Complexité et algorithmique - Miniprojet 1

Basil dalié

Novembre 2018

Chapitre 1

Introduction

Dans le cadre du module Complexité et algorithmique inscrit dans le programme pédagogique de quatrième année en Siences Informatiques à l'école Polytech Nice-Sophia, il nous a été demandé de réaliser quatre algorithmes pour résoudre le problème de la recherche du $n^{\text{ème}}$ bit sur une machine de turing.

Ce problème consiste à retourner, pour une donnée de longueur N constitué d'un mot m sur l'alphabet $\{A, B, C, D\}$ suivi de la lettre X puis d'un nombre n, la $n^{\text{ème}}$ lettre du mot m ou alors X si le nombre n est supérieur à la longueur de m (e.g. si m est AABA, le résultat doit être B si n est égal à 3 et X si n est égal à 5).

Chacun de ces algorithmes devaient respecter des contraintes distinctes : pour le premier, la machine de turing dispose de deux bandes, contre une bande pour les trois suivants. Le second devait avoir une complexité de l'ordre de $O(N^2)$ et les deux suivants de $O(N \log N)$. Pour le troisième algorithme la donnée n devait être donnée en binaire, alors qu'elle devait être donnée en unaire pour les trois autres.

Au cours de ces pages, vous pourrez trouver pour chacun des algorithmes réalisés, sa description, ainsi que la preuve et l'analyse de sa complexité

Chapitre 2

Première solution

2.1 Description

La première solution pour la recherche du $n^{\text{ème}}$ bit est une machine de Turing à deux bandes, le nombre n est donné en unaire. la complexité de l'algorithme est de l'ordre de $O(N)$.

2.1.1 Alphabet

Pour cette solution, l'ensemble des symboles T est égal à l'union de l'ensemble des symboles de données I avec l'ensemble $\{\_\}$ où le symbole $\_$ représente le symbole blanc. Ses éléments sont : $\{A, B, C, D, X, 1, \_\}$, le symbole 1 représente l'unique chiffre pour représenter la donnée n en unaire.

2.1.2 Ensemble d'état

La machine de turing comporte 11 états sans compter l'état d'erreur $@NO$.

2.1.3 Fonction de transition

état	l1	l2	e1	e2	d1	d2	nouvel état
MOVE_TO_X_ON_TAPE_ONE	A	_	A	_	R	S	MOVE_TO_X_ON_TAPE_ONE
	B	_	B	_	R	S	MOVE_TO_X_ON_TAPE_ONE
	C	_	C	_	R	S	MOVE_TO_X_ON_TAPE_ONE
	D	_	D	_	R	S	MOVE_TO_X_ON_TAPE_ONE
	X	_	X	_	R	S	WRITE_N_ON_TAPE_TWO
WRITE_N_ON_TAPE_TWO	1	_	_	1	R	R	WRITE_N_ON_TAPE_TWO
	_	_	_	_	L	L	MOVE_TO_WORD_RIGHT_ON_TAPE_ONE
MOVE_TO_WORD_RIGHT_ON_TAPE_ONE	_	1	_	1	L	S	MOVE_TO_WORD_RIGHT_ON_TAPE_ONE
	X	1	X	1	L	S	MOVE_TO_WORD_LEFT_ON_TAPE_ONE
MOVE_TO_WORD_LEFT_ON_TAPE_ONE	A	1	A	1	L	S	MOVE_TO_WORD_LEFT_ON_TAPE_ONE
	B	1	B	1	L	S	MOVE_TO_WORD_LEFT_ON_TAPE_ONE
	C	1	C	1	L	S	MOVE_TO_WORD_LEFT_ON_TAPE_ONE
	D	1	D	1	L	S	MOVE_TO_WORD_LEFT_ON_TAPE_ONE
	_	1	_	1	R	S	MOVE_TO_WORD_LEFT_ON_TAPE_TWO
MOVE_TO_WORD_LEFT_ON_TAPE_TWO	A	1	A	1	S	L	MOVE_TO_WORD_LEFT_ON_TAPE_TWO
	B	1	B	1	S	L	MOVE_TO_WORD_LEFT_ON_TAPE_TWO
	C	1	C	1	S	L	MOVE_TO_WORD_LEFT_ON_TAPE_TWO
	D	1	D	1	S	L	MOVE_TO_WORD_LEFT_ON_TAPE_TWO
	A	_	A	_	S	R	ERASE_RIGHT
	B	_	B	_	S	R	ERASE_RIGHT
	C	_	C	_	S	R	ERASE_RIGHT
	D	_	D	_	S	R	ERASE_RIGHT
ERASE_RIGHT	A	1	A	1	R	R	ERASE_RIGHT
	B	1	B	1	R	R	ERASE_RIGHT
	C	1	C	1	R	R	ERASE_RIGHT
	D	1	D	1	R	R	ERASE_RIGHT
	A	_	_	_	R	R	ERASE_RIGHT
	B	_	_	_	R	R	ERASE_RIGHT
	C	_	_	_	R	R	ERASE_RIGHT
	D	_	_	_	R	R	ERASE_RIGHT
	X	_	_	_	L	L	MOVE_TO_NTH
	X	1	X	1	L	L	ERASE_LEFT
MOVE_TO_NTH	_	_	_	_	L	L	MOVE_TO_NTH
	A	1	A	1	L	L	ERASE_LEFT
	B	1	B	1	L	L	ERASE_LEFT
	C	1	C	1	L	L	ERASE_LEFT
	D	1	D	1	L	L	ERASE_LEFT
ERASE_LEFT	A	1	_	1	L	L	ERASE_LEFT
	B	1	_	1	L	L	ERASE_LEFT
	C	1	_	1	L	L	ERASE_LEFT
	D	1	_	1	L	L	ERASE_LEFT
	_	_	_	_	R	R	ERASE_TAPE_TWO
ERASE_TAPE_TWO	A	1	A	_	S	R	ERASE_TAPE_TWO
	B	1	B	_	S	R	ERASE_TAPE_TWO
	C	1	C	_	S	R	ERASE_TAPE_TWO
	D	1	D	_	S	R	ERASE_TAPE_TWO
	_	1	_	_	S	R	ERASE_TAPE_TWO
	A	_	A	_	S	L	MOVE_TO_NTH_FINAL
	B	_	B	_	S	L	MOVE_TO_NTH_FINAL
	C	_	C	_	S	L	MOVE_TO_NTH_FINAL
	D	_	D	_	S	L	MOVE_TO_NTH_FINAL
MOVE_TO_NTH_FINAL	_	_	_	_	R	S	MOVE_TO_NTH_FINAL
	A	_	A	_	S	S	YES
	B	_	B	_	S	S	YES
	C	_	C	_	S	S	YES
	D	_	D	_	S	S	YES

2.1.4 Description de l'algorithme