

[문제 1] Pairwise interaction

가상의 3차원 공간 $\{x, y, z; 0 < x, y, z < 100\}$ 에 N 개의 입자가 분포되어 있다. 각 입자는 자기 자신을 제외한 모든 입자와 상호작용을 하며 두 입자 $R_i = (x_i, y_i, z_i), R_j = (x_j, y_j, z_j)$ 의 상호작용력은 아래와 같이 정의 한다.

$$F(R_i, R_j) = \frac{R_j - R_i}{\|R_j - R_i\|^3}$$
$$F_x(R_i, R_j) = \frac{x_j - x_i}{\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right)^3}$$
$$F_y(R_i, R_j) = \frac{y_j - y_i}{\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right)^3}$$
$$F_z(R_i, R_j) = \frac{z_j - z_i}{\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right)^3}$$

또한 총 작용하는 합력은 아래와 같이 정의한다.

$$f_i(f_{xi}, f_{yi}, f_{zi}) = \sum_{j=1}^N F(R_i, R_j), i \neq j$$

순차 코드를 참조하여 주어진 입자들의 좌표를 가지고, 각각의 입자가 받는 합력을 구하는 최적화된 병렬코드 (OpenMP 또는 MPI)를 작성하시오. ($N=100,000$, 각 입자의 좌표 값으로 "position.txt"를 입력값으로 사용한다.)

순차 코드 (C)

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#include<math.h>
#define PNUM 100000

int main()
{
    int i,j;
    float fx[PNUM],fy[PNUM],fz[PNUM], x[PNUM], y[PNUM], z[PNUM];
    float r3, bufx, bufy, bufz;
    FILE *fp;
    fp=fopen("./position.txt","r");
    for(i=0;i<PNUM;i++)
    {
        fscanf(fp,"%f %f %f",&bufx,&bufy,&bufz);
        x[i]=bufx;
        y[i]=bufy;
        z[i]=bufz;
    }
    fclose(fp);
    for(i=0;i<PNUM;i++)
    {
        fx[i]=0.0;fy[i]=0.0;fz[i]=0.0;
        for(j=0;j<PNUM;j++)
        {
            if(i==j) continue;
            r3=1.0/((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j])+(z[i]-z[j])*(z[i]-z[j]));
            r3*=sqrt(r3);
            fx[i]+=(x[j]-x[i])*r3;
            fy[i]+=(y[j]-y[i])*r3;
            fz[i]+=(z[j]-z[i])*r3;
        }
    }
    for(i=0;i<PNUM;i++)
    {
        printf("%f\t%f\t%f\n",fx[i],fy[i],fz[i]);
    }
    return 0;
}
```

순차코드 (Fortran)

```
program pairwise

implicit none

integer, parameter :: PNUM=100000
integer :: i, j
real, dimension(PNUM) :: fx, fy, fz, x, y, z
real :: r3

open(10, FILE='./position.txt', STATUS='OLD')
do i=1,PNUM
    read(10,*) x(i),y(i),z(i)
enddo
close(10)

fx=0.0;fy=0.0;fz=0.0
do i=1,PNUM
    do j=1,PNUM
        if(i==j) cycle
        r3=1.0/((x(i)-x(j))*(x(i)-x(j))+(y(i)-y(j))*(y(i)-y(j))+(z(i)-z(j))*(z(i)-z(j)))
        r3=r3*sqrt(r3)
        fx(i)=fx(i)+(x(j)-x(i))*r3
        fy(i)=fy(i)+(y(j)-y(i))*r3
        fz(i)=fz(i)+(z(j)-z(i))*r3
    enddo
enddo

do i=1,PNUM
    write(*,*) fx(i), fy(i), fz(i)
enddo

end program pairwise
```

[문제2] Jacobi iterative solver

$Ax = b$ 와 같은 선형 시스템을 풀기위한 Jacobi iterative algorithm은 아래와 같다.

먼저 A 를 대각 성분 행렬 D 와 대각 성분 아랫부분 행렬 L , 대각 성분 윗부분 행렬 U 로 분해하면, 행렬 A 는 다음과 같이 표현된다.

$$A = L + D + U$$

반복 횟수를 n 으로 표시하면 벡터 x 는 다음과 같이 구할 수 있다.

$$Dx^n = b - (L + U)x^n$$

여기서 $D^{-1} = \frac{1}{D}$ 이다. 프로그램 작성을 위한 알고리즘은 다음과 같다.

■ 알고리즘

