

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

```
1 | initialiser un sac contenant S
2 | tant que le sac n'est pas vide :
3 |     v = pop le premier élément du sac
4 |     si v n'est pas marqué :
5 |         marquer v et le traiter
6 |         pour tout sommet w voisin de v :
7 |             ajouter w au sac
```

Implémentation - 3.2

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

```
1 | type graphe = int list array
2 |
3 | let parcours_pre g s =
4 | let n = Array.length g in (*nb sommets*)
5 | let non_vus = Array.make n true in
6 | let rec visite x voisins =
7 |     if non_vus.(x) then
8 |         (print_int x;
9 |          non_vus.(x) <- false);
10 | match voisins with
11 | [] -> () (*plus de voisins à traiter*)
12 | v::q when non_vus.(v) ->
13 |     visite v g.(v);
14 |     visite x q
15 | | v::q -> visite x q
16 | in visite s g.(s)
```

Implémentation - 3.3

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

```
1 | type graphe = int list array
2 | let parcours_post g s =
3 | let n = Array.length g in (*nb sommets*)
4 | let non_vus = Array.make n true in
5 | let rec visite x voisins =
6 |     if non_vus.(x) then
7 |         non_vus.(x) <- false;
8 |         match voisins with
9 |         | [] -> print_int x (*plus de voisins à traiter*)
10 |        | v::q when non_vus.(v) ->
11 |            visite v g.(v);
12 |            visite x q
13 |        | v::q -> visite x q
14 | in visite s g.(s)
```

Définition 3.4 - fonction heuristique admissible

Soit $G = (S, A, w)$ un graphe orienté pondéré. Dans le cadre de l'algorithme A^* appliqué à G d'un sommet s à un sommet t , une *heuristique* $h : S \rightarrow \mathbb{R}_+$ est dite *admissible* lorsque :

$$\forall v \in S, h(v) \leq d(v, t) = \inf \left\{ \sum_{i=1}^{p-1} w(s_i, s_{i+1}), (s_1, \dots, s_p) \text{ chemin de } v = s_1 \text{ à } t = s_p \right\}$$

Autrement dit lorsque h sous-estime la distance au sommet cible.

Définition 3.5 - fonction heuristique monotone

Soit $G = (S, A, w)$ un graphe orienté pondéré. Dans le cadre de l'algorithme A^* appliqué à G d'un sommet s à un sommet t , une *heuristique* $h : S \rightarrow \mathbb{R}_+$ est dite *monotone* lorsque :

$$\begin{cases} \forall (u, v) \in A, h(u) \leq w(u, v) + h(v) \\ h(t) = 0 \end{cases}$$

Par récurrence, on a également h admissible.