**EMP**

**Algorithmique &Structures de données**

**Corrigé // Examen 2  // 2013 2014**

 **SOIT**

 **F UN FICHIER DE ( ENTIER , VECTEUR ( 20 ) , ENTIER ) ENTETE ( ENTIER ) BUFFER Buf , Buf2 ;**

 **{ Un bloc du fichier contient**

 **Premier champ : nombre d’éléments dans le vecteur**

 **Deuxième champ : tableau de 20 entiers**

 **Troisième champ : Adresse du bloc suivant**

 **}**

 **{L’entete contient un entier qui désigne la tete du fichier (liste de blocs) }**

 **A UN ARB DE TABLEAU ( 5 ) ;**

 **Inserer UNE ACTION ;**

 **Egal UNE FONCTION ( BOOLEEN ) ;**

 **Remplir UNE ACTION ;ur**

 **Requete UNE ACTION ;**

 **Poids UNE FONCTION ( ENTIER ) ;**

 **Constr UNE ACTION ;**

 **Lister\_arbre UNE ACTION ;**

 **Lister\_fichier UNE ACTION ;**

 **Aerer UNE ACTION ;**

 **V UN TABLEAU ( 5 ) ;**

 **I UN ENTIER ;**

 **Ind\_buf : ENTIER ;**

 **L : ENTIER ;**

 **Next : ENTIER ;**

 **Nb : ENTIER ;**

 **Nb\_vecteurs : ENTIER ;**

 **Taille\_vecteur : ENTIER ;**

 **Taille\_vecteur\_buf : ENTIER ;**

 **DEBUT**

 **{Construire un arbre de recherche binaire : principal}**

 **Nb\_vecteurs := 30 ;**

 **Taille\_vecteur := 5 ;**

 **Taille\_vecteur\_buf := 20 ;**

 **A := NIL ;**

 **POUR I := 1 , Nb\_vecteurs**

 **APPEL Remplir ( V ) ;**

 **APPEL Inserer ( A , V )**

 **FPOUR ;**

 **ECRIRE ( 'Parcours de l’’arbre' ) ;**

 **APPEL Lister\_arbre ( A ) ;**

 **ECRIRE ( 'Requete ...' ) ;**

 **APPEL Requete ( A , 500 , 3000 ) ;**

 **ECRIRE ( 'Construction...' ) ;**

 **OUVRIR ( F , 'lof.txt' , 'N' ) ;**

 **Ind\_buf := 0 ;**

 **Nb := 0 ;**

 **L := ALLOC\_BLOC ( F ) ;**

 **AFF\_ENTETE ( F , 1 , L ) ;**

 **APPEL Constr ( A ) ;**

 **FERMER ( F ) ;**

 **ECRIRE ( 'Le fichier : ' ) ;**

 **APPEL Lister\_fichier ;**

 **APPEL Aerer ( 3 ) ;**

 **ECRIRE ( 'Le fichier aeré: ' ) ;**

 **APPEL Lister\_fichier ;**

 **FIN**

**{Construire un arbre de recherche binaire : Insérer}**

 **ACTION Inserer ( A , V )**

 **SOIT**

 **A , P , Prec , R DES ARB DE TABLEAU ( 5 ) ;**

 **V UN TABLEAU ( 5 ) ;**

 **Trouv UN BOOLEEN ;**

 **DEBUT**

 **P := A ;**

 **Trouv := FAUX ;**

 **Prec := NIL ;**

 **TQ ( P <> NIL ) ET NON Trouv**

 **SI ( Poids ( V ) = Poids ( INFO ( P ) ) ) ET NON Egal ( V , INFO ( P ) )**

 **Trouv := VRAI**

 **SINON**

 **Prec := P ;**

 **SI Poids ( V ) < Poids ( INFO ( P ) )**

 **P := FG ( P )**

 **SINON**

 **P := FD ( P )**

 **FSI**

 **FSI**

 **FTQ ;**

 **SI NON Trouv**

 **CREERNOEUD ( R ) ;**

 **AFF\_INFO ( R , V ) ;**

 **SI Prec = NIL**

 **A := R**

 **SINON**

 **SI Poids ( V ) < Poids ( INFO ( Prec ) )**

 **AFF\_FG ( Prec , R )**

 **SINON**

 **AFF\_FD ( Prec , R )**

 **FSI**

 **FSI**

 **FSI**

 **FIN**

**{Déterminer dans l’arbre tous les tableaux de poids compris entre deux poids P1 et P2 