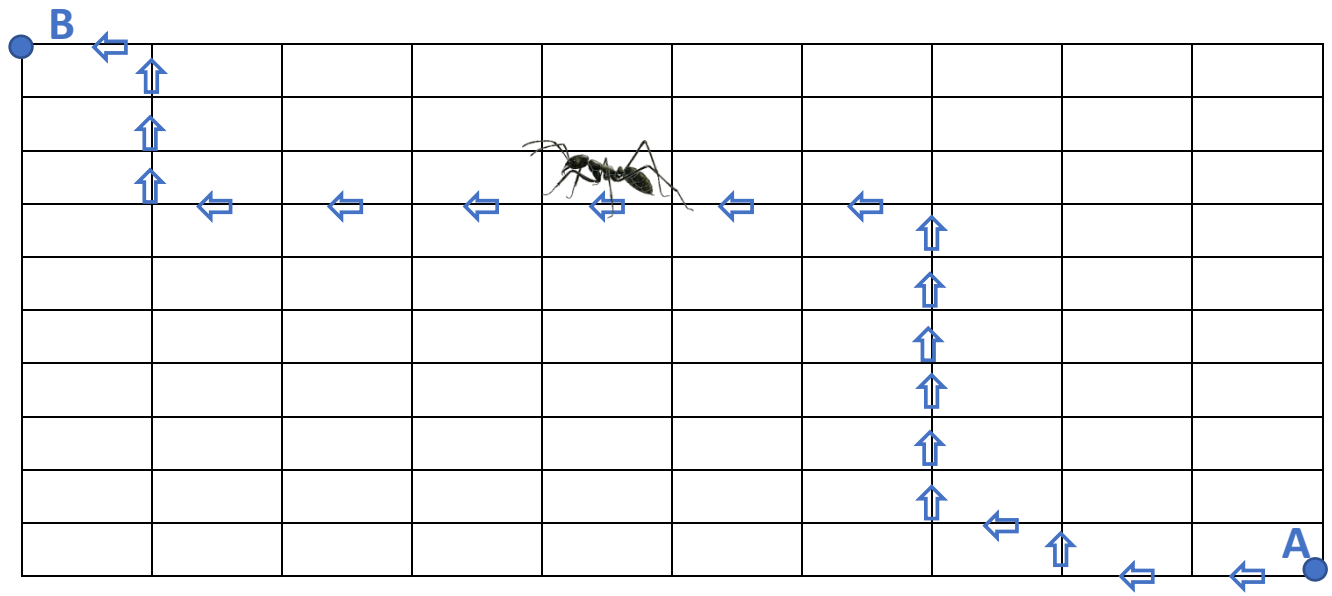
Chapitre 21 - Programmation dynamique

En informatique, la programmation dynamique est une méthode algorithmique qui permet de résoudre des problèmes d'optimisation efficacement.
Pour optimiser des problèmes de grande taille, pour lesquels une solution naïve serait trop gourmande en ressources, il est nécessaire d'écrire son code en changeant de paradigme. Parmi les paradigmes algorithmiques déjà vus, on peut citer l'algorithme glouton, les k plus proches voisins, diviser pour régner. On en voit ici un nouveau : la programmation dynamique. D'une manière générale, la méthode consiste « ***à diviser un problème complexe en sous-problèmes dont la taille varie durant l'exécution. Les résultats de ces sous-problèmes sont mémorisés afin de pouvoir être réutilisés ensuite.*** »

On voit dans ce chapitre plusieurs exemples concrets. On fait une synthèse de ces différentes situations en fin de chapitre.

1- PROBLEME DE LA FOURMI :

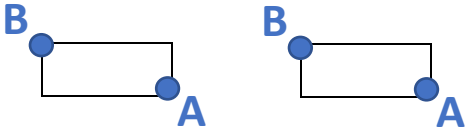
Alice pose le problème suivant à Basile. Elle dessine sur une feuille de papier une petite grille de N lignes et N colonnes. Par exemple, si $N = 10$, on obtient la figure suivante :



