并行程序设计作业

魏星迪 SA24011277

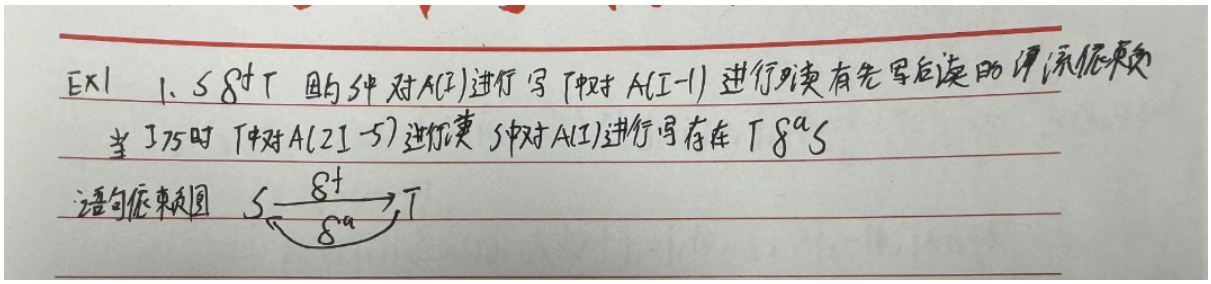
2024.11.9

1 ex1

1.1 ex1.1

1. 找出以下循环中的存在依赖关系(包括依赖类型)，画出语句依赖图。

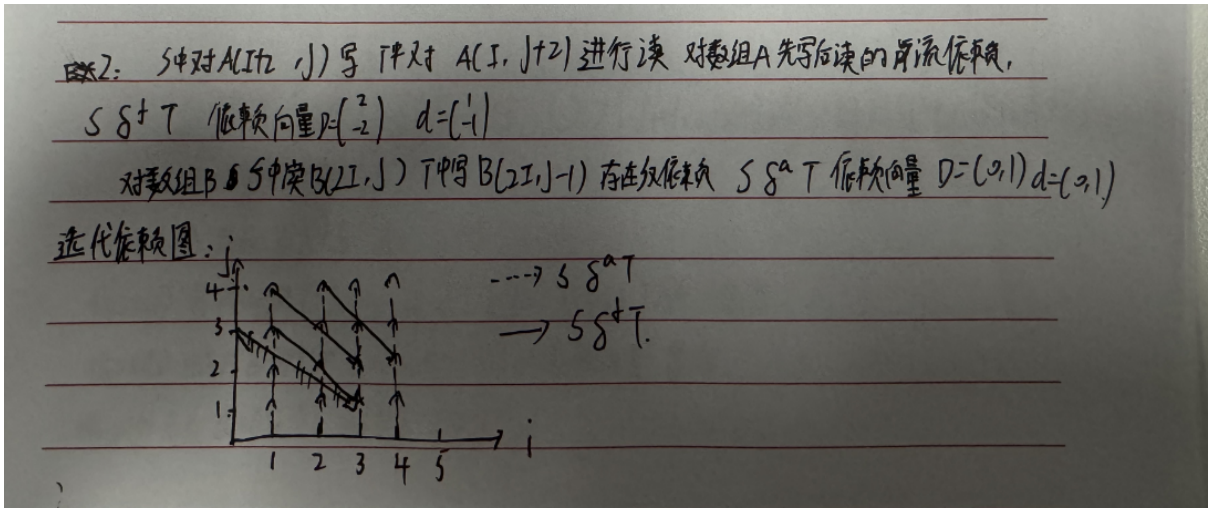
```
for I = 0 to 100 do
S:    A(I) = C(I)+2;
T:        B(I) = A(I-1)-A(2*I-5);
    end for
```



1.2 ex1.2

2. 找出以下循环中的存在依赖关系(包括依赖类型、依赖向量)，画出迭代依赖图(注意：要“窥一斑而知全豹”)。

```
for I = 1 to 100 do
    for J = 1 to 50 do
S:      A(I+2,J) = B(2*I,J) - 5;
T:      B(2*I,J-1) = A(I,J+2) + 4;
    end for
end for
```



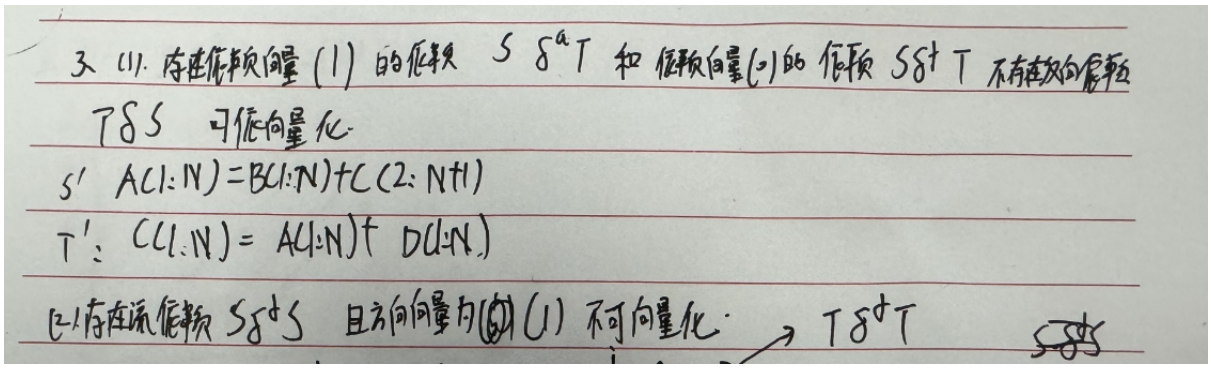
1.3 ex1.3

3. 向量化以下循环。如果不能，请说明原因。

(1)
```
for I = 1 to N do
S:    A(I) = B(I) + C(I+1);
T:    C(I) = A(I)* D(I);
end for
```