donnés}**

 **ACTION Requete ( A , P1 , P2 ) ;**

 **SOIT**

 **P UNE PILE DE ARB DE TABLEAU ( 5 ) ;**

 **Pt , A : ARB DE TABLEAU ( 5 ) ;**

 **Possible : BOOLEEN ;**

 **P1 , P2 : ENTIERS ;**

 **DEBUT**

 **CREERPILE ( P ) ;**

 **Pt := A ;**

 **Possible := VRAI ;**

 **TQ Possible**

 **TQ Pt <> NIL :**

 **EMPILER ( P , Pt ) ;**

 **Pt := FG ( Pt )**

 **FTQ ;**

 **SI NON PILEVIDE ( P )**

 **DEPILER ( P , Pt ) ;**

 **SI ( Poids ( INFO ( Pt ) ) >= P1 ) ET ( Poids ( INFO ( Pt ) ) <= P2 )**

 **ECRIRE ( INFO ( Pt ) )**

 **FSI ;**

 **Pt := FD ( Pt )**

 **SINON**

 **Possible := FAUX**

 **FSI**

 **FTQ**

 **FIN**

 ***FONCTION Poids ( T ) : ENTIER***

 ***SOIT***

 ***T UN TABLEAU ( 5 ) ;***

 ***I UN ENTIER ;***

 ***DEBUT***

 ***Poids := 0 ;***

 ***POUR I := 1 , Taille\_vecteur***

 ***Poids := Poids + ELEMENT ( T [ I ] )***

 ***FPOUR***

 ***FIN***

 ***FONCTION Egal ( T1 , T2 ) : BOOLEEN ;***

 ***SOIT***

 ***T1 , T2 DES TABLEAUX ( 5 ) ;***

 ***I UN ENTIER ;***

 ***Trouv : BOOLEEN ;***

 ***DEBUT***

 ***Trouv := FAUX ;***

 ***I := 1 ;***

 ***TQ ( I <= Taille\_vecteur ) ET NON Trouv***

 ***SI ELEMENT ( T1 [ I ] ) <> ELEMENT ( T2 [ I ] )***

 ***Trouv := VRAI***

 ***SINON***

 ***I := I + 1***

 ***FSI***

 ***FTQ ;***

 ***Egal := NON Trouv ;***

 ***FIN***

 **ACTION Remplir ( T )**

 **SOIT**

 **T UN TABLEAU ( 5 ) ;**

 **I : ENTIER ;**

 **DEBUT**

 **POUR I := 1 , Taille\_vecteur**

 **AFF\_ELEMENT ( T [ I ] , ALEANOMBRE ( 1000 ) )**

 **FPOUR**

 **FIN**

**{lister les éléments de l’arbre ordonnés selon leurs poids}**

 **ACTION Lister\_arbre ( A )**

 **SOIT**

 **A UN ARB DE TABLEAU ( 5 ) ;**

 **DEBUT**

 **SI A <> NIL**

 **APPEL Lister\_arbre ( FG ( A ) ) ;**

 **ECRIRE ( INFO ( A ) , 'Poids=' , Poids ( INFO ( A ) ) ) ;**

 **APPEL Lister\_arbre ( FD ( A ) )**

 **FSI**

 **FIN**

**{ Construire un fichier LÔF }**

 **ACTION Constr ( A )**

 **SOIT**

 **A UN ARB DE TABLEAU ( 5 ) ;**

 **K UN ENTIER ;**

 **DEBUT**

 **SI A <> NIL**

 **APPEL Constr ( FG ( A ) ) ;**

 **Nb := Nb + Taille\_vecteur ;**

 **POUR K := 1 , Taille\_vecteur**

 **Ind\_buf := Ind\_buf + 1 ;**

 **AFF\_ELEMENT ( STRUCT ( Buf , 2 ) [ Ind\_buf ] , ELEMENT ( INFO ( A ) [ K ] ) )**

 **FPOUR ;**

 **SI ( Ind\_buf = Taille\_vecteur\_buf ) OU ( Nb = Taille\_vecteur \* Nb\_vecteurs )**

 **AFF\_STRUCT ( Buf , 1 , Ind\_buf ) ;**

 **SI Nb = Taille\_vecteur \* Nb\_vecteurs**

 **Next := - 1**

 **SINON**

 **Next := ALLOC\_BLOC ( F ) + 1 ;**

 **FSI ;**

 **AFF\_STRUCT ( Buf , 3 , Next ) ;**

 **ECRIREDIR ( F , Buf , L ) ;**

 **L := Next ;**

 **Ind\_buf := 0 ;**

 **FSI ;**

 **APPEL Constr ( FD ( A ) )**

 **FSI**

 **FIN**

**{Lister tous les éléments du fichier }**

 **ACTION Lister\_fichier ;**

 **SOIT**

 **L UN ENTIER ;**

 **DEBUT**

 **OUVRIR ( F , 'lof.txt' , 'A' ) ;**

 **L := ENTETE ( F , 1 ) ;**

 **TQ L <> - 1**

 **LIREDIR ( F , Buf , L ) ;**

 **ECRIRE ( 'Bloc = ' , L ) ;**

 **ECRIRE ( Buf ) ;**

 **L := STRUCT ( Buf , 3 ) ;**

 **FTQ ;**

 **FERMER ( F )**

 **FIN**

**{Algorithme d’aération des blocs M et M+1}**

 **ACTION Aerer ( M ) ;**

 **SOIT**

 **I , K , N , M : ENTIER ;**

 **Suivant\_m : ENTIER ;**

 **DEBUT**

 **OUVRIR ( F , 'lof.txt' , 'A' ) ;**

 **N := ALLOC\_BLOC ( F ) ;**

 **LIREDIR ( F , Buf , M ) ;**

 **Suivant\_m := STRUCT ( Buf , 3 ) ;**

 **K := 0 ;**

 **POUR I := ( 2 \* Taille\_vecteur\_buf ) / 3 + 1 , Taille\_vecteur\_buf**

 **K := K + 1 ;**

 **AFF\_ELEMENT ( STRUCT ( Buf2 , 2 ) [ K ] , ELEMENT ( STRUCT ( Buf , 2 ) [ I ] ) )**

 **FPOUR ;**

 **AFF\_STRUCT ( Buf2 , 1 , Taille\_vecteur\_buf - ( 2 \* Taille\_vecteur\_buf ) / 3 ) ;**

 **AFF\_STRUCT ( Buf , 1 , ( 2 \* Taille\_vecteur\_buf ) / 3 ) ;**

 **AFF\_STRUCT ( Buf , 3 , N ) ;**

 **ECRIREDIR ( F , Buf , M ) ;**

 **LIREDIR ( F , Buf , Suivant\_m ) ;**

 **POUR I := ( 2 \* Taille\_vecteur\_buf ) / 3 + 1 , Taille\_vecteur\_buf**

 **K := K + 1 ;**

 **AFF\_ELEMENT ( STRUCT ( Buf2 , 2 ) [ K ] , ELEMENT ( STRUCT ( Buf , 2 ) [ I ] ) )**

 **FPOUR ;**

 **AFF\_STRUCT ( Buf2 , 1 , 2 \* ( Taille\_vecteur\_buf - ( 2 \* Taille\_vecteur\_buf ) / 3 ) ) ;**

 **AFF\_STRUCT ( Buf , 1 , ( 2 \* Taille\_vecteur\_buf ) / 3 ) ;**

 **ECRIREDIR ( F , Buf , Suivant\_m ) ;**

 **AFF\_STRUCT ( Buf2 , 1 , K ) ;**

 **AFF\_STRUCT ( Buf2 , 3 , Suivant\_m ) ;**

 **ECRIREDIR ( F , Buf2 , N ) ;**

 **FERMER ( F ) ;**

 **FIN**

**Trace :**

**Construction…**

**Parcours de l’arbre fichier**

**510 86 125 60 408 Poids= 1189**

**157 601 210 145 179 Poids= 1292**

**262 546 596 129 124 Poids= 1657**

**135 806 629 69 79 Poids= 1718**

**894 69 265 10 491 Poids= 1729**

**721 106 5 186 765 Poids= 1783**

**20 570 304 808 218 Poids= 1920**

**563 464 654 87 187 Poids= 1955**

**155 582 148 889 191 Poids= 1965**

**80 708 373 857 114 Poids= 2132**

**572 40 810 520 216 Poids= 2158**

**45 929 354 726 185 Poids= 2239**

**75 631 651 31 953 Poids= 2341**

**705 172 601 447 524 Poids= 2449**

**602 267 693 454 458 Poids= 2474**

**424 486 986 62 585 Poids= 2543**

**884 392 614 693 10 Poids= 2593**

**768 373 603 259 693 Poids= 2696**

**587 622 977 83 621 Poids= 2890**

**437 791 362 658 694 Poids= 2942**

**820 419 945 434 357 Poids= 2975**

**872 155 577 778 643 Poids= 3025**

**801 515 860 71 811 Poids= 3058**

**957 833 656 208 457 Poids= 3111**

**708 216 919 709 600 Poids= 3152**

**611 711 435 735 663 Poids= 3155**

**648 920 803 1 894 Poids= 3266**

**637 953 277 718 806 Poids= 3391**

**984 678 132 814 999 Poids= 3607**

**943 937 965 830 78 Poids= 3753**

**Requête ...