Programmation Fonctionnelle – Projet 2017

Xuehui JIA

mars 2017

1 L'introduction

C e projet a proposé un manière de conserver de grandes chaînes de caractères , il propose une alternative à la notion usuelle de chaîne de caractères que nous appelons string_builder. c'est une structure d'arbre binaire ,les feuilles sont les chaînes de caractère plus petit . On peut lire le texte, chercher ou modifier quelques mots dans ce texte par parcourir ce arbre binaire , ce manière permet les opération plus simple et plus efficace que String.

Pour réaliser ce nouveau structure de grandes chaînes de caractères, on utilise l'algorithme d'arbre binaire optimalqui est décrié comme l'algorithme ci-dessous :

— Transformer l'arbre en une liste de ses feuilles respectant l'ordre.

— Tant qu'il existe au moins 2 éléments dans la liste :

1. trouver les deux éléments successifs dont la concaténation a le coût le plus faible,
2. les retirer, les concaténer et insérer le résultat à leur position.

— Concaténer les deux derniers éléments.

2 La réalisation

2.1 Question 1. le type string_builder.

Définir la fonction word qui prend en argument une chaîne de caractères et qui renvoie le string_builder correspondant. Définir la fonction concat qui prend en argument deux string_builder et qui renvoie le nouveau string_builder résultant de leur concaténation.

L a définition de string_builder: c'est un arbre de type (int*string)

int: la longueur du string , string : la chaîne de caractère

La fonction word permet de transformer un string à string_builder . Il générer un feuille Noeud((length,str),Vide,Vide).

La fonction concat prend en argument deux string_builder et qui renvoie le nouveau string_builder résultant de leur concaténation, j'ajoute un fonction ran_int(), qui produit un int aléatoire entre 0,1 pour indiquer l'ordre de abr1 et abr2, soit abr1 à gauche, soit abr2 à gauche .

2.2 Question 2. la fonction char_at

Définir une fonction char_at qui prend en argument un entier i et un string_builder représentant le mot [c0 ; . . . ; cn1], et qui renvoie le caractère ci .

P our prendre un certain caractère , je pense il faut transformer un arbre à un chaîne de caractère , donc j'ecrit une fonction auxiliaire to_list pour recevoir un string corroborant , et après , on peut utiliser String.get pour prendre ce caractère

2.3 Question 3. la fonction sub_string

Définir une fonction sub_string qui prend en arguments un entier i, un entier m et un string_builder sb représentant le mot [c0 ; . . . ; cn1] et qui renvoie un string_builder représentant le mot [ci ; . . . ; ci+m1], c'est-à-dire la sous-chaîne de c débutant au caractère i et de longueur m.

T ransformer l'arbre à une string et utiliser la function String.sub

2.4 Question 4. Définir la fonction cost

Définir la fonction cost qui prend en argument un string_builder et qui renvoie son coût selon la définition précédente.

P our obtenir le coût , il faut connaître le pronondeur des feuille ,donc j'ecrit une fonction niveau qui retoure un nouveau arbre , son "int" ne présente plus la longueur de string mais le profonduer, après je donne ce nouveau arbre à fonction cost pour calculer par parcourir .

2.5 Question 5.la fonction ramdom_string

Définir une fonction random_string qui prend en argument un entier i et qui génère un arbre de profondeur i.

M on algorithme est ci-dessous :

1. obtenir le profondeur d'arbre: une fonction profondeur prend en argument (niveau abr), parcourir le et retour le plus grand niveau comme le profondeur de ce arbre.

2.obtenir un chaîne de caractère aléatoire :

-utiliser Char.escaped (Char.chr (97+(Random.int 26))) pour obtenir un char aléatoire

-utiliser Random.int pour obtenir un int aléatoire indiquant la longueur du string

-générer une liste de caractère , son longueur est aléatoire et chaque de son élément est un caractère aléatoire .

-transformer la liste à un string en utilisant String.concat

3.définir la fonction random_string:prendre un argument abr pour conserver le résultat , chaque fois de récurrence on combine abr avec un feuille (string aléatoire),jusqu'à son profondeur soit égale à i.

2.6 Question 6. la fonction list_of_string

Définir une fonction list_of_string qui prend en argument un string_builder et qui renvoie la liste des chaînes de caractères dans le même ordre que dans l'arbre (parcours infixe).

O n parcourt l'arbre de manière infixe ,donc on prend le noeud, après ,on prend l'enfant gauche , et après , on prend l'enfant droit. Chaque fois qu'on rencontre un feuille (Noeud(a,Vide,Vide)),on ajoute son string dans une liste.

2.7 Question 7. la fonction balance

Définir une fonction balance qui prend en argument un string_builder et qui renvoie un nouveau string_builder équilibré.

C omme l'algo fourni :

— Transformer l'arbre en une liste de ses feuilles respectant l'ordre.
— Tant qu'il existe au moins 2 éléments dans la liste :
  1. trouver les deux éléments successifs dont la concaténation a le coût le plus faible,
  2. les retirer, les concaténer et insérer le résultat à leur position.
— Concaténer les deux derniers éléments.