(2)
```
for I = 1 to N do
S:      A(I) = A(I-1) + 1
end for
```

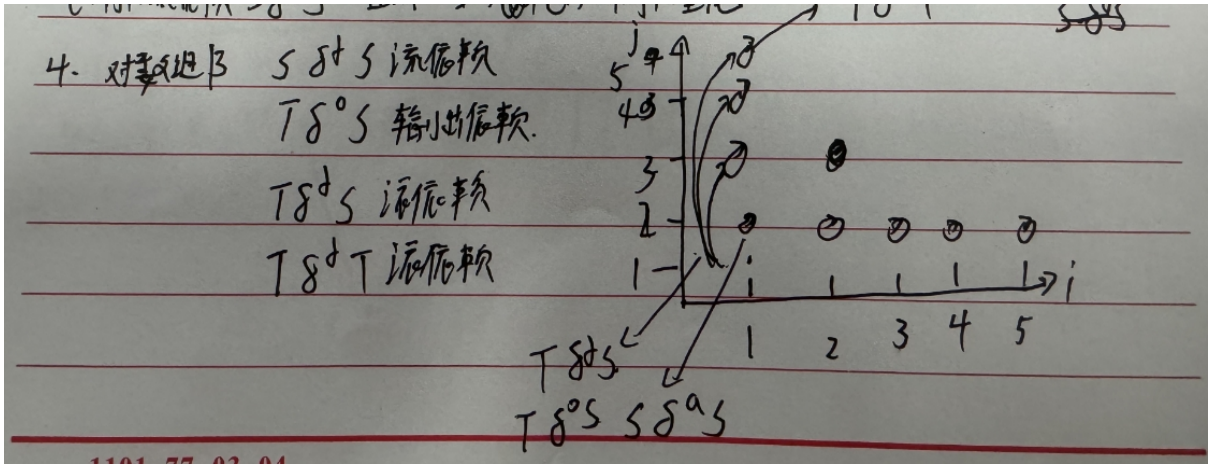
↗距离为0　　↑距离为1



1.4 ex1.4

4. 分析以下循环中的存在依赖关系(包括依赖类型)，画出迭代依赖图。

```
for I = 1 to 5 do
S:  B(I) = B(I) / A(I,I);
    for J = I+1 to 5 do
T:      B(J) = B(J) - A(I,J) * B(I);
    end for
end for
```

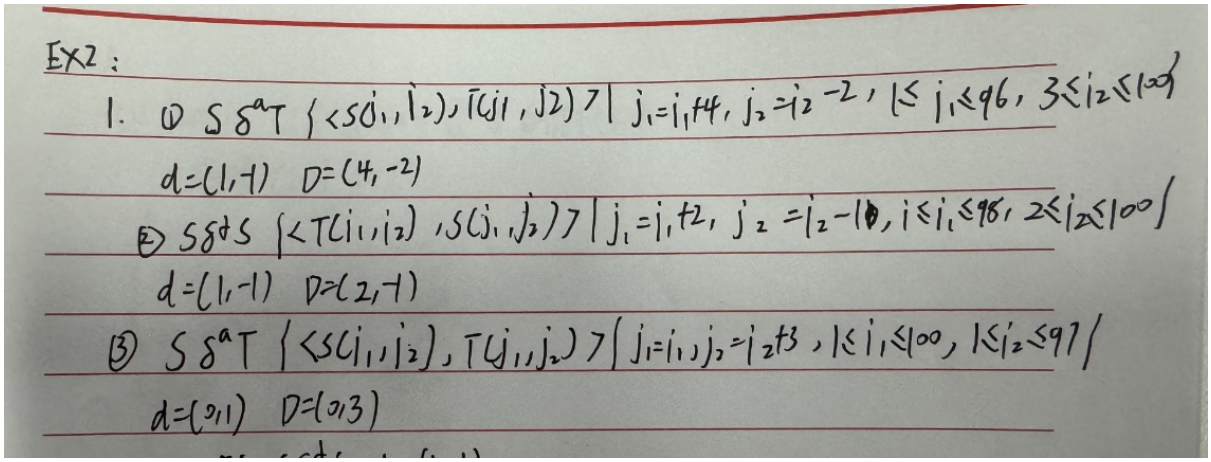


2 EX2

2.1 ex2.1

1. 描述以下循环中的存在依赖关系(包括迭代对、依赖类型、依赖向量和距离向量)

```
for I = 1 to 100 do
    for J = 1 to 100 do
S:      A(I,J) = B(I+4,J-2) - B(I-2,J+1) + B(I,J+3);
T:      B(I,J) = D(I,J-1) - C(I+2,J)
    endfor
endfor
```



2.2 ex2.2

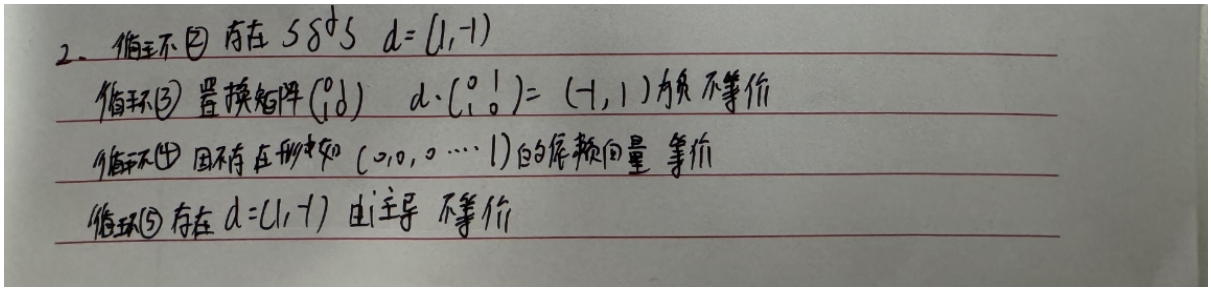
2．分析循环②是否分别与循环③、④和⑤等价？

循环②：
```
for I = 1 to 100 do
    for J = 4 to 100 do
S:      A(I,J) = A(I-1,J+1)
    endfor
endfor
```

循环③：
```
for J = 4 to 100 do
    for I = 1 to 100 do
S:      A(I,J) = A(I-1,J+1)
    endfor
endfor
```

循环④：
```
for I = 1 to 100 do
    doall J = 4 to 100 do
S:      A(I,J) = A(I-1,J+1)
    enddoall
endfor
```

循环⑤：
```
doall I = 1 to 100 do
    for J = 4 to 100 do
S:      A(I,J) = A(I-1,J+1)
    endfor
enddoall
```



