TRAVAUX DIRIGÉS

Introduction à l'Informatique

Exercice 1

Dire si chacun des éléments suivants est ou non un algorithme. Justifier :

Entrées : $i \in \mathbb{N}$
Sorties : $42 - i$
1 $c \leftarrow 0$
2 **tant que** $i \neq 42$ **faire**
3 $\quad i \leftarrow i + 1$
4 $\quad c \leftarrow c + 1$
5 **renvoyer** c

(a) Calcul de 42-i sans soustraction

Tri de jeu de cartes : prenez un jeu de cartes. Mettez toutes les cartes sur un emplacement de défausse, et réservez un autre emplacement pour la pile triée (qui est donc vide au début). Parcourez la défausse, et prenez-y la plus petite carte. Mettez-la au sommet de la pile triée. Répétez les deux étapes précédentes jusqu'à ce que la défausse soit vide. Prenez le jeu trié sur la pile !

(b) Tri par sélection

Chocolat chaud praliné
Ingrédients : 100g de chocolat praliné, 1/2L de lait, sucre, 1 pointe de crème
Recette : Faire fondre le chocolat au bain marie tout en fouettant doucement jusqu'à ce qu'il soit homogène et onctueux. Faire chauffer le lait jusqu'à frémissement. Ajouter alors le chocolat fondu, fouetter. Ajouter une pointe de crème.
Servir chaud !

(c) Recette de chocolat chaud

Entrées : A un agorithme
Sorties : Vrai si A termine en temps fini sur toute entrée, Faux sinon
1 **pour chaque** *entrée possible* **faire**
2 $\quad$ appeler A(*entrée*)
3 $\quad$ **si** *cet appel ne termine pas* **alors**
4 $\quad\quad$ **renvoyer** Faux
5 **renvoyer** Vrai

(d) Algorithme de l'Arrêt

Entrées : $n \in \mathbb{N}^*$, T tableau de n entiers indexé de 0 à $n-1$
Sorties : m
1 $m \leftarrow T[0]$
2 **pour** i *allant de* 1 *à* $n-1$ **faire**
3 $\quad$ **si** $T[i] < m$ **alors**
4 $\quad\quad m \leftarrow T[i]$
5 **renvoyer** m

(e) Parcours d'un tableau

Entrées : $x \in \mathbb{N}^*$, $y \in \mathbb{N}^*$
Sorties : PGCD(x, y)
1 Compter de 0 à 10^{100}
2 $p \leftarrow$ EUCLIDE (x, y)
3 Compter à l'envers de 10^{100} à 0
4 **renvoyer** p

(f) Algorithme d'Euclide... ?

Exercice 2

1. **a.** Écrire un algorithme qui prend en entrée un entier positif n, et renvoie en sortie $\sum_{i=1}^{n} i$.
   b. On suppose que l'on a une fonction `caclule_f` qui calcule une fonction mathématique $f : \mathbb{N}^+ \to \mathbb{Z}$. Écrire un algorithme qui prend en entrée un entier positif n, et renvoie en sortie $\sum_{i=1}^{i=n} f(i)$.
2. En mathématiques, on définit également le symbole Π. C'est l'analogue de Σ pour le produit : $\prod_{i=1}^{n} a_i = a_1 \times a_2 \times \cdots \times a_n$.
   a. Refaire les questions précédentes avec Π au lieu de Σ .
   b. Dans certains cas, le calcul de $\prod_{i=1}^{n} a_i$ peut-être fortement accéléré : il n'est même pas nécessaire de calculer tous les a_i pour calculer le produit total. Proposer un exemple d'une telle situation, et proposer une modification de votre algorithme précédent prenant ce comportement en compte.

Exercice 3

On s'intéresse à l'algorithme d'Euclide donné en cours. On nommera ses entrées x et y dans cet ordre.

1. Écricre l'algorithme donné en cours en langage de pseudo-code *(sans récursion !*[i]*)*. Vérifier son bon fonctionnement sur des exemples.
2. Traduire ce pseudo-code dans votre langage de programmation favori. Vérifier son bon fonctionnement sur des exemples.
3. On se place dans le cas $m < n$.
   a. Montrer que dans ce cas, on effectue un calcul de reste superflu.
   b. Proposer une modification de l'algorithme qui évite ce calcul superflu, et tracer le nouveau graphe de flot de controle.
4. (Bonus.) Si vous êtes familier·e avec la notion de récursion, réécricre l'algorithme de manière récursive.

i. Si vous ne comprenez pas cette consigne, vous êtes dans le vert !

Exercice 4

Tracer le graphe de flot de contrôle de ces deux algorithmes [i]. Commenter.

Algorithme 1 : est_premier

Entrées : $x \in \mathbb{N}$
Sorties : "PREMIER" si x est premier, "NON-PREMIER" et une explication sinon

```
si x ≤ 1 alors
    renvoyer "NON-PREMIER car ≤ 1"
d ← 2
tant que Vrai faire
    si d ≥ x alors
        Quitter la boucle Tant Que
    sinon si d divise x alors
        renvoyer "NON-PREMIER car d divise x"
    d ← d + 1
renvoyer "PREMIER"
```

Algorithme 2 : est_premier_bis

Entrées : $x \in \mathbb{N}$
Sorties : "PREMIER" si x est premier, "NON-PREMIER" et une explication sinon

```
si x ≤ 1 alors
    renvoyer "NON-PREMIER car ≤ 1"
d ← 1
pour d allant de 2 à x − 1 faire
    si d divise x alors
        renvoyer "NON-PREMIER car d divise x"
renvoyer "PREMIER"
```

i. On dit qu'un entier naturel est premier s'il admet exactement deux diviseurs positifs distincts : 1 et lui-même.