Complexité basique:

1) L'algo énumère toutes les solutions une fois et il ne peut pas avoir une complexité plus grande que le nombre de solutions.

Soit : O(wh^n)

2) La factorielle apparaît si on considère que, quand l'annonceur 1 a été placé, on a, dans le pire case, une case de moins pour l'annonceur 2. Donc wh possibilité pour l'annonceur 1 ; wh - 1 pour l'annonceur 2, wh - 2 pour l'annonceur 3,...

Soit : wh * (wh - 1) * (wh - 2) * (...) * (wh - n) = **O((wh)! / n!)**

3) on peut dire encore mieux :

Soit : wh * (wh - w1h1) * (w2 - w1h1 - w2h2) * (...) * (wh - sum wihi)
(mais c'est plus difficile à formuler avec un fonction simple donc je ne vais pas élaborer plus cette méthodes)

Complexité ou on prends en compte que l'annonceur 1 peut ne pas être placé :

Pour **O(wh^n)** c'est facile, cela signifie que chaque annonceur a wh possibilités.
Donc 'il suffit de dire que l'annonceur a maintenant (wh+1) possibilités pour prendre en compte le cas ou il n'est pas placé. ça nous donne simplement **0((wh+1)^n)** .

Pour **O((wh)! / n!)** il suffit de dénombrer les cas manquants et les ajoutés à la précédente équation. On va par tâtonnement faire pour n=2 et n=3.
pour n=2 : (wh)*(wh-1) + 1*wh

à gauche de l'addition : les cas ou on place l'annonceur 1(vu précédemment)
à droite : les cas ou on ne place pas a1 ("1" car a1 a une seule possibilité, ne pas être placé puis wh car wh possibilité pour a2)

n=3 : (wh)*(wh-1)*(wh-2) + 1*wh *(wh-1)

Complexité ou on prends en compte que l'annonceur 1 et l'annonceur 2 et l'annonceur 3 peut ne pas être placés :

Pour n = 2 : (wh)*(wh-1) + 1*wh + 1*1 + wh*1

quand on place a1 et a2
quand on place pas a1 mais pas a2
quand on place ni a1 ni a2
quand on place a1 mais pas a2

Pour n = 3 : (wh)*(wh-1)*(wh-2) + 1*wh *(wh-1) + 1*1*(wh) + (wh)*1*(wh-1) + 1*1*1 + (wh)*1*1 + (wh)*(wh-1)*1 + 1*(wh)*1

(même idée)

Une formule de récurrence apparaît :

Pour n = 2, P2 = (wh)*(wh-1) + 1*wh + 1*1 + wh*1 = wh * wh + 1 * (wh + 1) = wh^2 + wh + 1

Pour n = 3,

P3 = (wh)*(wh-1)*(wh-2) + 1*wh *(wh-1)   + 1*1*(wh) + (wh)*1*(wh-1) + 1*1*1 + (wh)*1*1 + (wh)*(wh-1)*1 + 1*(wh)*1

= wh * (wh - 1) * (wh - 1) + 1 * wh * wh + wh * 1 * wh + 1 * 1 * (wh + 1)

= wh * ((wh - 1)^2 + wh) + 1 * (wh^2 + wh + 1)

= wh * (wh^2 -2wh + 1 + wh) + wh^2 + wh + 1) **= wh *(P2 - 2wh) + P2**

= wh^3 - wh^2 + wh + wh^2 + wh + 1

= wh^3 + 2wh + 1

On peut peut-être prouver que pour n annonceurs, on a un polynome **Pn = O(wh^n)**

Dans le calcul de n = 3, il y a une ligne où on retrouve plusieurs fois le polynome P2

P3 = wh * (wh^2 -2wh + 1 + wh) + wh^2 + wh + 1) **= wh *(P2 - 2wh) + P2**

Donc on peut en déduire que **: Pn+1 = wh^(n+1) - wh^n + wh + Pn**

On essaye entre P1 et P2 :
P1= wh + 1
On remplace P1 dans P2 avec la Formule de Pn+1 :
P2 = wh^2 - wh + wh + (wh + 1) = wh^2 + wh + 1

on remarque que ça fonctionne avec P1 et P2 aussi donc on va essayer de prouver cette formule.