

## Parallélisme

## LOI D'AMDAHL

### COURS 6 - ORDONNANCEMENT EN MÉMOIRE PARTAGÉE

ERIC GOUBAULT  
COMMISSARIAT À L'ÉNERGIE ATOMIQUE & CHAIRE ÉCOLE  
POLYTECHNIQUE/THALÈS  
SACLAY

LE 16 FÉVRIER 2011

- $s$  pourcent séquentiel,
- sur  $p$  processeurs,
- accélération du calcul complet par rapport à un calcul séquentiel sera au maximum de,

$$\frac{1}{s + \frac{1-s}{p}}$$

(maximum de  $\frac{1}{s}$ )

Conséquence: même si on parallélise 80 pourcent d'un code (le reste étant séquentiel), on ne pourra jamais dépasser, quel que soit la machine cible, une accélération d'un facteur 5!

## PROBLÈMES D'ORDONNANCEMENT

- Revenons à la mémoire partagée... et aux PRAM (CREW)!
- Problème important: paralléliser les nids de boucle
- ou au moins comprendre ce qui est intrinsèquement séquentiel, de ce qui peut être calculé en parallèle

## EXEMPLE DE NID DE BOUCLE

```
for (i=1;i<=N;i++) {
  for (j=i;j<=N+1;j++)
    for (k=j-i;k<=N;k++) {
      S1;
      S2;
    }
  for (r=1;r<=N;r++)
    S3;
}
```

## EXPLICATION

- Ceci n'est pas un nid de boucle parfait:
- **S1** et **S2** sont englobées par les boucles **i**, **j** et **k**
- alors que **S3** est englobée par les boucles **i** et **r**

5

## VECTEURS D'ITÉRATION

- Les itérations de  $n$  boucles parfaitement imbriquées sont représentées par un vecteur de dimension  $n$
- Par exemple, pour **S3** on a un vecteur d'itération en  $(i, r)$  dont le domaine est  $1 \leq i, r \leq N$
- Pour **S1** on a un vecteur d'itération en  $(i, j, k)$  dont le domaine est  $1 \leq i \leq N, i \leq j \leq N + 1$  et  $j - i \leq k \leq N$ .
- On note une instance de  $S$  à l'itération  $I$ ,  $S(I)$
- On suppose que les domaines d'itération forment des polyèdres.

## ORDRE SÉQUENTIEL D'EXÉCUTION

Définit l'ordre d'exécution "par défaut":

- Pour les nids de boucles non-parfaits, les domaines d'itérations sont incomparables a priori
- Il suffit de "compléter" les vecteurs d'itération de façon cohérente:  
 $I \rightarrow \tilde{I}$

- $$S(I) <_{seq} T(J) \Leftrightarrow$$
$$(\tilde{I} <_{seq} \tilde{J}) \text{ ou}$$
$$(\tilde{I} = \tilde{J} \text{ et } S <_{text} T)$$

7

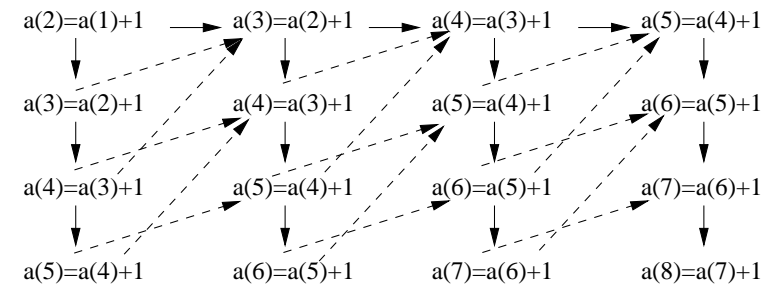
## DÉPENDANCES DE DONNÉES

- On veut mettre en parallèle certaines instructions: une partie de l'ordre séquentiel est à respecter absolument, une autre pas (permutation possible d'actions)
- On va définir un ordre partiel (Bernstein), ordre minimal à respecter pour produire un code sémantiquement équivalent à l'ordre initial
- En fait, l'ordre séquentiel va être une *extension* de l'ordre partiel de Bernstein
- Cet ordre partiel est défini à partir de 3 types de dépendances de données: flot, anti et sortie.

