

DM1 - Informatique Option

Arsène MALLET - MP*

Sommaire

Question 1	1
Question 2	1
Question 3	2
Question 4	3
Question 5	3
Question 6	3
Question 7	3
Question 8	4
Question 9	4
Question 10	4
Question 11	5

Question 1

Une solution optimale pour $F = \{42\}$ est $\{1, 2, 4, 5, 10, 20, 21, 42\}$. Cette solution est de profondeur 7. (pas plus petit puisque en binaire $42 = \underline{101010}_2$)

Question 2

On suppose $s \in \mathbb{N}^* \cup \{+\infty\}$ (la cas où $s = 0$ étant triviale) la profondeur de la solution optimale (en notant $s = +\infty$ si il n'y a pas de solution au jeu). Afin de montrer l'équivalence, il faut montrer qu'à chaque p -ième tour de boucle **tant que**, A contient tous les états de profondeur p (où $p < s \in \mathbb{N}$). Montrons le par récurrence :

Initialisation : Si $p = 0$ alors $A = \{e_0\}$, et donc contient bien l'unique état de profondeur 0. L'initialisation est vérifiée. **Hérédité** : Soit $p \in \mathbb{N}$, on suppose que A contienne tous les états de profondeur p en entrant dans le p -ième tour de la boucle **tant que**, montrons que A contient tous les états de profondeur $p + 1$ en entrant dans le $(p + 1)$ -ième tour de la boucle **tant que**.

Comme $p < s$, $A \cap F = \emptyset$. Ainsi, grâce à la boucle **pour tout**, $B = \{s(x) \mid x \in A\}$, or comme A contient tous les états de profondeurs p , alors B contient

tout les états de profondeur $p + 1$. Comme en fin de boucle **tant que**, $A \leftarrow B$, en entrant dans le $(p + 1)$ -ième tour de la boucle **tant que**, A contient tous les états de profondeur $p + 1$

Conclusion : à chaque p -ième tour de boucle **tant que**, A contient tous les états de profondeur p (où $p < s \in \mathbb{N}$)

Ainsi, le parcours en largeur renvoie **VRAI** si et seulement si il existe une profondeur $p' \in \mathbb{N}$ tel que $A_{p'} \cap F \neq \emptyset$, si et seulement si il existe une solution.

Question 3

Soit $p \in \mathbb{N}$, la profondeur de la solution trouvée.

On considère le nombre d'états total que l'on ajoute à A au cours des itérations de boucle comme étant l'ordre de la complexité temporelle (puisque le nombre d'opérations élémentaires dépend de ce nombre total d'états).

Ainsi on ajoute en fait à A tous les états atteignables depuis e_0 . En considérant les potentiels doublons, et sachant qu'à chaque état de profondeur $i < p \in \mathbb{N}$, il y a deux choix d'états de profondeur $i + 1$ alors le nombre total d'états ajouté à A est majorée par:

$$\sum_{i=0}^p (2^i) = 2^{p+1} - 1$$

$$< 2^{p+1}$$

Ainsi la complexité temporelle est majorée par 2^{p+1}

On cherche maintenant un minorant du nombre d'états ajouté à A . Pour cela on peut montrer que pour tout $i \in \mathbb{N}$, on ajoute tout les entiers entre 1 et 2^i en $2i$ itérations de boucles :

Soit $i \in \mathbb{N}$ et $n \in \llbracket 1, 2^i \rrbracket$. On note $\underline{b_k b_{k-1} \dots b_0}_2$ où $k \leq i, b_k = 1$ et $\forall l \in \llbracket 0, k-1 \rrbracket, b_l \in \{0, 1\}$ la représentation binaire de n Ainsi,

$$n = \sum_{l=0}^k (b_l 2^l) = b_0 + 2(b_1 + 2(\dots + 2b_k))$$

et, comme $\forall l \in \llbracket 0, k-1 \rrbracket, b_l \in \{0, 1\}$, n est atteignable en maximum $2k$ étapes (en partant de la dernière parenthèse, on multiplie par deux et/ou on ajoute 1 au fur et à mesure). Ainsi en p itérations de boucle, on ajoute au minimum l'ensemble $\llbracket 1, 2^{\lfloor p/2 \rfloor} \rrbracket$ à A , d'où le minorant. Ainsi la complexité temporelle est bornée à la fois inférieurement et supérieurement par deux fonctions exponentielles en p .

