ESP de Nouakchott
Exercices d'algorithmique

6 mars 2014

1 formalisme d'un algorithme, algorithmes itératifs, calcul de complexité

1. Chercher un élément dans un tableau

Algorithme 1: search(T : tableau de n élément, E : Élément, Found : booléen)

```
1: Entrée : T, E
2: PréC : {n ≥ 1}
3: Sortie : Found
4: PostC : {Found = (E ∈ T[1, ..., n])}
5: Found ← (T[1] = E)
6: I ← 1
7: Tant que (¬Found ∧ I ≠ n)
8:     Found ← (T[I+1] = E)
9:     I ← I + 1
```

Complexité en pire de cas :
- ligne 5 : 1 une comparaison et une affectation (total : 2 opérations)
- ligne 6 : 1 affectation (total : 1 opération)
- ligne 7) au pire de cas (l'élément n'existe pas) :
  (1 négation + 1 comparaison + 1 comparaison + l'opération logique (et)) $\times n = 4 \times n$
- ligne 8 : (1 addition + 1 comparaison + 1 affectation) × (n-1) $= 3 \times n - 3$
- ligne 9 : (1 addition + 1 affectation) × (n-1) $= 2 \times n - 2$
- nombre d'instructions total : $N_{search}(n) = 2 + 1 + 4 \times n + 3 \times n - 3 + 2 \times n - 2 = 9 \times n + 1$

2. algo retournant la dernière position d'un élément dans un tableau

Algorithme 2: lastPosition(T : tableau de n élément, E : Élément, Pos : Entier)

```
1: Entrée : T, E
2: PréC : {n ≥ 1}
3: Sortie : Pos
4: PostC : {(Pos = Max(k ∈ {1, ..., n}  t.q  T[k] = E) ∨ ((Pos = (n + 1)) ∧ (∄ k  t.q  T[k] = E)))}
5: Pos ← n + 1
6: I ← n
7: Tant que ((Pos = (n + 1)) ∧ (I ≥ 1))
8:     Si T[I] = E Alors
9:         Pos ← I
10:    I ← I − 1
```

complexité en pire de cas : A faire

3. Algorithme vérifiant si la somme des m premier éléments d'un tableau est égale à la moitié de la somme de tous les éléments

Algorithme 3: isHalfOfGlobalSum(T : tableau de n élément, m : Entier, B : booléen)

1: Entrée : T, m
2: PréC : $\{(n \geq 2) \wedge (1 \leq m < n)\}$
3: Sortie : B
4: PostC : $\{B = (\sum_{i=1}^{m} T[i] = \frac{\sum_{i=1}^{n} T[i]}{2})\}$
5: $sum \leftarrow 0$
6: $sum\_m \leftarrow 0$
7: Pour $i \leftarrow 1$ à m Faire
8: $\quad sum \leftarrow sum + T[i]$
9: $\quad sum_m \leftarrow sum\_m + T[i]$
10: Pour $i \leftarrow m+1$ à n Faire
11: $\quad sum \leftarrow sum + T[i]$
12: Si $sum = 2 \times sum\_m$ Alors
13: $\quad B \leftarrow vrai$
14: Sinon
15: $\quad B \leftarrow faux$

4. Comparaison de deux tableaux

Algorithme 4: compare(T_1 : tableau de n_1 élément, T_2 : tableau de n_2 élément, Rep : Entier)

1: Entrée : T_1, T_2
2: PréC : $\{(n_1 \geq 1) \wedge (n_2 \geq 1)\}$
3: Sortie : Rep
4: PostC : $\{(Rep = 0 \wedge T_1 = T_2) \vee (Rep = -1 \wedge T_1 < T_2) \vee (Rep = 1 \wedge T_1 > T_2)\}$
5: $Rep \leftarrow 0$
6: Si $n_1 < n_2$ Alors
7: $\quad Rep \leftarrow -1$
8: Sinon SI $n_1 > n_2$ Alors
9: $\quad Rep \leftarrow 1$
10: Sinon
11: $\quad Rep \leftarrow 0$
12: $I \leftarrow 1$
13: Tant que $((Rep = 0)) \wedge (I \leq n_1))$
14: $\quad$ Si $T_1[I] < T_2[I]$ Alors
15: $\quad\quad Rep \leftarrow -1$
16: $\quad$ Sinon SI $T_1[I] > T_2[I]$ Alors
17: $\quad\quad Rep \leftarrow 1$
18: $\quad I \leftarrow I + 1$