## FLOT, ANTI ET SORTIE...

- Toujours dirigées par l'ordre séquentiel
- Dépendance de flot de  $S(I)$  vers  $T(J)$ : si un emplacement mémoire commun est en écriture pour  $S(I)$  et en lecture pour  $T(J)$
- Dépendance anti de  $S(I)$  vers  $T(J)$ : si un emplacement mémoire commun est en lecture pour  $S(I)$  et en écriture pour  $T(J)$
- Dépendance de sortie de  $S(I)$  vers  $T(J)$ : si un emplacement mémoire commun est en écriture pour  $S(I)$  et en écriture pour  $T(J)$

## DÉPENDANCES



11

## CALCUL DES DÉPENDANCES

- Supposons que  $S(I)$  et  $T(J)$  accèdent au même tableau  $a$  ( $S$  en écriture,  $T$  en lecture)
- $S(I) : a(f(I)) = \dots$  et  $T(J) : \dots = a(g(J))$ ;
- L'accès au tableau est commun si  $f(I) = g(J)$
- Si  $f$  et  $g$  sont des fonctions affines à coefficients entiers: peut se tester en temps polynomial

## EXEMPLE

```
for (i=1;i<=N;i++)
  for (j=1;j<=N;j++)
    a(i+j) = a(i+j-1)+1;
```

## CALCUL DES DÉPENDANCES

- Chercher une dépendance de flot par exemple revient à prouver en plus  $S(I) <_{seq} T(J)$
- Encore une fois: en général résolution de systèmes d'égalités et d'inégalités affines
- Chercher les dépendances *directes*: plus compliqué (par programmation linéaire, problèmes d'optimisation dans les entiers → algorithme de Fourier-Motzkin, omega-test de B. Pugh utilisé dans **petit**)

13

## EXEMPLE

- $f(I) = i + j$  et  $g(J) = i + j - 1$
- On cherche dépendance entre écriture  $S(i', j')$  et lecture  $S(i, j)$
- $f(I) = g(J) \Leftrightarrow i' + j' = i + j - 1$
- $S(i', j') <_{seq} S(i, j) \Leftrightarrow ((i' \leq i - 1) \text{ ou } (i = i' \text{ et } j' \leq j - 1))$

## EXEMPLE

- Dépendance de flot directe implique résoudre:

$$\max_{<_{seq}} \{(i', j') \mid (i', j') <_{seq} (i, j), i' + j' = i + j - 1, 1 \leq i, i', j, j' \leq\}$$

- Solution:

$$\begin{aligned} &(i, j - 1) \text{ si } j \geq 2 \\ &(i - 1, j) \text{ si } j = 1 \end{aligned}$$

- Antidépendance directe de  $S(i, j)$  vers  $S(i', j')$  implique résoudre:

$$\min_{<_{seq}} \{(i', j') \mid (i, j) <_{seq} (i', j'), i' + j' = i + j - 1, 1 \leq i, i', j, j' \leq\}$$

- Solution (pour  $j \geq 3, i \leq N - 1$ ):

$$(i + 1, j - 2)$$

15

## APPROXIMATION DES DÉPENDANCES

Graphe de Dépendance Etendu (GDE):

- Sommets: instances  $S_i(I), 1 \leq i \leq s$  et  $I \in D_{S_i}$
- Arcs:  $S(I) \rightarrow T(J)$  pour chaque dépendance

## QUELQUES DÉFINITIONS

- Ensemble des paires de dépendances entre  $S$  et  $T$ :

$$\{(I, J) \mid S(I) \rightarrow T(J)\} \subseteq \mathbb{Z}^{n_S} \times \mathbb{Z}^{n_T}$$

- Ensemble de distance de ces dépendances:

$$\{(\tilde{J} - \tilde{I}) \mid S(I) \rightarrow T(J)\} \subseteq \mathbb{Z}^{n_{S,T}}$$

- Problème: on ne peut calculer le GDE à la compilation! (de toutes façons, est trop gros!)

17

## GRAPHE DE DÉPENDANCE RÉDUIT

...ou GDR:

- Sommets: instructions  $S_i$  ( $1 \leq i \leq s$ )
- Arcs:  $e : S \rightarrow T$  si il existe au moins un arc  $S(I) \rightarrow T(J)$  dans le GDE
- Etiquette:  $w(e)$  décrivant un sous-ensemble  $D_e$  de  $\mathbb{Z}^{n_{S,T}}$

Le tri topologique du GDR permet d'avoir une idée des portions séquentielles, et des portions parallélisables.

