3.4 / 20

Numéro de place 5 0 2 4 7 1

Numéro d'inscription 5 3 8 9 9

Signature

Nom XHOFFRAY

Prénom NILS

CONCOURS CENTRALE·SUPÉLEC

Épreuve Informatique

Ne rien porter sur cette feuille avant d'avoir complètement rempli l'entête

Feuille 01 / 02

II- Représentation d'ensembles

A- avec une liste triée

11. `let rec succ_list l x = match l with`

`| [] → -1`

`| a::[] → -1`

`| a::b::q → if x=a then b`

`else succ_list b::q x ;;`

Si on appelle n la longueur de la liste l dans le pire des cas la complexité est en $O(n)$.

B- avec un vecteur trié

12. - pour déterminer le maximum il suffit d'effectuer des comparaisons en gardant en mémoire le plus grand des éléments de chaque comparaison, dans le pire cas en $O(n)$

- pour tester l'appartenance idem que pour le maximum mais on ne stocke aucune valeur également en $O(n)$ dans le pire cas

- pour ajouter un élément x on effectue des comparaisons jusqu'à trouver la place de x dans le vecteur puis on l'affecte. Complexité en $O(n)$ dans le pire cas.

13. `let floor x = i = ref 0.n`

`while !i+1 =< x do incr i end ; !i ;;`

pire chose

3.4 / 20

Ne rien écrire dans la partie barrée

```
let succ_veet t x = let d = ref (floor (n-1)/2) in
    let a = ref 1 in
    let b = ref n in
    if t.(n) = x then -1
    else while t.(d) <> x do
        if t.(d) < x then a := (!d)
        else b := !d
        n := floor (b-a)/2 end;
    t.(d+1) ;;
```

14- Dans le pire des cas on effectuera $\frac{n}{2}$ comparaisons donc la complexité est en $O\left(\frac{n}{2}\right)$

15-
```
let union_veet t1 t2 =
    let n = Array.length t1 + Array.length t2 in
    let t = Array.make n in
    t.(0) <- t1.(0) + t2.(0)
    let i = ref 1 in
    let j = ref 1 in
    let k = ref 1 in
    while !i < Array.length t1 && !j < Array.length t2 && !k < n do
        if t1(!i) < t2(!j) then
            t.(!k) <- t1.(!i)
            incr i
            incr k
        else t.(!k) <- t2.(!j)
            incr j
            incr k end;
    t;;
```

C- avec un arbre binaire de recherche

16- Let rec min_arb a = let m = ref 0 in
match a with
| Nil → 0
| (n, g, d) → ~~if n < !m then m := n ;~~
~~if min_arb g < min_arb d then min_arb g~~
~~else min_arb d ;;~~
~~if !m = 0 then m := n~~
if !m = 0 then m := n
if n < !m then m := n
if min_arb g < min_arb d then min_arb g
else min_arb d ;;

17- Let rec partitionne_arb a x = let bool = ref false in
match a with
Nil → (false, Nil, Nil)
| (n, g, d) → if n < x then (bool, partitionne_arb (n, g, Nil) x, partitionne_arb d x)
if n > x then (bool, partitionne_arb g x, partitionne_arb (n, Nil, d) x)
if n = x then bool := true (bool, partitionne_arb g x, partitionne_arb d x);;

18- Let rec inserer_arb a x =
match a with
Nil → (x, Nil, Nil)
| (n, g, d) → if x < n then (n, inserer g x, d)
~~if x~~ if x > n then (n, g, inserer d x)
if x = n then (x, g, d) ;;

III - Représentation par arbres binaires complets

20- pour un arbre de hauteur $p \in \mathbb{N}$ on sait déjà que les numéros appartiennent à $[\![2^k ; 2^{k+1}-1]\!]$

par ici dans l'arbre dans il existe $k \in [\![0, p]\!]$ tel que $i \in [\![2^k, 2^{k+1}-1]\!]$

et le sous arbre de racine le numéro i aura 2^{p-k} feuille.