5. convertir un binaire à un décimal

Algorithme 5: BinaryToDec(T : tableau de n booléen, Dec : Entier)

1: Entrée : T
2: PréC : $\{n \geq 1\}$
3: Sortie : Dec
4: PostC : $\{(T[n]T[n-1]\dots T[1])_2 = (Dec)_{10}\}$
5: $Dec \leftarrow 0$
6: Pour $i \leftarrow 1$ jusqu'à n faire
7: SI $(T[i] = vrai)$ Alors
8: $Dec \leftarrow dec + pow(2, i-1)$

Complexité en meilleur de cas et en pire de cas :

2 Listes

1. ajout d'un élément à la fin d'une liste

Algorithme 6: Add(L : Liste, E : Elément)

1: Entrée : L, E
2: PréC : $\{L \neq NULL\}$
3: Sortie : L
4: PostC : $\{L = L \cup \{E\}\}$
5: $Q \leftarrow Allouer(Q)$(Q est un pointeur sur Place)
6: $Q->val \leftarrow E$
7: $Q->suiv \leftarrow NULL$
8: $Temp \leftarrow Allouer(Temp)$
9: $Temp \leftarrow L$
10: Tant que $(Temp->suiv) \neq NULL$
11: $Temp \leftarrow Temp->suiv$
12: $(Temp->suiv) \leftarrow Q$

2. Fusion des deux listes élément par élément

Algorithme 7: Add($L1$, $L2, L3$: Liste)

1: Entrée : L1, L2
2: PréC : {}
3: Sortie : $L3$
4: PostC : $\{L3 = L1(1)L2(1)L1(2)\ldots\}$
5: Si $(L1 = NULL \wedge L2 = NULL)$ Alors
6: $L3 \leftarrow NULL$
7: Sinon Si $(L1 \neq NULL \wedge L2 = NULL)$ Alors
8: $L3 \leftarrow L1$
9: Sinon Si $(L2 \neq NULL \wedge L1 = NULL)$ Alors
10: $L3 \leftarrow L2$
11: Sinon
12: $Q \leftarrow Allouer(Q)$
13: $L3 \leftarrow L1$
14: $Q \leftarrow L1$
15: $pos \leftarrow 1$
16: Tant que $(L1 \neq NULL \wedge L2 \neq NULL)$ Faire
17: Si $pos = 1$ Alors
18: $L1 \leftarrow L1 -> suiv$
19: $(Q -> suiv) \leftarrow L2$
20: $pos \leftarrow 2$
21: Sinon
22: $L2 \leftarrow (L2 -> suiv)$
23: $(Q -> suiv) \leftarrow L1$
24: $pos \leftarrow 1$
25: $Q \leftarrow (Q -> suiv)$

3. évaluation d'une expression postfix

Algorithme 8: Evaluer(expr : tableau de caractères, val : réel)

```
1:  Entrée : expr
2:  PréC : {}
3:  Sortie : val
4:  PostC : {}
5:  P ← PileVide()
6:  i ← 1
7:  Tant que (i ≤ longueur(expr)) Faire
8:     c ← expr[i]
9:     Si c est une valeur Alors
10:       empiler(P,c)
11:    Sinon Si c est un opérateur binaire Alors
12:       x ← sommet(P)
13:       dépiler(P)
14:       y ← sommet(P)
15:       dépiler(P)
16:       z ← x c y (appliquer l'opérateur binaire sur x et y)
17:       empiler(P, z)
18:    Sinon Si c est un opérateur unaire Alors
19:       x ← sommet(P)
20:       dépiler(P)
21:       z ← x c
22:       empiler(P, z)
23:       empiler(P, z)
24:    i ← i + 1
25: FinTq
26: val ← sommet(P)
```