## QUELQUES ABSTRACTIONS

...des étiquettes du GDR. Par exemple, niveaux de dépendance (GDRN):

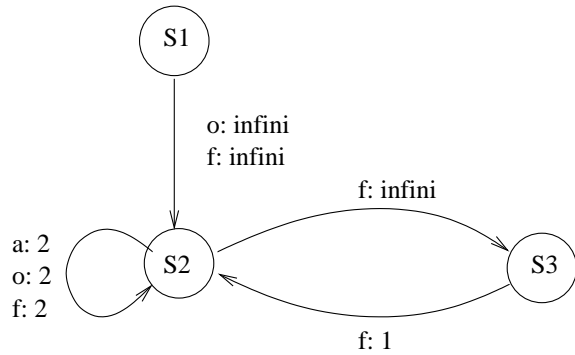
- Une dépendance entre  $S(I)$  et  $T(J)$  est boucle indépendante si elle a lieu au cours d'une même itération des boucles englobant  $S$  et  $T$
- Sinon elle est portée par la boucle...
- Etiquetage en conséquence, de  $e : S \rightarrow T$  du GDR:
  - $l(e) = \infty$  si  $S(I) \rightarrow T(J)$  avec  $\tilde{J} - \tilde{I} = 0$
  - $l(e) \in [1, n_{S,T}]$  si  $S(I) \rightarrow T(J)$ , et la première composante non nulle de  $\tilde{J} - \tilde{I}$  est la  $l(e)^{ième}$  composante

19

## EXEMPLE

```
for (i=2;i<=N;i++) {
  S1: s(i) = 0;
  for (j=1;j<i-1;j++)
    S2: s(i) = s(i)+a(j,i)*b(j);
  b(i) = b(i)-s(i);
}
```

GDRN



21

VECTEURS DE DIRECTION

Dans notre exemple:

- lecture  $a(i+j-2)$ -écriture  $a(i+j)$ : ensemble de distance  $\{(1, -2)\}$
- écriture  $a(i+j)$ -lecture  $a(i+j-1)$ : ensemble de distance  $\{(1, 0), (0, 1)\}$
- écriture  $a(i+j)$ -écriture  $a(i+j)$ : ensemble de distance  $\{(1, -1)\}$

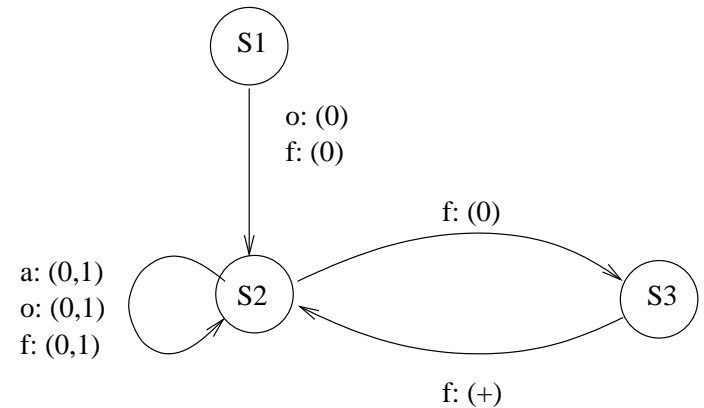
VECTEURS DE DIRECTION

Sont des abstractions de ces ensembles de distance:

- on note  $z+$  pour une composante si toutes les distances sur cette composante ont au moins la valeur  $z$
- on note  $z-$  pour une composante si toutes les distances sur cette composante ont au plus la valeur  $z$
- on note  $+$  à la place de  $1+$ ,  $-$  à la place de  $-1-$
- on note  $*$  si la composante peut prendre n'importe quelle valeur
- on note  $z$  si la composante a toujours la valeur  $z$

23

GDRV



## DÉMO PETIT

```

Petit
1: Entry
2: in integer N
3: inout real s(1:N), a(1:N,1:N), b(1:N)
4: for i = 2,N do
5:   s(i) = 0
6:   for j = 1,i-1 do
7:     s(i) = s(i)+a(j,i)*b(j)
8:   endfor
9:   b(i) = b(i)-s(i)
4: endfor
10: Exit
    
```