21- pour $i = 1$ dans le successeur gauche est 2 soit $2\times i$ et le droit $= 2\times i+1$

supposons trouvé $i \in [\![2, 2^p-1]\!]$ tel que son successeur droit ait pour indice $2i+1$, son successeur gauche $2i$ et son père $\lfloor \frac{i}{2} \rfloor$ la même entrée de $\frac{i}{2}$.

il existe $k \in [\![1, p]\!]$ et $t \in [\![0, 2^k-1]\!]$ tel que $i = 2^k + t$

ainsi $it+1 = 2^k + t + 1$ le successeur gauche de i vaut $2i$ et celui de $it+1$ est éloigné de 2 par rapport à celui de i donc de valeur $2it+2 = 2(it+1)$

obtenu par le successeur droit de $it+1$ qui vaut $2it+1+2 = 2it+3 = 2(it+1)+1$

de plus si i est pair donc i et $it+1$ ont le même père et $\lfloor \frac{it+1}{2} \rfloor = \lfloor \frac{i}{2} \rfloor$

si i impair donc le père de $it+1$ a pour indice $\lfloor \frac{i}{2} \rfloor + 1$ or i impair donc il existe $k \in \mathbb{N}$ tq $i = 2k+1$ et $\lfloor \frac{it+1}{2} \rfloor = \lfloor \frac{2kt+2}{2} \rfloor = \lfloor \frac{2kt}{2} \rfloor + 1 = \lfloor \frac{i}{2} \rfloor + 1$

car $\lfloor \frac{2k}{2} \rfloor = \lfloor \frac{2k+1}{2} \rfloor$.

donc pour tout ~~i~~ si ils existent, le successeur gauche noté $2i$, le droit $2i+1$ et le père $\lfloor \frac{i}{2} \rfloor$. $i \in [\![0, 2^p-1]\!]$

22- ~~Let operator E =~~ Let puissance_deux n = i = ref 1 in

if n = 1 then 0

else while 2 * (!i + 1) <= n then incr i end;

!i ;;

Let operator E = Let p = puissance_deux (Array.length E) in

bool = ref false in

for k = 0 to 2 ** (p-1) - 1 ~~do~~ begin

if 2 ** p + k - 2 ** p = x && E.(2 ** p + k) = true

~~then true else~~ then bool := true break ;

end ;

!bool ;;

La complexité est en $O(2^{p-1} \cdot 2^{p-1} + p)$

3.4 / 20

Numéro de place 5 0 2 4 7 1

Numéro d'inscription 5 3 8 9 9

Signature

Nom X H O F F R A 4

Prénom W I L S

CONCOURS CENTRALE·SUPÉLEC

Épreuve Informatique

Ne rien porter sur cette feuille avant d'avoir complètement rempli l'entête

Feuille 0 2 / 5 2

III - Représentation par arbres binaires complets

Q3- let rec fabrique l v = let E = Array.make v-1 false in
match l with
[] → failwith "la liste est vide"
| a::q → for k=0 to v/2 -1 begin
if (v/2+k-v) = a then E.(v/2+k) ← true ~~end~~
~~if i = neg v/2+k in~~ let i = neg v/2+k in break;
while !i =! 1 do
i := fpere (!i/2)
E.(!i) ← true end;
fabrique q v ;;

Q4- let insere E k = let v = Array.length E in
for j = 0 to v/2 - 1 begin
if v/2+j-v = k then E.(v/2+j) ← true ~~end~~;
~~if~~ let i = neg v/2+j in break;
while ! i =! 1 do
i := fpere (!i/2)
E.(!i) ← true end;
//

3.4 / 20

Ne rien écrire dans la partie barrée

```
3e- let rec cardinal E =
        let i = minlocal E in ;
        1 + cardinal (supprime (E successeur(i))) ;;
```

I- Lemmes et autres

1- $a^* \triangleright b^* = \{ v \circ w \mid (v,v,w) \in (X^*)^3 , w \in b^* , v \in a^* \} = b^* a^* + a^* b^*$

$(ba)^* \triangleright (ab)^* = \{ v \circ w \mid (v,v,w) \in (X^*)^3 , w \in (ab)^* , v \in (ba)^* \} = (ab)^*$

et $a^* \triangleright \{ a^p b a^q \mid (p,q) \in \mathbb{N}^2 , p \leq q \} = a^*$

2- On a : $L \triangleright (K_1 | K_2) = (L \triangleright K_1 \mid L \triangleright K_2)$ et $L \triangleright (K_1 . K_2) = L \triangleright K_1 . L \triangleright K_2$

~~Par b_1 on trouve $\delta(b_1) = \varepsilon^* b^* + b^* a^*$~~