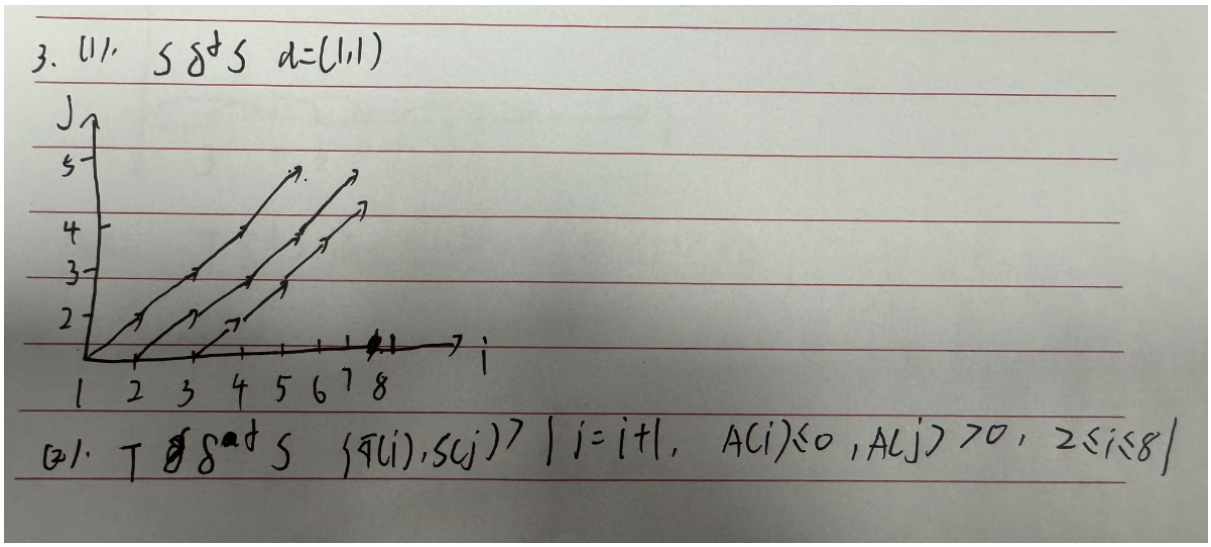
2.3 ex2.3

3 （1）给出下面循环中的依赖关系描述和迭代依赖图。

```
for I = 1 to 8 do
    for J = max(I-3,1) to min(I,5) do
S:      A(I+1, J+1) = A(I,J) + B(I,J)
    endfor
endfor
```

（2）分析下面循环中存在的数据依赖关系。

```
for I = 2 to 9 do
    if A(I) > 0 then
S:      A(I) = B(I-1) + 1
    else
T:      B(I) = A(I) * 2
    endif
endfor
```



3 EX3

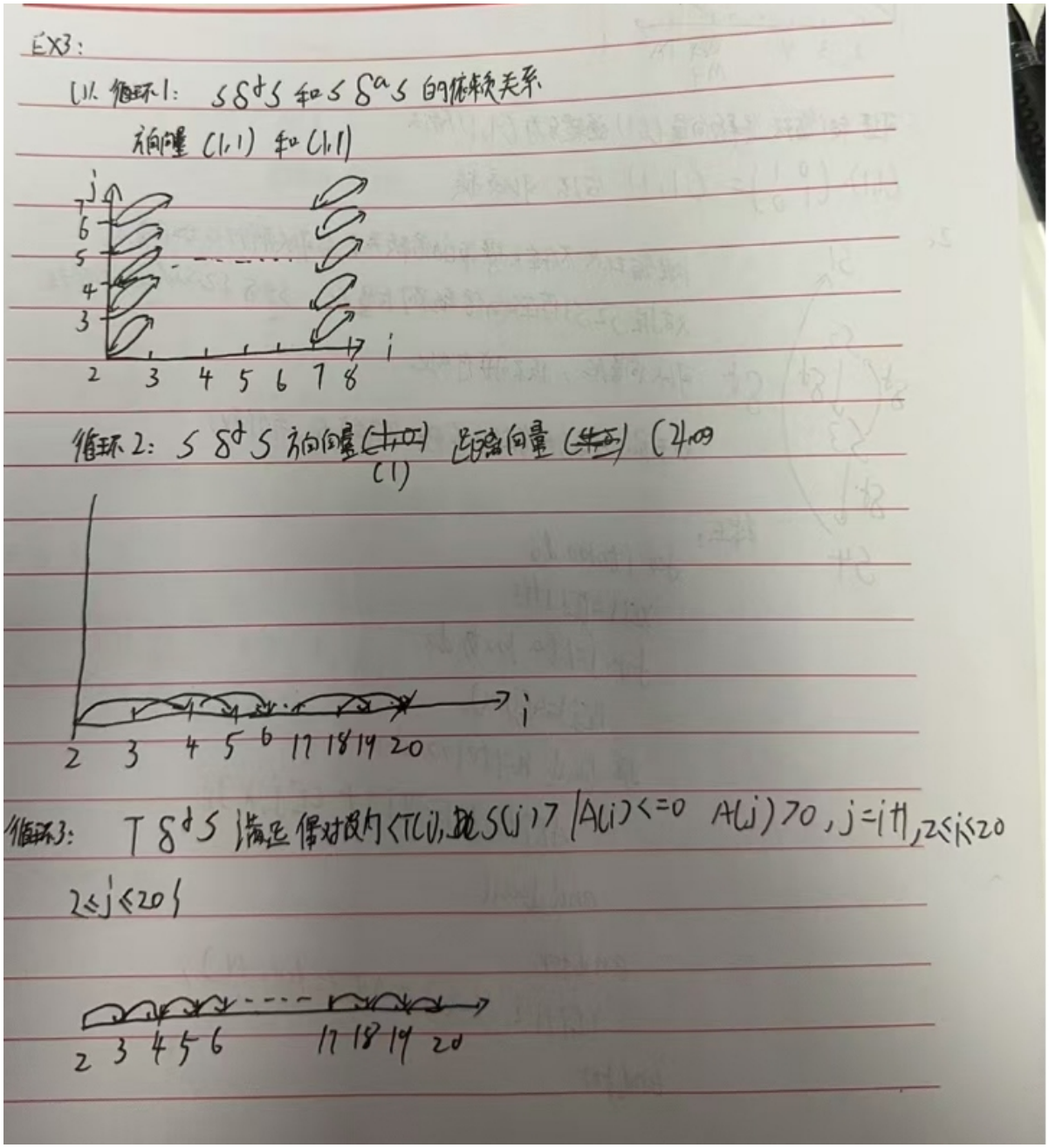
3.1 ex3.1

一、 分析以下循环中的依赖关系，并给出相应的迭代依赖图：

```
for i = 2 to 10 do   //循环 1
  for j = 2 to 10
S1:  A[i,j] =  ( A[i-1,j-1] + A[i+1,j+1] )* 0.5;
  endfor
endfor
```

```
for i = 2 to 20 do   // 循环 2
S2: A[2*i+2] = A[2*i-2] + B[i];
endfor
```

```
for i = 2 to 20 do  // 循环 3
   if A[i] > 0 then
S3     B[i] = C[i-1] + 1
   else
T1:    C[i] = B[i] - 1
   endif
endfor
```



3.2 ex3.2

二、 针对以下两个循环：

```
for i = 1 to M do   //循环1  M，N，C均是常量
  for j = 1 to N
S:  A[i+1,j+1] =  A[i,j] + C;
  endfor
endfor
```

