Implémentation 3.1 - *parcours "Whatever First Search"*

Il s'agit du parcours générique d'un graphe, dans lequel le *sac* est une structure de donnée qui déterminera le mode de parcours.

- **Entrée** : un graphe $G = (S, A)$
- **Sortie** : dépendante du but du parcours

```
initialiser un sac contenant S
tant que le sac n'est pas vide :
    v = pop le premier élément du sac
    si v n'est pas marqué :
        marquer v et le traîter
        pour tout sommet w voisin de v :
            ajouter w au sac
```

Implémentation 3.2 - *parcours préfixe*

Ici, le sac du parcours "Whaterver First Search" est une pile

```
type graphe = int list array

let parcours_pre g s =
let n = Array.length g in (*nb sommets*)
let non_vus = Array.make n true in
let rec visite x voisins =
    if non_vus.(x) then
        (print_int x;
        non_vus.(x) <- false);
    match voisins with
    | [] -> () (*plus de voisins à traiter*)
    | v::q when non_vus.(v) ->
        visite v g.(v);
        visite x q
    | v::q -> visite x q
in visite s g.(s)
```

Implémentation 3.3 - *parcours postfixe*

Ici, le sac du parcours "Whaterver First Search" est une pile

```
type graphe = int list array
let parcours_post g s =
let n = Array.length g in (*nb sommets*)
let non_vus = Array.make n true in
let rec visite x voisins =
    if non_vus.(x) then
        non_vus.(x) <- false;
    match voisins with
    | [] -> print_int x (*plus de voisins à traiter*)
    | v::q when non_vus.(v) ->
        visite v g.(v);
        visite x q
    | v::q -> visite x q
in visite s g.(s)
```