Allocation contigüe

Yanis Zatout

Octobre 2019

Allocation contigüe: Avantages

- L'accès à coût réduit aux cases mémoires d'une structure de données

Allocation contigüe: Avantages

- L'accès à coût réduit aux cases mémoires d'une structure de données
  - Seule la première case de la matrice statique est nécessaire

Allocation contigüe: Avantages

- L'accès à coût réduit aux cases mémoires d'une structure de données
  - Seule la première case de la matrice statique est nécessaire
- L'ajout et le retrait d'éléments très facilement

Allocation contigüe: Avantages

- L'accès à coût réduit aux cases mémoires d'une structure de données
  - Seule la première case de la matrice statique est nécessaire
- L'ajout et le retrait d'éléments très facilement
- Facile à détruire

Allocation contigüe: allocation statique

Lorsqu'on alloue une matrice de façon statique, l'allocation est contigüe, exemple:

Allocation contigüe: allocation statique

Lorsqu'on alloue une matrice de façon statique, l'allocation est contigüe, exemple:

```
1       int A[2][3]={ { 1,2,3},{4,5,6}};
2       for (i=0;i<rows;i++)
3           for (j=0;j<cols;j++)
4               printf("%4d␣",A[i][j]);
5           printf("\n");
6       }
```

Allocation contigüe: allocation statique

Lorsqu'on alloue une matrice de façon statique, l'allocation est contigüe, exemple:

```
1       int A[2][3]={ { 1,2,3},{4,5,6}};
2       for (i=0;i<rows;i++)
3           for (j=0;j<cols;j++)
4               printf("%4d ",A[i][j]);
5           printf("\n");
6       }
```

Sortie

```
   1    2    3
   4    5    6
```

Allocation contigüe allocation statique

De même cette, fois en parcourant la matrice avec un pointeur:

Allocation contigüe allocation statique

De même cette, fois en parcourant la matrice avec un pointeur:

```
1       pA=&(A[0][0]);
2       for (i=0;i<rows;i++){
3           for (j=0;j<cols;j++){
4               printf("%4d␣",*pA);
5               pA++; // On passe a l'adresse suivante
6       }
7       printf("\n");
8   }
```

Allocation contigüe allocation statique

De même cette, fois en parcourant la matrice avec un pointeur:

```
1       pA=&(A[0][0]);
2       for (i=0;i<rows;i++){
3           for (j=0;j<cols;j++){
4               printf("%4d ",*pA);
5               pA++; // On passe a l'adresse suivante
6       }
7       printf("\n");
8   }
```

Sortie

```
   1    2    3
   4    5    6
```

Allocation contigüe: allocation d'un tableau de taille dynamique

La mémoire allouée dynamiquement avec les fonctions calloc, malloc ou realloc est toujours contigüe:

```
1  int tailleTab = 5;
2  int * tab = (int*) malloc(tailleTab*sizeof(int));
3  int * pTab = tab;
4  for (int i = 0; i < tailleTab; i++){
5     printf("&tab[%d]=%p,␣␣pTab=%p\n", i,&tab[i],pTab);
6     pTab++;
7  }
```

Allocation contigüe: allocation d'un tableau de taille dynamique

La mémoire allouée dynamiquement avec les fonctions calloc, malloc ou realloc est toujours contigüe:

```
1  int tailleTab = 5;
2  int * tab = (int*) malloc(tailleTab*sizeof(int));
3  int * pTab = tab;
4  for (int i = 0; i < tailleTab; i++){
5     printf("&tab[%d]=%p,  pTab=%p\n", i,&tab[i],pTab);
6     pTab++;
7  }
```

Sortie

```
&tab[0]=0x55f6931df260,  pTab=0x55f6931df260
&tab[1]=0x55f6931df264,  pTab=0x55f6931df264
&tab[2]=0x55f6931df268,  pTab=0x55f6931df268
&tab[3]=0x55f6931df26c,  pTab=0x55f6931df26c
&tab[4]=0x55f6931df270,  pTab=0x55f6931df270
```

Allocation non contigue: comment faire?

On suppose que l'utilisateur pourra donner les valeurs de N et M qui seront les dimentions de la matrice
Pour allouer une matrice de façon contigüe seulement selon ses colonnes, il faut:

- Allouer les N lignes de la matrice de façon contigüe

Allocation non contigue: comment faire?