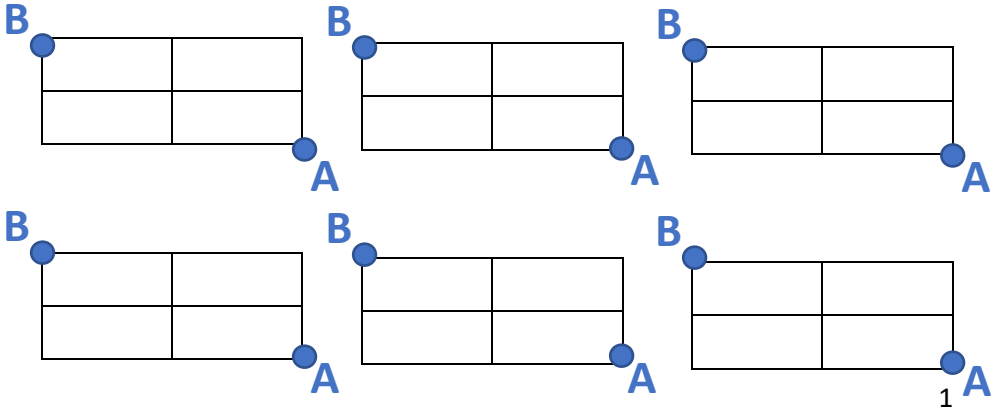
Une fourmi part du point **A** et veut aller sur le point **B**. Elle ne peut que suivre une ligne et se déplacer vers la gauche ou vers le haut, sans pouvoir revenir en arrière.

Question : **Combien de chemins différents, cette fourmi peut-elle emprunter ?**

1- Nombre de chemins pour $N = 1$:

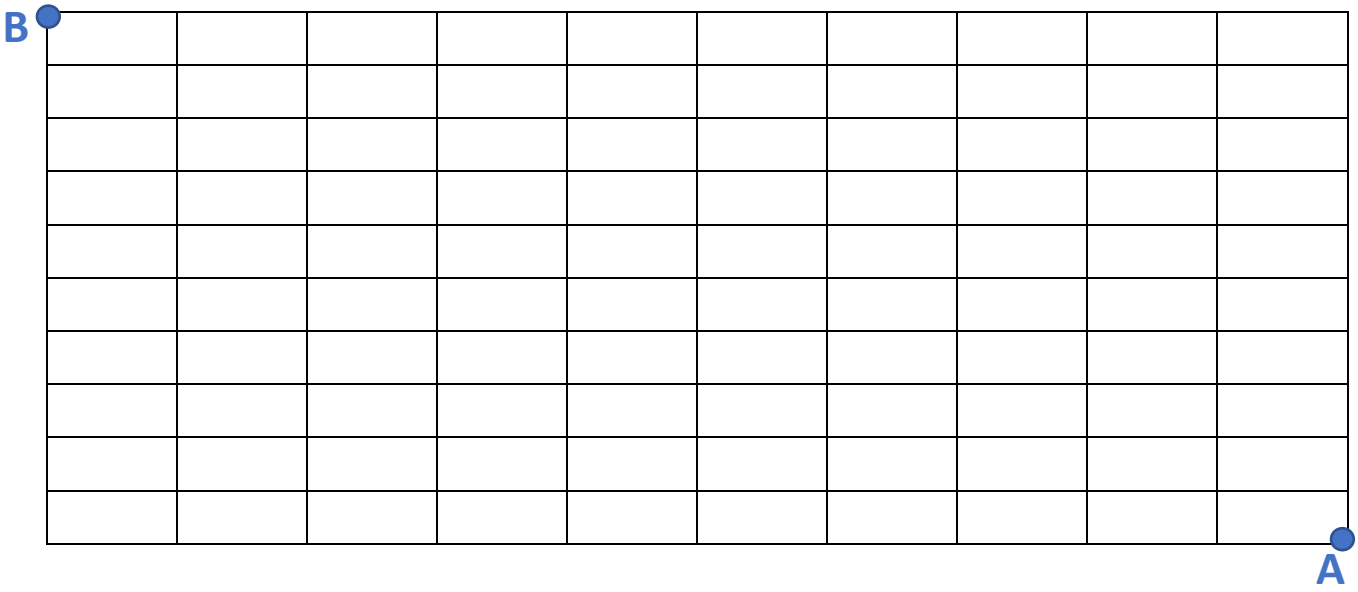


2- Nombre de chemins pour $N = 2$:



3- Nombre de chemins pour N quelconque :

On se propose ici d'écrire un algorithme qui utilise le principe de la programmation dynamique. Cela consiste à résoudre le problème en le décomposant en sous-problèmes, puis à résoudre les sous-problèmes, des plus petits aux plus grands en stockant les résultats intermédiaires.



⇨ Ecrire le code de la fonction *fourmi()* ayant pour paramètre un entier naturel N et retournant le nombre de chemins possibles. On donne 2 exemples d'exécutions :

```
>>> fourmi(1)
2
```

```
>>> fourmi(2)
6
```


```
def fourmi(N) :
```