**

**510 86 125 60 408**

**157 601 210 145 179**

**262 546 596 129 124**

**135 806 629 69 79**

**894 69 265 10 491**

**721 106 5 186 765**

**20 570 304 808 218**

**563 464 654 87 187**

**155 582 148 889 191**

**80 708 373 857 114**

**572 40 810 520 216**

**45 929 354 726 185**

**75 631 651 31 953**

**705 172 601 447 524**

**602 267 693 454 458**

**424 486 986 62 585**

**884 392 614 693 10**

**768 373 603 259 693**

**587 622 977 83 621**

**437 791 362 658 694**

**820 419 945 434 357**

**Construction...**

**Le fichier :**

**Bloc = 1**

**20 510 86 125 60 408 157 601 210 145 179 262 546 596 129 124 135 806 629 69 79 2**

**Bloc = 2**

**20 894 69 265 10 491 721 106 5 186 765 20 570 304 808 218 563 464 654 87 187 3**

**Bloc = 3**

**20 155 582 148 889 191 80 708 373 857 114 572 40 810 520 216 45 929 354 726 185 4**

**Bloc = 4**

**20 75 631 651 31 953 705 172 601 447 524 602 267 693 454 458 424 486 986 62 585 5**

**Bloc = 5**

**20 884 392 614 693 10 768 373 603 259 693 587 622 977 83 621 437 791 362 658 694 6**

**Bloc = 6**

**20 820 419 945 434 357 872 155 577 778 643 801 515 860 71 811 957 833 656 208 457 7**

**Bloc = 7**

**20 708 216 919 709 600 611 711 435 735 663 648 920 803 1 894 637 953 277 718 806 8**

**Bloc = 8**

**10 984 678 132 814 999 943 937 965 830 78 648 920 803 1 894 637 953 277 718 806 -1**

**Le fichier aéré:**

**Bloc = 1**

**20 510 86 125 60 408 157 601 210 145 179 262 546 596 129 124 135 806 629 69 79 2**

**Bloc = 2**

**20 894 69 265 10 491 721 106 5 186 765 20 570 304 808 218 563 464 654 87 187 3**

**Bloc = 3**

**13 155 582 148 889 191 80 708 373 857 114 572 40 810 520 216 45 929 354 726 185 9**

**Bloc = 9**

**14 520 216 45 929 354 726 185 454 458 424 486 986 62 585 0 0 0 0 0 0 4**

**Bloc = 4**

**13 75 631 651 31 953 705 172 601 447 524 602 267 693 454 458 424 486 986 62 585 5**

**Bloc = 5**

**20 884 392 614 693 10 768 373 603 259 693 587 622 977 83 621 437 791 362 658 694 6**

**Bloc = 6**

**20 820 419 945 434 357 872 155 577 778 643 801 515 860 71 811 957 833 656 208 457 7**

**Bloc = 7**

**20 708 216 919 709 600 611 711 435 735 663 648 920 803 1 894 637 953 277 718 806 8**

**Bloc = 8**

**10 984 678 132 814 999 943 937 965 830 78 648 920 803 1 894 637 953 277 718 806 -1**

**Machines abstraites**

* **Vecteurs**

**Définition : Un vecteur de dimension N peut être vu comme une application de l’ensemble I= [1, Max1]X [1, Max2].....X[1, MaxN] vers un ensemble de valeurs. Un vecteur est donc un ensemble de couples (indice1, a), (indice2, b), ....., (indiceN, e) dont les premiers éléments sont appelés les indices et appartiennent à l’ensemble I. Une machine simple sur les vecteurs consiste à agir uniquement sur un élément, c’est à dire le consulter ou le modifier.**