(1) 给出迭代依赖示意图。

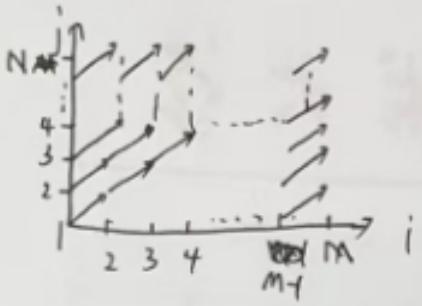
(2) 简述能否逆转外层的i循环？能否交换内外循环次序？

```
for i = 1 to 100 do // 循环2  N 是常量
   X[i] = Y[i] + 10; // 语句S1
   for j = 1 to 100 do
      B[j] = A[j, N]; // 语句S2
      for k = 1 to 100 do
         A[j+1, k] = B[j] + C[j, k]; // 语句S3
      endfor // loop-k
      Y[i+j] = A[j+1, N]; // 语句S4
   endfor // loop-j
endfor // loop-i
```

(1) 给出此循环的语句依赖图。

(2) 尝试向量化/并行化此循环。

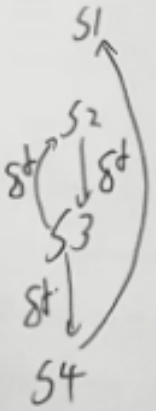
Q1. 1. S8 S5 依赖向量(1,1)^T 距离向量(1,1)



不可逆 转化循环 依赖向量(1,1) 逆变后为(-1,1) 合法

$(1,1)\cdot\begin{pmatrix}0&1\\1&0\end{pmatrix}=(1,1)$ 合法 可以换

2.



内层循环K 不存在K携带的依赖关系 S3可以并行化 和向量化

对于S2,S3存在环向依赖 向量化 S4与S2,S3存在环向依赖 可以向量化，但不可并行化

对于i说. S1与S4存在环的循环 不向量化，并行化

综上：

```
for i to 100 do
  X[i] = [C,]+1.0
  for j=1 to 100 do
    B[j]=A[j,N];
    doall k=1 to 100 do
      A[j+1,k]=B[j]+C[j,k];
    end doall
  endfor
  Y[i+1:i+100]=A[2:101,N],
endfor.
```

3.3 ex3.3

三、 针对以下循环/程序：

```
for i = 1 to 100 do   //循环 1
  for j = 1 to 50 do
S:  A[3*i+2,2*j-1] =  A[5*j,i+3] + 2;
  endfor
endfor
```

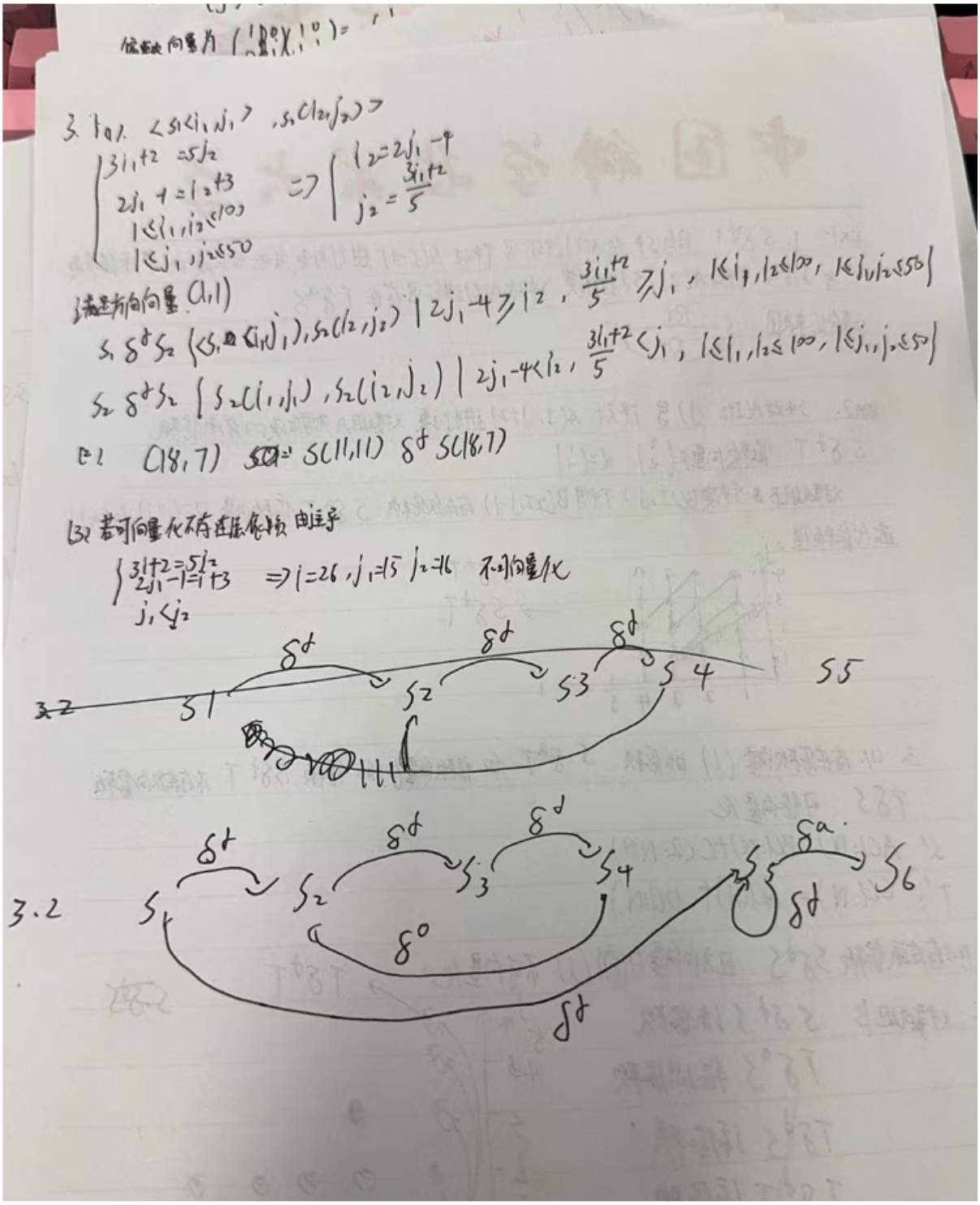
（1） 给出满足依赖方向向量(1,1)的迭代依赖对集合的描述。

（2） 找出与迭代（i=11, j=11）相依赖的迭代（m,n）并指出是哪种依赖？

（3） 能否向量化最内层的 j 循环？如不行，简述理由。

```
S1: x = y * 2
    for i = 1 to 100 do
S2:   C[i] = B[i] + x
S3:   A[i] = C[i-1] + z
S4:   C[i+1] = A[i] * B[i]
      for j = 1 to 50 do
S5:     D[i,j] = D[i,j-1] + x
      endfor
    endfor
S6: z = y + 4
```