Mes étapes sont ci-dessous :

-1.D'abord , il faut transformer la liste du list_of_String à une nouvelle liste de type (string ,length),on ajout la longueur du string .

-2.pour obtenir le résultat , il faut faire une fonction récursif pour combiner les éléments successifs dont la concaténation a le coût le plus faible jusqu'à il n'existe qu'un élément .

-3.donc j'ecrit une fonction minimiser pour combiner les éléments constamment

-4.pour minimiser , il faut utiliser une fonction concat, pour se distinguer du concat dans question 1, on l'appel concat2.

-5.dans fonction concat2,la difficulté est comment je peut conserver les éléments déjà parcouru .

Je donne en argument un liste l2 pour les conserve, si l'élément courant n'est pas ce que je veux , ajouter le dans l2, si l'élément courant est exactement ce que je veut , remplace le par la concaténation .

-6.comme je doit parcourir cette liste et trouver l'élément qui permet de la concaténation avec son successeur a le coût le plus faible, j'ai besoin d'une fonction qui décrit la règle , on l'appel min2.

min2 donne le coût le plus faible, en utilisant fonction cost et min .

-7.après j'ai sait le coût le plus faible par fonction min2, j'besoin de fonction min_retrer pour obtenir ce élément .

min_retirer calcule le coût de concaténation et compare avec le résultat de min2, si on trouve ce éléments ,retour le , on a besoin d'aussi unne fonction pour obtenir son successeur ,on l'appel succ.

jusqu'à là , on a toutes les fonctions nécessaires , l'exécution de cette question est dans la fin du rapport .

2.8 Question 8. les fonctions pour évaluer

Proposer une fonction qui calcule les gains (ou les pertes) en coût de la fonction balance sur un grand nombre d'arbres générés aléatoirement. La fonction peut, entre autre, renvoyer le min, le max, la moyenne et la valeur médiane.

C ette question permet d'obtenir les gains (ou les pertes) en coût de la fonction balance sur un grand nombre d'arbres générés aléatoirement. On peut évaluer l'algorithme précédent par le min, le max, la moyenne et la valeur médiane.

Donc on a besoin de fonction pertes pour calculer la perte du coût après fonction balance, une fonction gene_arb pour obtenir un grand nombre d'arbres générés aléatoirement, aussi des fonction pour calculer le min, le max, la moyenne et la valeur médiane.

A la fin fonction résultat donne une liste de type int :le premier élément :le plus petit perte ,2ème : le plus grand perte ,3ème: la moyenne ,4ème: la médiane .

Dans cette question ,la difficulté est comment obtenir la valeur médiane

Pour obtenir la valeur médiane , il faut connaître son index. on calcule le nombre d' éléments dans la liste. Si il est impair ,l'index est longueur -1)/2 on retour (0,longueur -1)/2).Sinon, l'index est (longueur /2),on retour (1,longueur -1)/2).

Après d'obtenir l'index, on utilise fonction medi_pertes distingue les deux situation et donne la valeur médiane .

3 Conlusion

3.1 Les limits

1. Il ne fonctionne pas quand'on donne un nombre très grand dan la fonction gene_abr.

2. Dans la fonction medi_pertes, la valeur médiane a possibilité de n'est pas très précise car le type est int.

3.2 Conclusion du travail et amélioration

Dans ce projet, je pense il faut avoir un certain ordre pour combiner les arbre . Car une phrase sans ordre n'a pas de sens , donc on peux écrit un fonction plus précise mais pas combiner simplement . cette fonction prend en argument une liste de string et retour un arbre correspondant , on ajout les noeud en même manière ,par exemple infixe.

On peut utiliser arbre binaire optimal dans plusieurs domaine , par exemple codage de Huffman.

Dans mes travail , je pense je doit baisser le nombre de fonction , faire les code plus simple .

Je pense mon fonction pour générer un grand nombre d'arbres peut être amélioré , c'est très coûteux .

J'ai accompli tous les questions , et j'ai connu mieux ce cours , c'est très intéressant parce que je peut utiliser un peu de code pour réaliser une question complexe, et Ocaml respect beaucoup de type , ça me faire connaître plus de structure de données . En plus , Ocaml utilise beaucoup de récurrences, il nous demande d'avoir des logiques plus claire, il est un outil très bien pour apprendre l'algorithme .