La complexité spatiale est de l'ordre du cardinal maximal de A au cours des itérations, qui est majoré par le nombre d'états de profondeur p , lui-même majoré par 2^p (séquence de p éléments et 2 choix par états). De plus, comme on ajoute au moins $2^{\lfloor p/2 \rfloor}$ éléments à A en p étapes (cf. complexité temporelle), il y

a au moins une itération de boucle où $Card(A) \geq \frac{2^{\lfloor p/2 \rfloor}}{p}$ éléments. Ainsi la complexité spatiale est également bornée à la fois inférieurement et supérieurement par deux fonctions exponentielles en p .

Question 4

```
let bfs () =
  let rec dissimulatewhile a b p=
    match a with
    | [] -> (match b with
              | [] -> -1
              | _ -> dissimulatewhile b [] p+1)
    | h::t -> if final h then p else dissimulatewhile t (List.append (suivants(h)) b) p
  in dissimulatewhile [initial] [] 0 ;;
```

Question 5

Comme montrer à la question 2, l'ensemble A contient tous les états possible de profondeur p en entrant dans la p -ième itération de boucle. Ici la p -ième itération correspond à un appel `dissimulateWhile` de second argument `[]` et de troisième argument p . En effet ce troisième indice est incrémenté de 1 à chaque fois que a est complètement vidé et que $a \leftarrow b$. Ainsi tous les états de profondeur p sont vérifiés avant de tester les états de profondeur $p + 1$, ainsi `bfs` renvoie toujours une profondeur optimale lorsqu'une solution existe.

Question 6

Soit $m, p \in \mathbb{N}$ tq $p \leq m$

Question 7

```
let ids () =
  let rec dfs m e p =
    if p > m then false
    else if final e then true else
      let flag = ref false in
      let voisins = ref (suivants e) in
      while (!voisins <> []) && (!flag = false) do
        flag := dfs m (List.hd !voisins) (p+1);
        voisins := List.tl !voisins
      done;
      !flag
  in
  let m = ref 0 in
  while not (dfs !m initial 0) do
    m := !m + 1;
```

```

done;
!m ;;

```

Question 8

Comme montrer à la question 6, $\text{DFS}(m, e_0, 0)$ renvoie VRAI si et seulement si une solution de profondeur inférieure ou égale à m existe. Or comme ids incrémente à chaque fois la valeur de m de 1 (en partant de 0) et fait ensuite appel à $\text{DFS}(m, e_0, 0)$ alors tant qu'une solution de profondeur p n'existe pas m continue d'augmenter. Ainsi lorsque $\text{DFS}(m, e_0, 0)$ renvoie VRAI pour la première fois, alors ids renvoie la valeur de m tel que $\text{DFS}(m, e_0, 0) = \text{true}$, c'est donc d'une part qu'une solution existe, mais également que c'est une solution de profondeur optimale.

Question 9

Question 10

```

let min = ref 0 ;;

let rec dfsstar m e p =
  let c = p + (h e.value) in
  if c > m then
    ((if c < !min then min := c) ; false)
  else if (final e) then true else
    let flag = ref false in
    let voisins = ref (suivants e) in
    while (!voisins <> []) && (!flag = false) do
      flag := dfsstar m (List.hd !voisins) (p+1);
      voisins := List.tl !voisins
    done;
    !flag ;;

let idastar () =
  let m = ref (h initial.value) in
  let flag = ref false in
  let m_final = ref !m in
  while !m <> max_int && not !flag do
    begin
      min := max_int ;
      if dfsstar !m initial 0 then (flag := true ; m_final := !m);
      m := !min;
    end
  done;
  !m_final ;;

```

Question 11

En prenant:

$$\begin{array}{lcl} h & : & E \rightarrow \mathbb{N} \\ x & \mapsto & \begin{cases} 0, & \text{si } n = t \\ 1, & \text{sinon} \end{cases} \end{array}$$

On peut montrer que h est bien admissible :