On suppose que l'utilisateur pourra donner les valeurs de N et M qui seront les dimentions de la matrice
Pour allouer une matrice de façon contigüe seulement selon ses colonnes, il faut:

- Allouer les N lignes de la matrice de façon contigüe
- Allouer à chaque ligne M colonne de façon contigüe cette fois aussi

Allocation non contigue: comment faire?

On suppose que l'utilisateur pourra donner les valeurs de N et M qui seront les dimentions de la matrice
Pour allouer une matrice de façon contigüe seulement selon ses colonnes, il faut:

- Allouer les N lignes de la matrice de façon contigüe
- Allouer à chaque ligne M colonne de façon contigüe cette fois aussi

Ceci vient du fait qu'une matrice n'est qu'un tableau contenant en chaque cases des tableaux. Ce sont donc des tableaux en 2D.

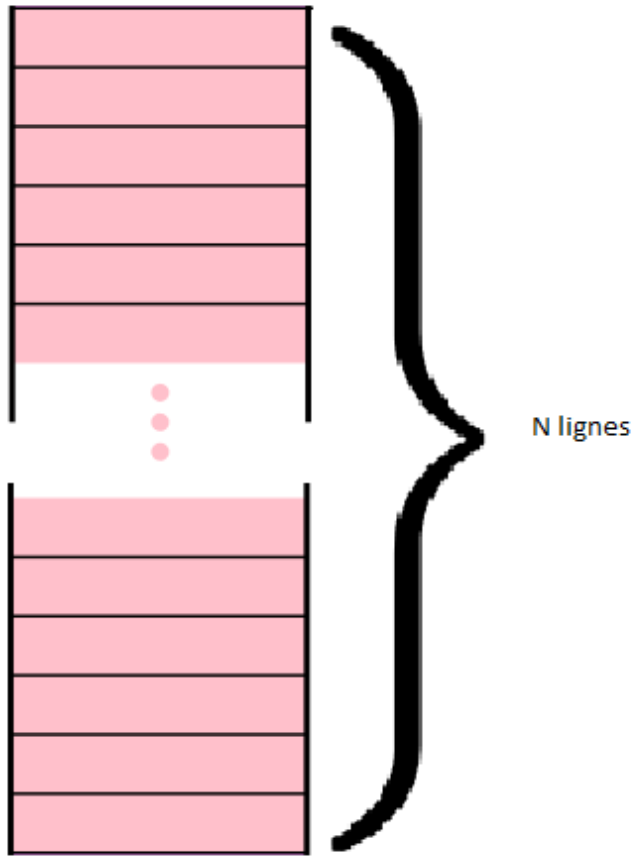
Allocation non contigüe: comment faire?

Le code:

```
1 temp->val = (double ** ) calloc(N, sizeof(double *));
2 for(int i = 0; i < N; i++ )
3     temp->val[i] = (double *)calloc(M, sizeof(double));
```

