▶ **HX4 & HX1 – Option Informatique**
Septième TP Caml
Vendredi 29 avril 2011

Petit Poucet dans le métro parisien

"There must be some way out of here," said the joker to the thief,
"There's too much confusion, I can't get no relief."
– Extrait de la chanson "All along the watchtower" de B. Dylan –

1 Petit Poucet dans son labyrinthe

Dans toute cette section un labyrinthe c'est un string vect. '#' représente un obstacle, '.' représente un chemin. Par exemple :

```
let laby = [|
"######";
".....#";
"###..#";
"#..#..";
"#.#..#";
"..##.#";
".#####"
 |];;
```

est un labyrinthe.

1.1 DFS : : Depth-first search

Voici un algorithme assez simple pour parcourir le labyrinthe :

```
DFS(case laby)
    Si la case n'a pas déjà été marquée
        marquer la case
        Pour chacune des cases voisines
            DFS(voisine)
```

▶ **Question 1** *Programmer l'algorithme du DFS.*

▶ **Question 2** *Programmer un algorithme qui donne le nombre de cases accessible depuis une certaine case.*

1.2 BFS : : Breadth-first search

Cette fois-ci, on ne s'intéresse plus seulement à savoir s'il existe un chemin mais à trouver le plus court chemin.

▶ **Question 3** *Écrire un programme qui étant donné les cases à distance n et les cases marquées déjà vues, donne les cases à distance $n+1$ et écrit dans un tableau pour chaque case à distance $n+1$ une case voisine à distance n.*

▶ **Question 4** *En déduire un programme qui affiche le plus court chemin entre deux points du labyrinthe.*

2 Métro parisien

2.1 DFS dans le metro parisien

▶ **Question 5** *Récupérer le fichier def.ml sur internet*

▶ **Question 6 *** *Essayer de comprendre la fonction traduit*

▶ **Question 7** *Écrire, en utilisant traduit, une fonction int_ligne, qui transforme une ligne où les stations sont des string, en ligne où les stations sont des int.*

▶ **Question 8** *Définir metro, un int list vect. Dans la i ième case on mets la liste des voisins de i sur le plan de metro.*

▶ **Question 9 *** *Écrire une fonction qui renvoie le plus court chemin entre deux stations données.*

▶ **Question 10** *Comment aller :*
- *de Rambuteau à Chemin Vert ?*
- *de Charonne à Lozere, ecole Polytechnique ?*
- *de Luxembourg à Lozere, ecole Polytechnique ?*
- *de Luxembourg à Bagneux ?*
- *de Luxembourg à nulle part ?*

.

▶ **HX4 & HX1 – Option Informatique**
Septième TP Caml
Vendredi 29 avril 2011

Petit Poucet dans le métro parisien

Un corrigé

▶ Question 1

```
let rec dfs (x,y) =
  let vois = it_list
    (fun ac (x,y) -> try (laby.(y).[x] ; (x,y)::ac) with |_ -> ac)
    [] [(x-1,y);(x+1,y);(x,y+1);(x,y-1)] in
  if laby.(y).[x] = '.' then
    begin
      laby.(y).[x] <- 'v' ;
      do_list dfs vois ;
    end
;;
```

▶ Question 2

```
let accessible (x,y) (x',y') laby =
  dfs (x,y) laby ;
  laby.(y).[x]='v'
```

▶ Question 3

```
let do_dist_n laby prec dn =
  let do_case c =
    let vois = vois laby c in
    do_list (fun (x,y) -> laby.(y).[x] <- 'm' ; prec.(y).(x) <- c) vois ;
    vois
  in
  flat_map do_case dn
;;

let rec do_dist laby c prec =
  let a = do_dist_n laby prec c in
  if a <> []
  then do_dist laby a prec
;;
```

▶ Question 4

```
let plus_court laby c1 (x,y) =
  let p = (make_matrix (vect_length laby) (vect_length laby) (0,0)) in
  let rec remonte (x,y) =
    if (x,y) = c1
    then []
    else (x,y)::(print_int x ; print_string "␣" ; print_int y ; print_string "\n"; remonte p.(y).(x))
  in
  do_dist laby [c1] p ;
  remonte (x,y)
;;
```

▶ Question 7

```
let int_lignes = map (fun x-> map traduit x) lignes ;;
```

▶ Question 8

```
let metro =
  let graphe = make_vect (!nb_trad+1) [] in

  let rec aj_ligne = function
    | [] -> ()
    | [a] -> ()
    | a :: ( t :: q) -> graphe.(a) <- t::graphe.(a) ; aj_ligne (t::q)
  in

  do_list aj_ligne int_lignes ;
  graphe
;;
\newpage
```

▶ Question 9

```
exception Pas_de_chemin ;;

let bfs graphe depart arrivee =

  let prec = make_vect (vect_length graphe) (-1) in

  let d = make_vect (vect_length graphe) 1000000 in

  let rec foo dist suiv = function
    | [] ->
      if suiv <> []
      then foo (dist+1) [] suiv
      else raise Pas_de_chemin
    | a::q ->
      if a = arrivee
      then dist
      else
        foo dist (it_list (fun x y -> if d.(y) = 1000000
            then (d.(y) <- dist+1 ; prec.(y) <- a ; y::x)
            else x) suiv graphe.(a)) q
  in
  d.(depart) <- 0 ;
  let dist = foo 0 [] [depart] in
  let rec remonte x l = if x = -1 then l else remonte (prec.(x)) (recupere x::l) in
  (dist,remonte arrivee [])
;;
```

▶ Question 10 .

```
- : int * string list =
 4,
 ["Rambuteau"; "Hotel de Ville"; "Saint Paul, Le Marais"; "Bastille";
  "Chemin Vert"]
- : int * string list =
 18,
 ["Charonne"; "Rue des Boulets"; "Nation"; "Gare de Lyon";
  "Maison-Alfort, Alfortville"; "Le Vert de Maisons"; "Villeneuve-Prairie";
  "Villeneuve-Triage"; "Villeneuve-Saint-Georges"; "Juvisy-sur-Orge";
  "Savigny-sur-Orge"; "Petit-Vaux"; "Gravigny-Balizy"; "Chilly-Mazarin";
  "Longjumeau"; "Massy-Palaiseau"; "Palaiseau"; "Palaiseau-Villebon";
  "Lozere, ecole Polytechnique"]
- : int * string list =
 16,
 ["Luxembourg"; "Port Royal"; "Denfert Rochereau"; "Cite Universitaire";
  "Gentilly"; "Bagneux"; "Bourg-la-Reine"; "Parc de Sceaux";
  "Croix de Berny"; "Antony"; "Fontaine Michalon"; "Les Baconnets";
  "Massy-Verrieres"; "Massy-Palaiseau"; "Palaiseau"; "Palaiseau-Villebon";
  "Lozere, ecole Polytechnique"]
- : int * string list =
 5,
 ["Luxembourg"; "Port Royal"; "Denfert Rochereau"; "Cite Universitaire";
  "Gentilly"; "Bagneux"]
#Uncaught exception: Pas_de_chemin
```