25

## DÉPENDANCES SOUS PETIT

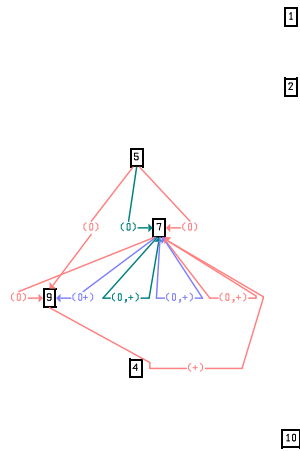
```

Petit
1: Entry
2: in integer N
3: inout real s(1:N), a(1:N,1:N), b(1:N)
4: for i = 2,N do
5:   s(i) = 0
6:   for j = 1,i-1 do
7:     s(i) = s(i)+a(j,i)*b(j)
8:   endfor
9:   b(i) = b(i)-s(i)
4: endfor
10: Exit
    
```

flow 9: b(i) --> 7: b(j) (+) [W0]

27

## GRAPHE DE DÉPENDANCES SOUS PETIT



## INFORMATIONS SUR LES DÉPENDANCES SOUS PETIT

```

dependenceWindow
Filters:  flow  output  anti  reduce
 memory  value  "carry"  "cyclic"  scalar
flow 9: b(i) --> 7: b(j) (+) [W0]
flow 9: b(i) --> 10: Exit [W]
    
```

{[i] -> [i',i] : 2 <= i < i' <= N}

## TRANSFORMATIONS DE BOUCLES

- Distribution de boucle
- Torsion de boucle
- Inversion de boucle
- Permutation de boucle
- etc.

29

## DISTRIBUTION DE BOUCLES

- Toute instance d'une instruction  $S$  peut être exécutée avant toute instance de  $T$  si on n'a jamais  $T(J) \rightarrow S(I)$  dans le GDE

- Code:

```
for (i=...) {  
    S1;  
    S2;  
}
```

31

## ALGORITHMES DE PARALLÉLISATION DE BOUCLES

- Allen et Kennedy (basé sur le GDRN) : distribution de boucles
- Lamport - transformations unimodulaires (basé sur le GDRV) : torsion, inversion, permutation de boucles

## TRANSFORMATION

Si on n'a pas de dépendance de  $S2$  vers  $S1$  ni de  $S1$  vers  $S2$ , alors on peut transformer le code en:

```
for (i=...)  
    S1;  
for (i=...)  
    S2;
```



### FUSION DE BOUCLES

```
forall (i=1;i<=N;i++)  
  D[i]=E[i]+F[i];  
forall (j=1;j<=N;j++)  
  E[j]=D[j]*F[j];
```

devient:

```
forall (i=1;i<=N;i++)  
{  
  D[i]=E[i]+F[i];  
  E[j]=D[j]*F[j];  
}
```

Cela permet une vectorisation et donc une réduction du coût des boucles parallèles.

33

### COMPOSITION DE BOUCLES

```
forall (j=1;j<=N;j++)  
  forall (k=1;k<=N;k++)  
  ...
```

devient,

```
forall (i=1;i<=N*N;i++)  
  ...
```

Cela permet de changer l'espace d'itérations (afin d'effectuer éventuellement d'autres transformations).

### ECHANGE DE BOUCLES

```
for (i=2;i<=N;i++)  
  for (j=2;j<=M;j++)  
    A[i,j]=A[i,j-1]+1;
```

devient,

```
for (j=1;j<=M;j++)  
  A[1:N,J]=A[1:N,j-1]+1;
```

35

### RÉDUCTION DE BOUCLE

Dans le cas où le cycle interne de dépendance est de type dépendance de flot on ne peut pas utiliser la méthode précédente mais, par exemple:

```
for (i=3;i<=N;i++)  
{ A[i]=B[i-2]-1;  
  B[i]=A[i-3]*k; }
```

devient,

```
for (j=3;j<=N;j=j+2)  
  forall (i=j; i<=j+1; i++)  
  { A[i]=B[i-2]-1;  
    B[i]=A[i-3]*k; }
```

## DÉROULEMENT DE BOUCLE

```
for (i=1;i<=100;i++)
  A[i]=B[i+2]*C[i-1];
devient,
for (i=1;i<=99;i=i+2)
{
  A[i]=B[i+2]*C[i-1];
  A[i+1]=B[i+3]*C[i];
}
```

