```
type Zbar=
    |f of int
    |inf
    |ninf
;;

let add x y=
    match (x,y) with
    |(f a,f b)->f (a+b)
    |(f _,ninf)->ninf
    |(f _,inf)->inf
    |(ninf,inf)->failwith "forme indéterminée"
    |(ninf,_)->ninf
    |(inf,ninf)->failwith "forme indéterminée"
    |(inf,_)->inf
;;

let comp x y=
    match (x,y) with
    |(f a,f b)->a<b
    |(f _,ninf)->false
    |(f _,inf)->true
    |(inf,_)->false (* pour que la condition dans l'algo ne soit pas
vérifiée quand on compare deux infinis *)
    |(ninf,_)->true
;;
```

```
let bellman_ford g s a= (* g étant le graphe : une matrice contenant la
longueur de chaque arc, s la source, de type int, et a l'arrivée, de
type int *)
    let n=vect_length g in
    let d=make_vect n inf (* vecteur des majorants des distances pour
chaque sommet *)
    and p=make_vect n (-1) (* vecteur des prédécesseurs de chaque
sommet, -1 désigne NIL *)
    in d.(s)<-f 0;
    for i=1 to n-1 do
        for u=0 to n-1 do
            for v=0 to n-1 do
                let t=add d.(u) g.(u).(v) in
                if comp t d.(v) (* vrai si plus petit *)
                then
                    begin
                    d.(v)<-t;
                    p.(v)<-u
                    end
            done
        done
    done;
```

Relâchement

```
    for u=0 to n-1 do
        for v=0 to n-1 do
            let t=add d.(u) g.(u).(v) in
            if comp t d.(v)
            then failwith "il y a un cycle de poids négatif"
        done
    done;
```

Test de validité

```
    let rec aux l=
        if hd l=s
        then l
        else let t=p.(hd l) in
        if t<> -1
        then aux (t::l)
        else failwith "aucun chemin"
    in aux [a]
;;
```

Reconstitution du chemin choisi