Quelques remarques sur le BE1 bis (Labyrinthe)

S7-2A

Sept. 2021

Version Élèves

2021-22

I Quelques remarques

• Deux heures ne suffisent pas du tout.
• Si le but est pédagogique, il faut alors pendre le temps d'expliquer. Sinon, il y a des sites plus compétents que nous pour balancer une solution !
• Le principe d'un parcours de graphe à la recherche d'une solution dans un espace d'états a été traité depuis très longtemps. Cela s'appelle **algorithme à essais successif**s (avec retours arrières et donc révocable, etc...)
• Ci-dessous une solution.

I-1 Parcours avec retours arrières

Un exemple typique de cette stratégie est la recherche d'une sortie dans un labyrinthe.

L'algorithme "à essais successifs" (AES) général :

```
Fonction AES_le_premier_succes_suffit;
Données : G : un graphe d'états ;
          Noeud_courant : le noeud (une variable = un état) courant que l'on traite dans cet appel
Résultat : Succès ou Echec (un booléen)
début
    si Noeud_courant est un état final alors
        renvoyer Succès
    sinon
        pour tous les Noeud_suivant successeurs de Noeud_courant dans G faire
            si Prometteur(G, Noeud_suivant) alors
                si AES_le_premier_succes_suffit(G, Noeud_suivant) = Succès alors
                    renvoyer Succès
                fin
            fin
        fin
        renvoyer Échec
    fin
fin
```

I-2 Le code Python correspondant au parcours du labyrinthe

Le code final est dans un fichier python joint à ce document.
• A noter

- Toute case a potentiellement 4 voisins possibles.
  Il suffit donc d'utiliser une liste de 4 couples qui donne les deltas (en (x, y))) pour se déplacer : voir la liste `voisins` dans le code.
- La valeur 5 dans une case semble inutile ! ! (voir aussi la constitution du trajet)
- IL NE FAUT PAS MARQUER les cases ici car on peut revenir en arrière pour trouver un autre chemin.
- Si vous êtes sages, je vous donnerais plus tard la version itérative (ainsi que le parcours en largeur !)
  ☞ Mais au bon entendeur salut :
  Ne pas laisser croire que l'on fait n'importe quoi jusqu'à ce que ça marche. Il y a des schémas de transformation de la récursivité. En particulier pour traiter la récursivité non terminale ! !

☞ Voire le fichier du code fourni.

I-3 Remarque sur la complexité de la recherche

On suppose ici un coût unitaire pour un essai (un appel récursif $\simeq$ un appel à *prometteur*). N est la taille de l'échiquier.

• Pour **N grand** : toutes les N^2 cases sont considérées pouvant avoir 4 voisins possibles et donc on aura une complexité $O(N^4) = O(2^{4\log N})$.

➞ Cette valeur est une sur-estimation (*pessimiste*) et se constate dans un cas d'échec (aucune solution pour N et (X,Y) de départ donnés).

II Heuristique du PCC

• Expliquer la numérotation
• Cela donne les PCCs