给出上述程序的语句依赖图。


3.2
S_1
S_2
S_3
S_4
S_5
S_6
δ^d
δ^a
δ^0

3.4 ex3.4

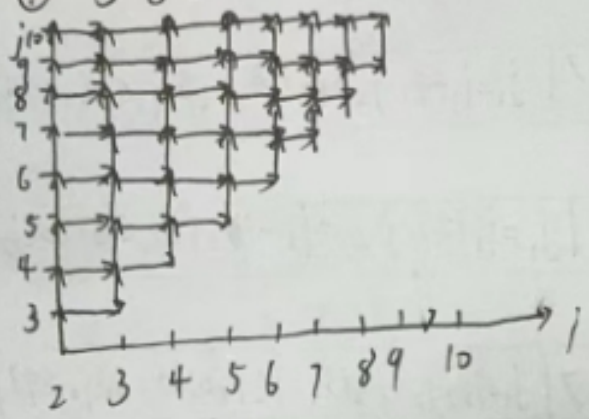
```
for i = 2 to 10 do   //循环 1
   for j = i to 10
    S A[i,j] =  ( A[i,j-1] + A[i-1,j] )* 0.5;
   endfor
endfor
```

```
for i = 1 to 16 do   // 循环 2
 S A[i+3] = A[i] + B[i];
endfor
```

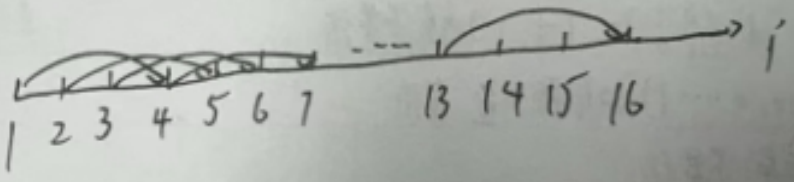
```
for k = 1 to 16 step 5 do   // 循环 3 ,k 的循环步长为 5
   for i = k to min(16,i+4) do //设 min 为求最小值函数
  S   A[i+3] = A[i] + B[i]
   endfor
endfor
```

3.4

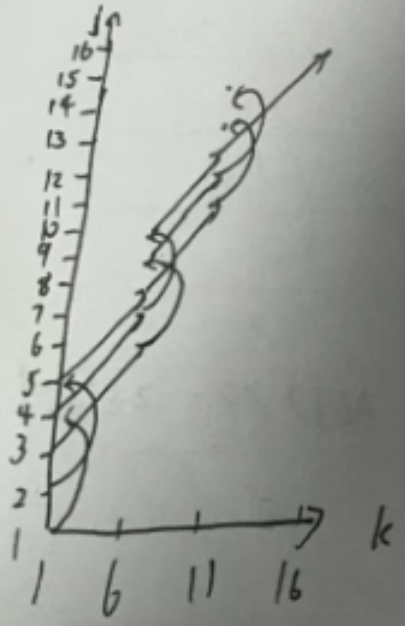
① 5 8ds (0,1), (1,0)



② 5 8ds (3)



③. 5 8ds



3.5 ex3.5

```
for i = 1 to 100 do   //循环 1
 S1 A[i] =  A[i] + B[i-1];
 S2 B[i] =  C[i-1] * 2 ;
 S3 C[i] =  1 / B[i] ;
 S4 D[i] =  C[i] * C[i] ;
endfor
```

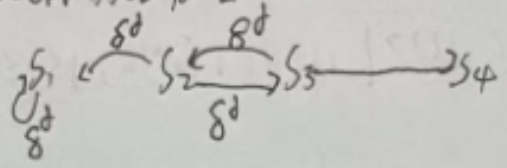
```
for i = 1 to 999 do   // 循环 2
 S A[i] = B[i] + C[i];
 T D[i] = ( A[i] + A[ 999-i+1 ] ) / 2 ;
endfor
```

```
for i = 1 to 100 do   // 循环 3
   for j = 1 to 100 do
   S  A[3*i+2*j, 2*j] = C[i,j] * 2     ;
   T  D[i,j]          = A[i-j+6, i+j] ;
   endfor
endfor
```

5.

① $S_1 \delta^* S_1$ $dD=0$ $S_2 \delta^* S_3$ $D=0$ $S_3 \delta^* S_4$ $D=0$

$S_3 \delta^* S_2$ $D=1$ $S_2 \delta^* S_1$ $D=1$



将 S_1 S_4 拉到最后此时 S_1, S_4 可向量化

```
for i=1 to 100 do
  B[i]=C[i+1]*2 ;// S2
  C[i]=1/B[i]; //S3
endfor
A[1:100]=A[1:100]+D[0:99]; //S1
D[1:100]=C[1:100]*C[1:100]; //S4
```

② $S \delta^* T$ $T \delta^* S$

S': $A[1:499]=B[1:499]+C[1:499]$;

T': $D[1:499]=(A[1:499]+A[99:597])/2$

S'': $A[500:999]=B[500:999]+C[500:999]$;

T'' $D[500:999]=(A[500:999]+A[500:1])/2$

③ 不消除联系

```
doall i=1 to 100 do
S': A[3i+2:3i+200, 2:200]=C[i,1:100]*2;
T': D[i,1:100]=A[i+5:i+94, i+1:i+100];
enddoall
```

4 EX4

4.1 ex4.1

1. 针对以下循环 1：

```
for I = 0 to 3 do
    for J = 0 to 3 do
     S A(I,J) = (A(I-1,J) + A(I,J-1)) / 2;
    endfor
endfor
```

1.1 描述以下循环中的存在依赖关系(包括迭代依赖图、依赖类型、依赖向量和距离向量等)