37

## ROTATION DE BOUCLE

[skewing]  
Pour le code:

```
for (i=1;i<=N;i++)
  for (j=1;j<=N;j++)
    a[i,j]=(a[i-1,j]+a[i,j-1])/2;
```

## TRANSFORMATION

En faisant une rotation de l'espace d'itérations de 45 degrés:

```
for (k=2;k<=N;k++)
  forall (l=2-k;l<=k-2;l+=2)
    a[(k-1)/2,(k+1)/2] = (a[(k-1)/2-1,(k+1)/2]+
                          a[(k-1)/2,(k+1)/2-1])/2;
for (k=1;k<=N;k++)
  forall (l=k-N;l<=N-k;l+=2)
    a[(k-1)/2,(k+1)/2] = (a[(k-1)/2-1,(k+1)/2]+
                          a[(k-1)/2,(k+1)/2-1])/2;
```

39

## EXEMPLE DE PARALLÉLISATION DE CODE

Phase de *remonte* après une décomposition LU:  
Soit donc à résoudre le système triangulaire supérieur

$$Ux = b$$

avec,

$$U = \begin{pmatrix} U_{1,1} & U_{1,2} & U_{1,3} & \cdots & U_{1,n} \\ 0 & U_{2,2} & U_{2,3} & \cdots & U_{2,n} \\ & & & \cdots & \\ 0 & 0 & \cdots & 0 & U_{n,n} \end{pmatrix}$$

et  $U_{i,i} \neq 0$  pour tout  $1 \leq i \leq n$ .

REMONTÉE...

On procède par “remontée” c’est à dire que l’on calcule successivement,

$$x_n = \frac{b_n}{U_{n,n}}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^n U_{i,j} x_j}{U_{i,i}}$$

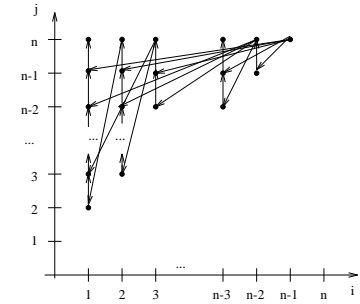
pour  $i = n - 1, n - 2, \dots, 1$ .

41

CODE SÉQUENTIEL

```
x[n]=b[n]/U[n,n];
for (i=n-1;i>=1;i--)
{
x[i]=0;
for (j=i+1;j<=n;j++)
L: x[i]=x[i]+U[i,j]*x[j];
x[i]=(b[i]-x[i])/U[i,i];
}
```

GRAPHE DE DÉPENDANCES



Il y a aussi des anti-dépendances des itérations  $(i, j)$  vers  $(i - 1, j)$ ...

43

PARALLÉLISATION

En faisant une rotation et une distribution de boucles:

```
H': forall (i=1;i<=n-1;i++)
    x[i]=b[i];
T': x[n]=b[n]/U[n][n];
H: for (t=1;t<=n-1;t++)
    forall (i=1;i<=n-t;i++)
        L: x[i]=x[i]-x[n-t+1]*U[i][n-t+1];
        T: x[n-t]=x[n-t]/U[n-t][n-t];
```

Le ratio d'accélération est d'ordre  $\frac{n}{4}$  asymptotiquement...

## Coût

En ne regardant que la partie **H** du code (la seule vraiment importante asymptotiquement), et on a:

- un coût de 2 pour **L**,
- un coût de 1 pour rapatrier  $\mathbf{x}[\mathbf{n-t}+1]$

45

## Coût SIMD

- tous ces calculs se recouvrent dans le **forall**. Ils sont donc à sommer pour chaque boucle sur **t**,
- tout comme le coût égal à un de **T**, donc un coût de  $4(n-1)$  pour **H**,
- a cela, il faut ajouter le temps pour **H'** égal à un puisque nous sommes en mode SIMD et encore un pour **T'**. Donc un coût total de  $4n-2$ .

## Coût séquentiel

Pour le code séquentiel,

- si on compte l'affectation à zéro comme une opération élémentaire de coût unitaire,
- le coût total est,

$$1 + 3(n-1) + 2(1 + 2 + \dots + n-1) = (n-1)(n+3) + 1$$

Le ratio d'accélération est donc de l'ordre de  $\frac{n}{4}$  asymptotiquement.

