TD2 – Séquences

1. Problème de Dijkstra

Problème
On dispose de cailloux rouges, blancs et bleus alignés dans un ordre quelconque.
Par échanges successifs de cailloux, on doit les trier par couleur (bleu, blanc puis rouge).
De plus, on ne doit tester qu'une seule fois la couleur de chaque caillou.

Analyse
On a une suite de cailloux. Les opérations élémentaires sont tester la couleur d'un caillou et échanger deux cailloux.

Méthode
Définir

- une situation intermédiaire s(i),
- le passage de s(i) à s(i+1) qui doit constituer un progrès vers la solution finale,
- une situation initiale s(0) vérifiant les hypothèses caractérisant s(i).

4 Situations intermédiaires intéressantes

1. bleus blancs rouges non triés
2. bleus blancs non triés rouges
3. bleus non triés blancs rouges
4. non triés bleus blancs rouges

La 2 et la 3 sont équivalentes et meilleures que la 1 et la 4 car les bleus et les rouges sont déjà à la bonne place. Prenons donc la 2.

de 1 à i	bleus
de i+1 à j	blancs
de j+1 à k	non triés
de k+1 à n	rouges

Où peut-on placer le caillou de rang j+1.
Si la couleur du caillou en j+1 est

- bleue : échange avec le premier caillou blanc, puis mettre à jour les indices
- blanche : il reste la où il est, puis mettre à jour les indices
- rouge : échange avec le dernier des non triés, puis mettre à jour les indices

Existe-t-il une situation initiale compatible ? oui, il suffit de remarquer qu'au début tous les cailloux sont dans la classe des non triés : j+1 = 1 ; j=0, k=n et i=0

Algo

```
Début
{lecture des données}
i←borneInf(s)-1
j←borneInf(s)-1
k←borneSup(s)
Répeter
      Selon Ième(s,j+1)
            bleu : échanger(Ième(s,j+1),Ième(s,i+1))
                   i←i+1
                   j←j+1
            blanc : j←j+1
            rouge: échanger(Ième(s,j+1),Ième(s,k))
                   k←k-1
      FinSelon
Jusqu'à k=j
Fin
```

2. Recherche dichotomique

```
Programme dichotomie
Const max = 10
Type tab = tableau [1..max] d'entier
Var   t : tab
      m, a, b, x : entier
      trouvé : booléen
Début
lire(x)
trouvé←faux
a←1
b←max
Répéter
      m←(a+b) div2
      Si x<t[m]
            Alors b←m-1
            Sinon Si x>t[m]
                        Alors A←m+1
                        Sinon trouvé←vrai
                  FinSi
      FinSi
Jusqu'à (trouvé ou a>b)
Si trouvé
      Alors écrire(m)
      Sinon écrire('non trouvé')
FinSi
Fin
```

3. Fusion de deux tableaux triés

```
Programme Fusion
Const Max = 10
Type  tab = tableau [1..Max] d'entier
      tabfusion = tableau [1..2Max] d'entier
Var   t1, t2 : tab
      t3 : tabfusion

            Procédure fusion (E t, u : tab, m, n :entier ; S v : tabfusion)
            Var i, j, k : entier
            Début
            i←1
            j←1
            k←1
            TantQue (i<=m) et (j<=n) Faire
                  Si t[i]<=u[j]
                        Alors v[k]←t[i]
                              i←i+1
                              k←k+1
                        Sinon v[k]←v[j]
                              j←j+1
                              k←k+1
                  FinSi
            FinTantQue
            Si i<=m
                  Alors Pour j←i à m inc +1 Faire
                              v[k]←t[j]
                              k←k+1
                        FinPour
                  Sinon Pour i←j à n inc +1 Faire
                              v[k]←t[i]
                              k←k+1
                        FinPour
            FinSi
            Fin
```

```
Début
(* Lecture de t1 ayant m éléments et t2 ayant n éléments*)
Fusion(t1,t2,m,n,t3)
(* Affichage de t3*)
Fin
```

4. Produit de matrices

$c_{ij} = \Sigma a_{ik} b_{kj}$

```
Programme prod-matrice
Var   a, b, c : SeqDouble d'entiers
      i, j, s : entier
Début
Pour i← borneInf(a) à borneSup(a) Inc +1 Faire
      Pour j← borneInf(a) à borneSup(a) Inc +1 Faire
            s←0
            Pour k←1 à borneSup(a) Inc +1 Faire
                  s←s+ième(a,i,k)*ième(b,k,j)
            FinPour
            ChangerIème(c,i,j,s)
      FinPour
FinPour
Fin
```

Adaptation au matrices de complexes

Sorte : Comp
Utilise : Réel
Opérations :
- Créer (Réel, Réel) → Comp
- R (Comp) → Réel
- I (Comp) → Réel
- Somme-comp (Comp, Comp) → Comp
- Prod-comp (Comp, Comp) → Comp

```
Programme produit-matrice-complexe
Const max = 20
Type comp = Enregistrement
               r, i : réel
            FinEnregistrement
      mat-comp = tableau [1..max,1..max] de comp
Var   i, j : entier
      a, b, c : mat-comp

      Fonction somme-comp(a, b : comp) : comp
      Var c : comp
      Début
      c.r←a.r+b.r
      c.i←a.i+b.i
      retourner(c)
      Fin

      Fonction prod-comp(a, b : comp) : comp
      Var c : comp
      Début
      c.r←(a.r*b.r)-(a.i*b.i)
      c.i←(a.r*b.i)+(a.i*b.r)
      retourner(c)
      Fin
```

```
      Procédure prod-mat-comp(E a, b : mat-comp ; S c : mat-comp)
      Var i, j, k : entier
          s : comp
      Début
      Pour i←1 à max Inc +1 Faire
            Pour j←1 à max Inc +1 Faire
                  s.r←0
                  s.i←0
                  Pour k←1 à max Inc +1 Faire
                        s←(som-comp(s,prod-comp(a[i,k],b[k,j]))
                  FinPour
                  c[i,j]←s
            FinPour
      FinPour
      Fin

Début
Pour i←1 à max Inc +1 Faire
      Pour j←1 à max Inc +1 Faire
            lire(a[i,j].r)
            lire(a[i,j].i)
            lire(b[i,j].r)
            lire(b[i,j].i)
      Finpour
FinPour
prod-mat-comp(a,b,c)
Pour i←1 à max Inc +1 Faire
      Pour j←1 à max Inc +1 Faire
            écrire(c[i,j].r)
            écrire(c[i,j].i)
      Finpour
FinPour
Fin
```