1.2 令 K1 = I + J 和 K2 = J 变换上述循环 1，并给出新循环 2 的迭代依赖图：

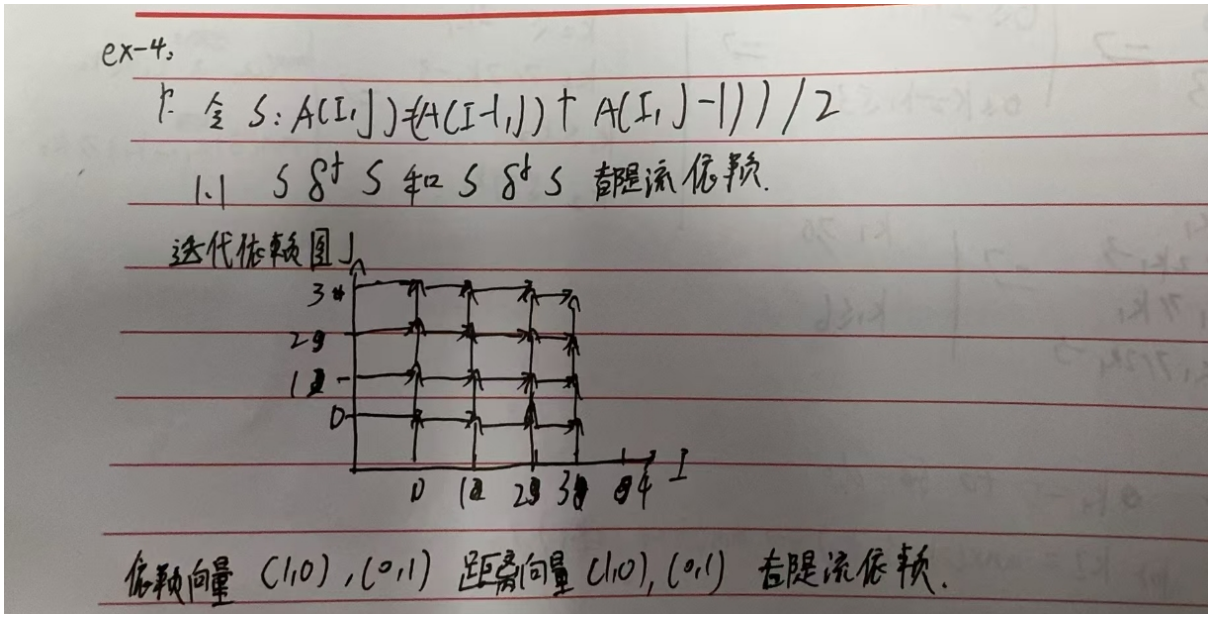
```
for K1 = … …
    for K2 = … …
        A(…,…) = … …
```

1.3 令 K1 = I + J 和 K2 = I + 2*J 变换上述循环 1，并给出新循环 3 的迭代依赖图：

```
for K1 = … …
    for K2 = … …
        A(…,…) = … …
```

1.4 令 K1 = 2*I + J 和 K2 = 3*I + 2*J 变换上述循环 1，并给出新循环 4 的迭代依赖图：

```
for K1 = … …
    for K2 = … …
        A(…,…) = … …
```



1.2. $\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} I \\ J \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$

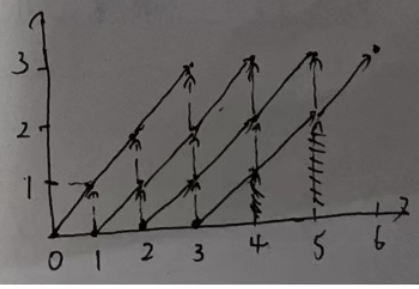
依赖向量为 $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ 均为正

$\begin{cases} 0 \le I \le 3 \\ 0 \le J \le 3 \end{cases} \Rightarrow \begin{cases} 0 \le k_1 \le 3 \\ k_1 - k_2 \\ 0 \le k_2 \le 3 \end{cases} \Rightarrow \begin{cases} k_2 \le k_1 \\ k_2 \ge k_1 - 3 \\ k_2 \ge 0 \\ k_2 \le 3 \end{cases}$

$\max(0, k_1-3) \le k_2 \le \min(3, k_1)$.

$6 \ge k_1 \ge 0$

线上：
```
for k1 = 0 to 6 do
    for k2 = max(0,k1-3) to min(3,k1) do
        i = k1 - k2
        j = k2
        H(i,j)
    endfor
endfor
```



1.3 $\begin{pmatrix} k_1 \\ k_2 \end{pmatrix} = \begin{pmatrix} I \\ J \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$

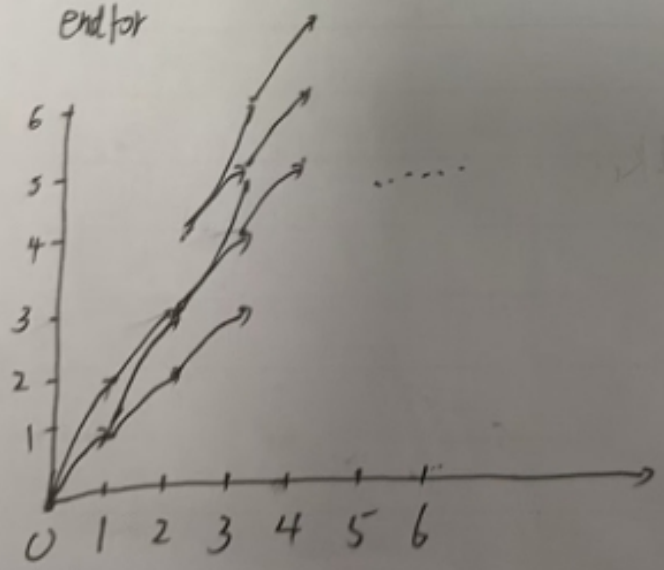
依赖向量: $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$ 均在.

$\begin{cases} 0 \le I \le 3 \\ 0 \le J \le 3 \end{cases} \Rightarrow \begin{cases} 0 \le 2k_1 - k_2 \le 3 \\ 0 \le k_2 - k_1 \le 3 \end{cases} \Rightarrow \begin{cases} k_2 \le 2k_1 \\ k_2 \ge 2k_1 - 3 \\ k_1 \le k_2 \\ k_2 \le 3 + k_1 \end{cases} \Rightarrow \begin{cases} \max(2k_1 - 3, k_1) \le k_2 \\ \min(3 + k_1, 2k_1) \ge k \end{cases}$

$\begin{cases} 2k_1 \ge k_1 \\ 2k_1 \ge 2k_1 - 3 \\ 3 + k_1 \ge k_1 \\ 3 + k_1 \ge 2k_1 - 3 \end{cases} \Rightarrow \begin{cases} k_1 \ge 0 \\ k_1 \le 6 \end{cases}$

综上

```
for k1 = 0 to 6 do
    for k2 = max(2k1-3, k1) to min(3+k1, 2k1) do
        I = 2k1 - k2
        J = k2 - k1
        A[I,J] = (A[I-1,J] + A[I,J-1]) / 2.
    endfor
endfor
```



