TD 7 : Nids de boucles

Exercice 1 - Vecteurs de dépendance (rattrapage 2022)

Déterminer l'ensemble des vecteurs de dépendance dans les codes `C` suivants.

1.1

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

1.2

```
for (i = 0; i < N; ++i) {
  for (j = 1; j < N; ++j)
    A[i][j] = A[i][j-1] + B[i][j];
  for (j = 0; j < N-1; ++j)
    B[i][j+1] = A[i][j];
}
```

Exercice 2 - Vecteurs de dépendance (exam. 2021)

On considère le code `C` suivant :

```
for (i = 0; i < N-2; ++i)
    for (j = 1; j < N; ++j) {
        A[i+2][j] = A[i][j] * B[i][j];
        B[i+1][j] = B[i][j-1] + A[i][j-1];
    }
```

2.1 Déterminer l'ensemble des vecteurs de dépendance.

2.2 Peut-on faire les transformations suivantes sans changer la sémantique du programme :

(i) échanger les boucles ?
(ii) inverser l'ordre dans la boucle interne ?
(iii) faire une fission de la boucle interne ?
(iv) faire une fission de la boucle interne puis de la boucle externe ?

Exercice 3 - Nid de boucle (exam. 2023)

On considère le code `C` suivant :

```
1  for (i=0; i < N; ++i) {
2      for (j=0; j<N; ++j) {
3          S[j] += A[i][j];
4          if (j+1<N) A[i][j+1] += A[i][j];
5      }
6  }
```

3.1 Soit $u \in [\![0,\ N-1]\!]$. Combien de fois accède-t-on à la case mémoire désignée par `S[u]` ?

3.2 Déterminer l'ensemble des vecteurs de dépendance.

3.3 Montrer que l'échange des deux boucles est une transformation valide du code.

3.4 Comparer l'usage du cache dans les deux versions de code (avant et après échange), puis déterminer la version qui sera la plus efficace en pratique.

3.5 Pour éviter le `if` à la ligne 4, on propose d'opérer une fission de la boucle interne. On aura alors deux boucles `for` utilisant la variable `j`, et on pourra mettre `j < N-1` comme test d'arrêt dans la seconde boucle.

Cette transformation est-elle valide ?