47

## ALLEN ET KENNEDY: PRINCIPE

- Remplacer certaines boucles **for** par des boucles **forall**
- Utiliser la distribution de boucles pour diminuer le nombre d'instructions dans les boucles, et donc réduire les dépendances

## ALGORITHME

- Commencer avec  $k = 1$
- Supprimer dans le GDRN  $G$  toutes les arêtes de niveau inférieur à  $k$
- Calculer les composantes fortement connexes (CFC) de  $G$
- Pour tout CFC  $C$  dans l'ordre topologique:
  - Si  $C$  est réduit à une seule instruction  $S$  sans arête, alors générer des boucles parallèles dans toutes les dimensions restantes (i.e. niveaux  $k$  à  $n_S$ ) et générer le code pour  $S$
  - Sinon,  $l = l_{min}(C)$ , et générer des boucles parallèles du niveau  $k$  au niveau  $l - 1$ , et une boucle séquentielle pour le niveau  $l$ . Puis reboucler l'algorithme avec  $C$  et  $k = l + 1$

49

## EXEMPLE

```
for (i=1;i<=N;i++)
  for (j=1;j<=N;i++) {
    S1: a(i+1,j+1) = a(i+1,j)+b(i,j+2);
    S2: b(i+1,j) = a(i+1,j-1)+b(i,j-1);
    S3: a(i,j+2) = b(i+1,j+1)-1;
  }
```

## EXEMPLE

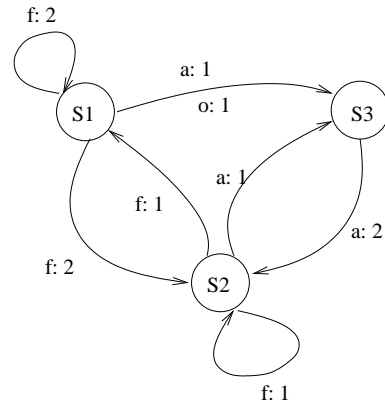
- Flot  $S1 \rightarrow S1$ , variable  $a$ , distance  $(0, 1)$ ,
- Flot  $S1 \rightarrow S2$ , variable  $a$ , distance  $(0, 2)$ ,
- Flot  $S2 \rightarrow S1$ , variable  $b$ , distance  $(1, -2)$ ,
- Flot  $S2 \rightarrow S2$ , variable  $b$ , distance  $(1, 1)$ ,

51

## EXEMPLE

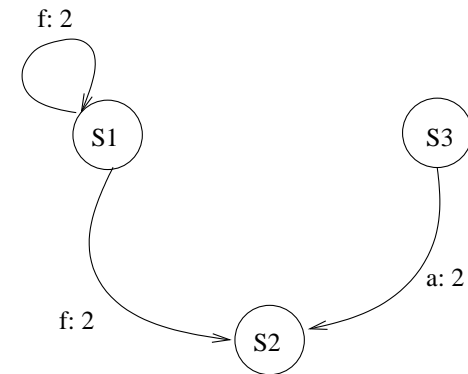
- Anti  $S1 \rightarrow S3$ , variable  $a$ , distance  $(1, -2)$ ,
- Anti  $S2 \rightarrow S3$ , variable  $a$ , distance  $(1, -3)$ ,
- Anti  $S3 \rightarrow S2$ , variable  $b$ , distance  $(0, 1)$ ,
- Sortie  $S1 \rightarrow S3$ , variable  $a$ , distance  $(1, -1)$ .

## EXEMPLE



53

## GDRN MODIFIÉ



55

## ALGORITHME

- Le GDRN est fortement connexe et a des dépendances de niveau 1
- La boucle sur  $i$  sera donc séquentielle
- On enlève maintenant les dépendances de niveau 1...

## PARALLÉLISATION FINALE

```
for (i=1;i<=N;i++) {
  for (j=1;j<=N;j++)
    S1: a(i+1,j+1) = a(i+1,j)+b(i,j+2);
  forall (j=1;j<=N;j++)
    S3: a(i,j+2) = b(i+1,j+1)-1;
  forall (j=1;j<=N;j++)
    S2: b(i+1,j) = a(i+1,j-1)+b(i,j-1);
}
```