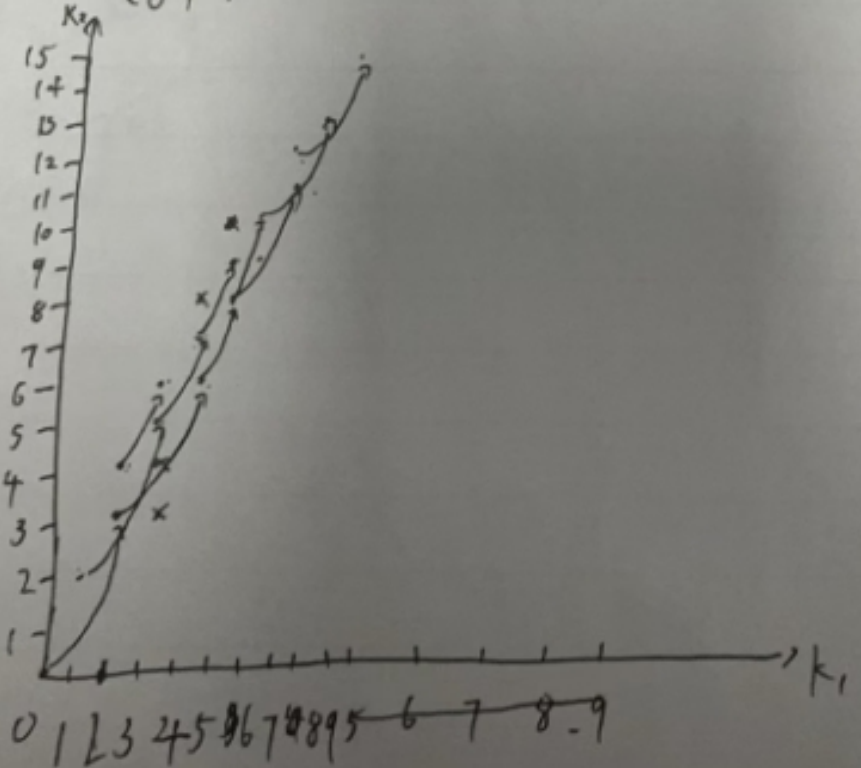
14

$$\begin{cases} |\alpha| \le 3 \\ |\alpha| \le 3 \end{cases} \Rightarrow \begin{cases} 0 \le 2k_1 - k_2 \le 3 \\ 0 \le 2k_2 - 3k_1 \le 3 \end{cases} \Rightarrow \begin{cases} k_2 \le 2k_1 \\ k_2 \ge 2k_1 - 3 \\ k_2 \ge \frac{3}{2}k_1 \\ k_2 \le \frac{3+3k_1}{2} \end{cases}$$

$$\begin{cases} k_2 \le \min(2k_1, \frac{3+3k_1}{2}) \\ k_2 \ge \max(2k_1 - 3, \frac{3}{2}k_1) \end{cases} \quad 0 \le k_1 \le 4$$

```
for k1 = 0 to 4 do
    for k2 = max(2k1 - 3, 3/2 k1) to min(2k1, (3+3k1)/2) do
        I = 2k1 - k2
        J = 2k2 - 3k1
        H(I,J):
    endfor
endfor
```

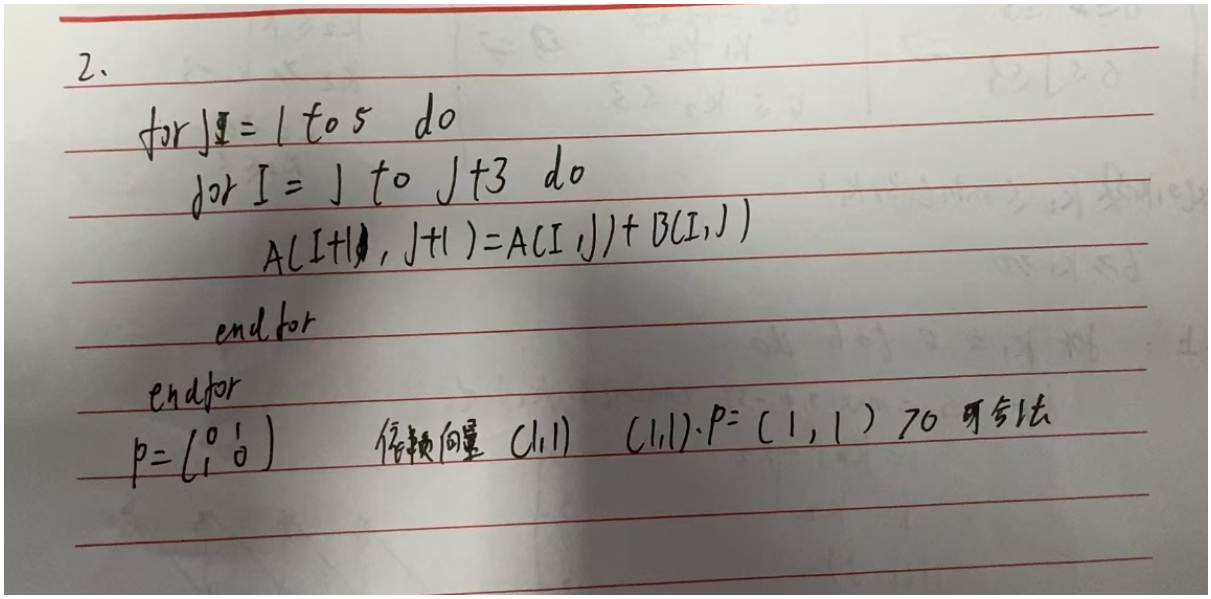
依赖向量为 $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}$



4.2 ex4.2

2. 尝试交换以下循环 5 的内外层循环。

```
for I = 1 to 8 do
    for J = max(I-3,1) to min(I,5) do
     S A(I+1, J+1) = A(I,J) + B(I,J)
    endfor
endfor
```