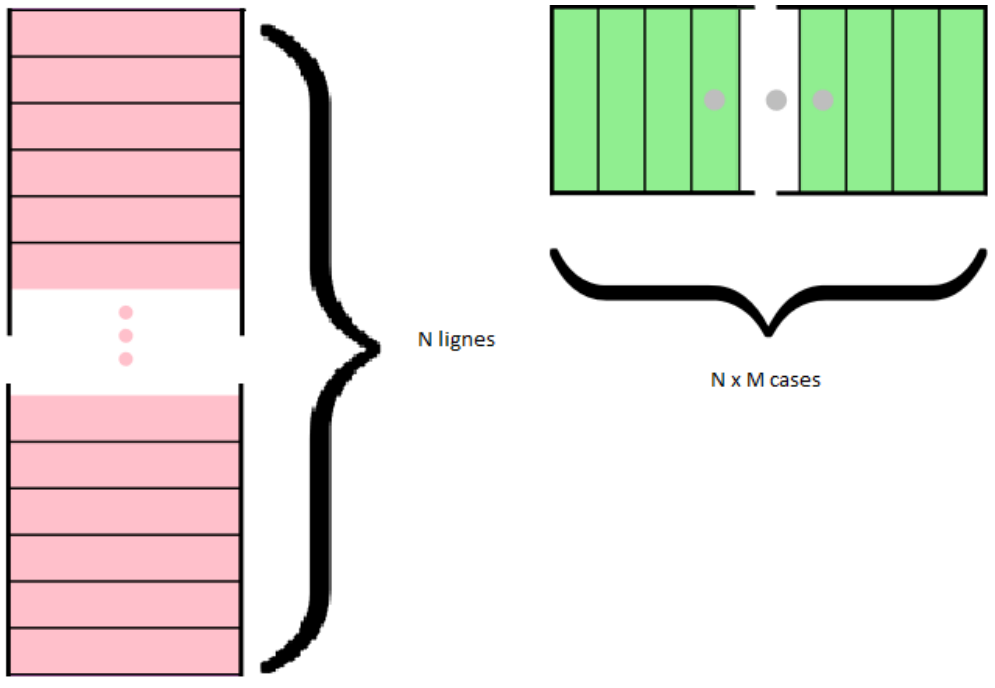
Allocation contigüe: principe

Question: Comment allouer de façon contigüe en gardant toujours la notation à 2 entrées ($tab[i][j]$)?

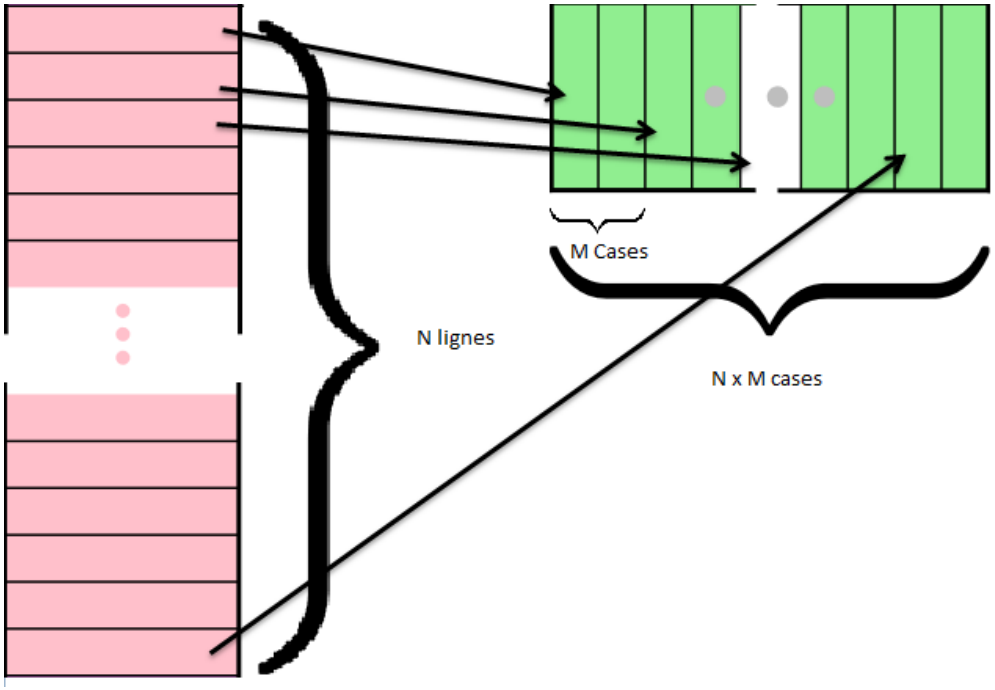


Allocation contigüe: principe

Question: Comment allouer de façon contigüe?



Question: Comment allouer de façon contigüe en gardant toujours la notation à 2 entrées ($tab[i][j]$)?


M Cases
N lignes
N x M cases

Allocation contigüe: exemple de code

Code "simple":

```
1  double **val = (double**) malloc(N*sizeof(double*));
2  double *x = (double*)calloc(N*M, sizeof(double));
3  for(int i = 0; i < N; i++)
4       val[i] = x +i*M;
```

Allocation contigüe: exemple de code inutilement complexe

Code assez compliqué qui effectue la même tâche:

```
1       double * tmp;
2       int i;
3       tmp = (double*) malloc(N*M*sizeof(double));
4       (*ptr) = (double**)malloc(N*sizeof(double*));
5       for(i = 0; i< N; i++)
6           (*ptr)[i] = &tmp[(i*sizeof(double))*M];
```