4 Les fonctions essentielles

```
let concat abr1 abr2=let i=ran_int2 ()
    in if (i=0) then Noeud((0,"node"),abr1,abr2)
    else Noeud((0,"node"),abr2,abr1);;
let x=word "a";;  let y=word "b";;
let z=word "c";;   let m=word "d";;
let a=concat x y;;
let b=concat z a;;
let c=concat m b;;
```

```
  c;;
- : string_builder =
Noeud ((0, "node"), Noeud ((1, "d"), Vide, Vide),
 Noeud ((0, "node"),
  Noeud ((0, "node"), Noeud ((1, "a"), Vide, Vide),
   Noeud ((1, "b"), Vide, Vide)),
  Noeud ((1, "c"), Vide, Vide)))
#
```

```
let rec to_list abr=match abr with
Vide->[]
|Noeud((a,b),fg,fd)->if((fg=Vide)&&(fd=Vide)) then to_list fg @ b ::to_list fg
else (to_list fg )@(to_list fd);;

let ltostr str=String.concat "" str;;

let char_at abr i=String.get (ltostr (to_list abr)) (i-1);;


char_at c 3;;
```

```
# - : char = 'b'
# c;;
- : string_builder =
Noeud ((0, "node"), Noeud ((1, "d"), Vide, Vide),
 Noeud ((0, "node"),
  Noeud ((0, "node"), Noeud ((1, "a"), Vide, Vide),
   Noeud ((1, "b"), Vide, Vide)),
  Noeud ((1, "c"), Vide, Vide)))
# char_at c 3;;
- : char = 'b'
#
```

```
let rec random_string i abr=
        let abr=concat abr (word (ltostr (ran_string [] 0 (ran_int ()))))
        in
        if (profondeur(niveau abr 0)=i) then abr else random_string i abr;;


let f =random_string 11 Vide;;
```

```
  f;;
- : string_builder =
Noeud ((0, "node"), Noeud ((6, "aymdno"), Vide, Vide),
 Noeud ((0, "node"),
  Noeud ((0, "node"),
   Noeud ((0, "node"), Noeud ((1, "y"), Vide, Vide),
    Noeud ((0, "node"), Noeud ((1, "l"), Vide, Vide),
     Noeud ((0, "node"), Noeud ((6, "lepbrj"), Vide, Vide),
      Noeud ((0, "node"),
       Noeud ((0, "node"), Noeud ((8, "sgzwfmvz"), Vide, Vide),
        Noeud ((0, "node"),
         Noeud ((0, "node"), Vide, Noeud ((5, "athgb"), Vide, Vide)),
         Noeud ((1, "j"), Vide, Vide))),
       Noeud ((10, "xgghgyrjyv"), Vide, Vide))))),
  Noeud ((1, "a"), Vide, Vide)),
 Noeud ((9, "iukhrxiag"), Vide, Vide)))
```

```
let rec transformer l= match l with
|[]->[]
|x::q->((word x),String.length x)::transformer q;;

let rec succ e l=match l with
|(a1,l1)::(a2,l2)::q->if ((a1,l1)=e) then a2
else succ e ((a2,l2)::q)
```

```
|_->Vide;;


let rec min2 l=match l with
|[]->0
|e::[]->1000000
|(abr,len)::reste->min (cost(concat abr (succ (abr,len) l))) ( ( min2 reste ));;


let rec min_retirer l=match l with
|[]->Vide
|(abr,len)::reste-> if ((cost(concat abr (succ (abr,len) l)))=min2 l) then fst (abr,le
else  min_retirer reste;;



let rec concat2 a l l2=if((List.length l)=1) then l else match l with
|[]->[]
|[e]->l
|(abr1,len1)::(abr2,len2)::q->
if(abr1=a) then  l2@((concat abr1 abr2),(len1+len2))::q
else (concat2 a ((abr2,len2)::q) (l2@[(abr1,len1)]));;


let rec minimiser l=match l with
|[]->[]
|[e]->[e]
|_->if (List.length l=1)then l
else
minimiser (concat2 (min_retirer l) l []);;


let balance abr=
    fst (prendre (minimiser (transformer (list_of_string abr))));;


balance f;;
```

```
# balance f;;
- : string_builder =
Noeud ((0, "node"),
 Noeud ((0, "node"), Noeud ((10, "xgghgyrjyv"), Vide, Vide),
  Noeud ((0, "node"), Noeud ((9, "iukhrxiag"), Vide, Vide),
   Noeud ((1, "a"), Vide, Vide))),
 Noeud ((0, "node"),
  Noeud ((0, "node"),
   Noeud ((0, "node"), Noeud ((6, "aymdno"), Vide, Vide),
    Noeud ((0, "node"), Noeud ((1, "y"), Vide, Vide),
     Noeud ((1, "l"), Vide, Vide))),
   Noeud ((6, "lepbrj"), Vide, Vide)),
  Noeud ((0, "node"),
   Noeud ((0, "node"), Noeud ((5, "athgb"), Vide, Vide),
    Noeud ((1, "j"), Vide, Vide)),
   Noeud ((8, "sgzwfmvz"), Vide, Vide))))
#
```

```
let pertes abr=(cost abr)-(cost (niveau(balance abr) 0));;

let rec gene_abrs ini i l=
                if(ini=i)then l
        else  gene_abrs (ini+1) i ((pertes (random_string (ran_int()) Vide))::l);;

let resultat l=
    [(min_pertes l);(max_pertes l);(moyenne_pertes l);(medi_pertes l (mediane l))];;


let l1=gene_abr 0 6 [];;
let l2=gene_abr 0 5 [];;
resultat l1;;
resultat l2;;
```