récursif des coûts

```
function ComputeCost (I, J: integer) returns integer
 if I = 0 and J =0 then return 0
   else
      return Min (
         ComputeCost(I-1, J-1) +
           substitutionCost(Sequence1[I], Sequence2[J])
         ComputeCost(I, J-1) + InsertionCost
         ComputeCost(I-1, J) + InsertionCost
         )
```

β: `InsertionCost`



Calcul récursif des coûts

```
function ComputeCost (I, J: integer) returns integer
 if I = 0 and J =0 then return 0
   else if I = 0 and J ≥ 1 then return J*InsertionCost
   else if I ≥ 1 and J = 0 then return I*InsertionCost
   else
      return Min (
         ComputeCost(I-1, J-1) +
           substitutionCost(Sequence1[I], Sequence2[J])
         ComputeCost(I, J-1) + InsertionCost
         ComputeCost(I-1, J) + InsertionCost
         )
```