Compresseur de Huffman

Contents

1 Analyse

1.1 Types de données abstraits

Nom: ArbreDeHuffman

Paramètre: Element

Utilise: **Naturel**, **Booleen**

Opérations:

arbreDeHuffman: Element $\times$ **Naturel** $\rightarrow$ ArbreDeHuffman

combiner: ArbreDeHuffman $\times$ ArbreDeHuffman $\rightarrow$ ArbreDeHuffman

obtenirPondération: ArbreDeHuffman $\rightarrow$ **Naturel**

obtenirElement: ArbreDeHuffman $\nrightarrow$ Element

estUneFeuille: ArbreDeHuffman $\rightarrow$ **Booleen**

obtenirArbreGauche: ArbreDeHuffman $\nrightarrow$ ArbreDeHuffman

obtenirArbreDroit: ArbreDeHuffman $\nrightarrow$ ArbreDeHuffman

Axiomes:

- obtenirElement (arbreDeHuffman(élément,pondération)) = élément
- obtenirPondération (arbreDeHuffman(élément,pondération)) = pondération
- obtenirPondération (combiner(arbreGauche,arbreDroit)) = obtenirPondération(arbreGauche)+obtenirPondération(arbreDroit)
- estUneFeuille (arbreDeHuffman(élément,pondération))
- non estUneFeuille (combiner(arbreGauche,arbreDroit))
- obtenirArbreGauche (combiner(arbreGauche,arbreDroit)) = arbreGauche

- obtenirArbreDroit (combiner(arbreGauche,arbreDroit)) = arbreDroit

Préconditions: obtenirElement(arbre)estUneFeuille(arbre)

non estUneFeuille(arbre)

obtenirArbreGauche(arbre):obtenirArbreDroit(arbre): non estUneFeuille(arbre)

Nom: FileDePriorité

Paramètre: Element

Utilise: Naturel, Boolean

fileDePriorité: $\rightarrow$ $\rightarrow$ FileDePriorité

inserer: FileDePriorité $\times$ Element $\times$ NaturelNonNul $\rightarrow$ $\rightarrow$ FileDePriorité

supprimerPremier: FileDePriorité $\rightarrow$ $\rightarrow$ FileDePriorité

supprimerDernier: FileDePriorité $\rightarrow$ $\rightarrow$ FileDePriorité

obtenirPremier: FileDePriorité $\rightarrow$ $\rightarrow$ Element

obtenirDernier: FileDePriorité $\rightarrow$ $\rightarrow$ Element

estVide: FileDePriorité $\rightarrow$ $\rightarrow$ Boolean

longueur: FileDePriorité $\rightarrow$ $\rightarrow$ Naturel

: estVide(fileDePriorité())

: non estVide(inserer(fileDePriorité(), el, 1))

: obtenirPremier(inserer(fileDePriorité(), el, 1)) = el

: obtenirDernier(inserer(fileDePriorité(), el, 1)) = el

: longueur(fileDePriorité()) = 0

: longueur(inserer(l, e, p)) = 1 + longueur(l)

: supprimerDernier()

supprimerPremier(l): non(estVide(l))

supprimerDernier(l): non(estVide(l))

obtenirPremier(l): non(estVide(l))

obtenirDernier(l): non(estVide(l))

Nom: Octet

Utilise: **Booleen**, Bit, **Naturel**

Opérations: octet: Bit × Bit × Bit × Bit × Bit × Bit × Bit × Bit → Octet

estEgal: Octet × Octet → **Booleen**

obtenirBit: Octet × 1..8 → Bit

Sémantiques: octet: Crée un octet avec les 8 bits spécifiés.

estEgal: Vérifie si les deux octets sont égaux.

obtenirBit: Obtient le bit à la position spécifiée dans l'octet (position de 0 à 7).

Axiomes:
- obtenirBit (00000001, 1) = 1
- estEgal (octet1,octet2) = (obtenirBit (octet1, 1..8) = obtenirBit (octet2, 1..8))

Nom: Statistiques

Paramètre: Element

Utilise: **Naturel**, **Booleen**, Ensemble

Opérations: statistiques: Ensemble d'Element → Statistiques

incrementerOccurenceElement: Statistiques × Element → Statistiques

obtenirOccurenceElement: Statistiques × Element → **Naturel**

Axiomes:
- obtenirOccurenceElement(statistiques(set), elem) = 0
- obtenirOccurenceElement(incrementerOccurenceElement(stat, elem) = obtenirOccurenceElement(stat, elem) + 1

Nom: TableDeCodage

Paramètre: Element, Code

Utilise: **Naturel**, **Booleen**

Opérations: TableDeCodage: TableDeCodage →

estVide: TableDeCodage → **Booleen**

elementPresent: TableDeCodage × Element → **Booleen**

codePresent: TableDeCodage × Code → **Booleen**

ajouterElement: TableDeCodage × Element ↛ TableDeCodage

elementPossedeCode: TableDeCodage × Element ↛ **Booleen**

assignerCodeElement: TableDeCodage × Element × Code ↛ TableDeCodage

obtenirCodeElement: TableDeCodage × Element ↛ Code

obtenirElementCode: TableDeCodage × Code ↛ Element

Axiomes:
- non(estVide(ajouterElement(table, elem)))
- estPresent(ajouterElement(table, elem), elem)
- elementPresent(ajouterElement(stat, elem), elem)
- codePresent(assignerCodeElement(table, elem, code), code)
- obtenirElemCode(assignerCodeElement(table, elem, code), code) = elem
- obtenirCodeElem(assignerCodeElement(table, elem, code), elem) = code
- non(codePresent(table, code)) ⇒ obtenirElementCode(table, code) = NIL

Préconditions: ajouterElement(table, elem)non(elementPresent(table, elem))

elementPresent(table, elem)

elementPossedeCode(table, elem):assignerCodeElement(table, elem, code): non(codePresent(table, code)) ET elementPresent(table, elem)

obtenirCodeElement(table, elem): elementPossedeCode(table, elem) ET elementPresent(table, elem)

1.2 Analyse descendante