**Opérations :**

|  |
| --- |
| **ELEMENT ( T [i, j, ...] ) : Accès à l'élément T[i, j, ...] du vecteur T.****AFF\_ELEMENT ( T [i, j, ...], Val ) : Affecter à l'élément T[i, j, ...] la valeur Val.** |

* **Arbres de recherche binaire**

**Définition : Un arbre de recherche binaire permet de représenter un ensemble de données muni d’une relation d’ordre. Toutes les données se trouvant dans le sous arbre gauche de tout noeud avec l’information x sont inférieures à x, toutes les données se trouvant dans le sous arbre droit de tout noeud avec l’information x sont supérieures à x. Comme c’est une structure de données dynamique (composé d’un ensemble de noeuds formant un graphe dans lequel tout noeud a au plus un prédécesseur et au plus deux successeurs ) une machine consiste à fournir des opérations pour faire l’allocation dynamique, des opérations pour remplir un noeud et des opérations pour consulter un noeud. Intuitivement, un noeud est composé de 3 champs : l’information et deux champs adresses fournissant les fils gauche et droit. . La machine agit uniquement au niveau du noeud.**

**Opérations :**

|  |
| --- |
| **CREERNOEUD ( Val ) : Créer un noeud avec l'information Val et retourne l'adresse du noeud.** **Les autres champs sont à NIL.****LIBERERNOEUD ( P ) : Libèrer le noeud d'adresse P.****FG ( P ) : Accès au champ Fils gauche du noeud référencé par P.****FD ( P ) : Accès au champ Fils droit du noeud référencé par P.****PERE ( P ) : Accès au champ Père du noeud référencé par P.****INFO ( P ) : Accès au champ Info du noeud référencé par P.****AFF\_FG ( P, Q ) : Affecter au champ Fils gauche du noeud référencé par p, l'adresse Q****AFF\_FD ( P, Q ) : Affecter au champ Fils droit du noeud référencé par p, l'adresse Q****AFF\_PERE ( P, Q ) : Affecter au champ Père du noeud référencé par p, l'adresse Q****AFF\_INFO( P, Val ) : Affecter au champ Info du noeud référencé par p, la valeur Val** |

* **Structures**

**Définition : une structure est un ensemble d’éléments hétérogènes. Un élément d’une structure peut être un scalaire ou un vecteur à une dimension de scalaires. Une machine simple sur les structures consiste à récupérer le i-ème champ ou d’affecter une valeur dans son i-ème champ.**

**Opérations**

|  |
| --- |
| **STRUCT(S, i) : accès au i-ème champs.****AFF\_STRUCT(S, i, Val) : affecter la valeur val au i-ème champ.** |

* **Fichiers**

**Définition : un fichier est ensemble de structures, généralement rangées sur le disque. Les structures peuvent être des articles(niveau utilisateur) ou des blocs(niveau concepteur). Le fichier renferme une structure particulière (entête) nécessaire pour la conception de structures de fichiers. Une machine abstraite sur les fichiers consiste à fournir des opérations permettant de faire l’accès séquentiel et l’accès direct.**

**Opérations**

|  |
| --- |
| **OUVRIR (Flogique, Fphysique, Mode) : ouvrir le fichier logique et l’associer au fichier physique** **en précisant si le fichier est nouveau ('N') ou ancien ('A').****FERMER(Flogique) : fermer le fichier.****LIRESEQ (Flogique, V) : Lecture dans la variable V l'article se trouvant à la position courante.****ECRIRESEQ (Flogique, V) : Ecriture de l'article V à la position courante.****LIREDIR (Flogique, V, n) : lecture du n-ième article du fichier.****ECRIREDIR (Flogique, V, n) : écriture de l'article V à la n-ième position.****RAJOUTER (Flogique, V) : rajoute un article en fin du fichier****FINFICH (Flogique) : prédicat égal à vrai si la fin du fichier est rencontrée, faux sinon.****ALLOC\_BLOC(Flogique) : position d'un bloc ( ou article ) dans laquelle on pourra écrire.****ENTETE (F, i) : récupérer la i-ème caractéristique du fichier.****AFF\_ENTETE(F, i, v) : affecter v comme i-ème caractéristique du fichier.** |