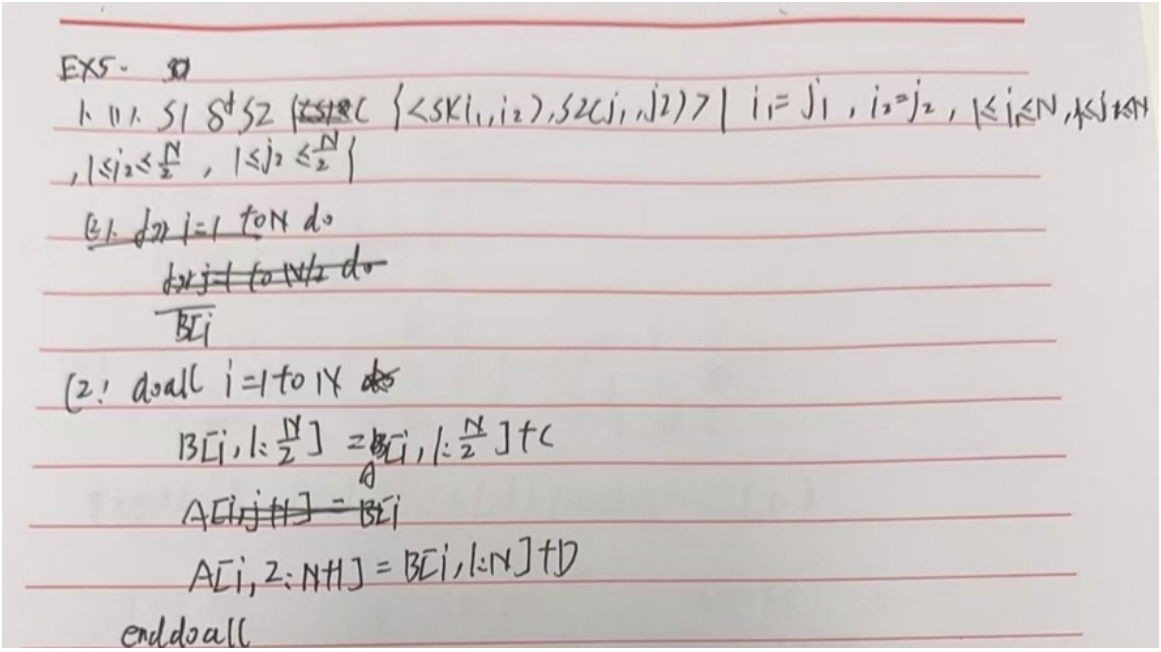
5 EX5

5.1 ex5.1

```
for i = 1 to N do   //循环 1
   for j = 1 to N/2 do
   S1:   B[ i, j ] =  A[ i, j ] + C;
   endfor
   for j = 1 to N do
   S2:   A[ i, j+1 ] =  B[ i, j ] + D;
   endfor
endfor
```

(1) 描述循环中存在的依赖关系。

(2) 尝试对循环 1 进行并行化/向量化

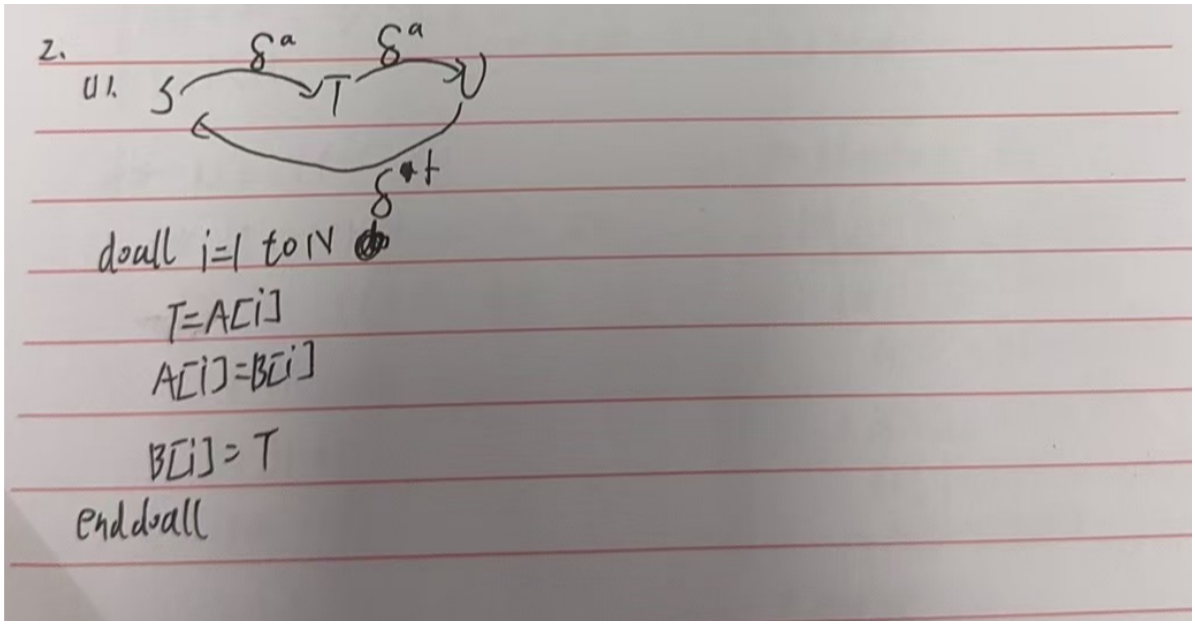


5.2 ex5.2

```
for i = 1 to N do    //循环 2
   S:  T = A[ i ]
   T:  A[ i ] = B[ i ]
   U:  B[ i ] = T
endfor
```

(1) 给出该循环的语句依赖图。

(2) 尝试对循环 2 并行化/向量化。



5.3 ex5.3

```
for k = 1 to L do    //循环 3
    for j = 1 to M do
        for i = 1 to N
S:   A[ i+1,j+1,k+1 ] = A[ i,j,k+1 ] + A[ i,j+1,k ] +
                        A[ i+1,j+1,k ]
        endfor
    endfor
endfor
```

(1)给出循环 3 的依赖方向向量。

(2)令 k1 = k 、j1 = j + k 和 i1 = i 变换如下，

```
    for k1 =  … …
        for j1 =  … …
            for i1 = … …
              A[…,…] = … …
```

- 给出变换后的循环，及其新的依赖方向向量。

- 尝试 OpenMP 并行化变换后的循环。

3.

(1) (0,1,1) (1,0,1) (1,0,0)

(2) $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

等价依赖向量 (0,1,1) (1,1,1) (1,1,0).

$$\begin{cases} 1 \le k \le L \\ 1 \le j \le M \\ 1 \le i \le N \end{cases} \Rightarrow \begin{cases} 1 \le k1 \le L \\ 1 \le j1-k1 \le M \\ 1 \le i1 \le N \end{cases} \Rightarrow \begin{cases} 1 \le k1 \le L \\ 1+k1 \le j1 \le M+k1 \\ 1 \le i1 \le N \end{cases}$$

```
for k1 = 1 to L do
  for j1 = 1+k1 to M+k1 do
    for i1 = 1 to N do
      k = k1
      j = j1-k1
      i = i1
      H(i,j,k)
    endfor
  endfor
endfor
```

(2)

```
for k1 = 1 to L do
  for j1 = 1+k1 to M+k1 do
    # parallel for
    for i1 = 1 to N do
      A[i1+1][j1-k1][k1+1]
      = A[i1][j1-k1][k1+1]
      + A[i1][j1+1-k1][k1]
      + A[i1+1][j1+1-k1][k1]
    endfor
  endfor
endfor
```