```
set  $x_{old} = \text{initial guess value}$ 
for loop(until max iteration)
    {
        set  $sum_1 = sum_2 = 0.0$ 
         $Lx: sum_1 = sum_1 + A[i][j]x_{old}[j]$     for  $0 \leq j < i$ 
         $Ux: sum_2 = sum_2 + A[i][j]x_{old}[j]$     for  $i+1 \leq j < N$ 
         $x[i] = (b[i] - sum_1 - sum_2) / A[i][i]$ 
        calculate residual
        if ( $residual \leq tolerance$ ) exit
        set  $x_{old} = x$ 
    }
```

아래의 순차 코드를 참고하여, 행렬 크기를 500×500 으로 설정하고, 행렬 A 와 벡터 b 를 아래와 같이 설정한 후 병렬 프로그램을 작성하시오.

$$A = \begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 1 & -2 \end{pmatrix}, \quad b \text{는 } b = [1, 2, \dots, N]$$

수렴 조건을 만족시키기 위한 tolerance값은 $10e-10$ 으로 설정하고, 해를 수렴시키기 위한 최대 반복 횟수는 10000000(수정 가능)으로 설정한다. x 의 초기 추정값(initial guess value)은 모두 1로 설정한다.

병렬화 힌트 : 위의 알고리즘에서 Lx , Ux 를 계산함에 있어서 프로세서 사이에 의존성이 없으므로 sum_1 , sum_2 , x_{old} 값을 프로세서별로 동시에 계산할 수 있다.

	A	x
P1		
P2	L U	
P3		

1. 순차코드[C]

```
#include <stdio.h>
#include <stdlib.h>

int Jacobi(int N, double** A, double *x, double *b, double* res, double abstol)
{
    int i,j,k;
    long maxiters=10000000;
    double sum1, sum2;
    double *xold=(double*)malloc(sizeof(double)*N);

    // set initial guess
    for(i=0;i<N;i++){
        xold[i]=1.0;
    }

    for(k=0;k<maxiters;k++){
        for(i=0;i<N;i++){
            sum1=0.0;    sum2=0.0;
            for(j=0;j<i;j++)
                sum1=sum1+A[i][j]*xold[j];
            for(j=i+1;j<N;j++)
                sum2=sum2+A[i][j]*xold[j];

            x[i]=(-sum1-sum2 +b[i])/A[i][i];
        }
        *res=0.0;
        for(i=0;i<N;i++){
            *res=*res+(x[i]-xold[i])*(x[i]-xold[i]);
        }
        if(sqrt(*res)<abstol){
            free(xold);
            return k;
        }
        for(i=0;i<N;i++)
            xold[i]=x[i];
    }

    printf("Jacobi: Maximum Number of Iterations Reached Without Convergence\n");
    return k;
    free(xold);
}

int main(void)
{
    int i,j;
    int iters;
    int const N=500;
```

```

double **A, *x, *q;
double res=10e10;

x=(double*)malloc(sizeof(double)*N);
q=(double*)malloc(sizeof(double)*N);

A=(double**)malloc(sizeof(*A)*N);
for(i=0;i<N;i++) A[i]=(double*)malloc(sizeof(double)*N);

for(i=0;i<N;i++){
    q[i]=i+1;
    for(j=0;j<N;j++){
        A[i][j]=0.0;
    }
}
for(i=0;i<N;i++){
    A[i][i]=-2.0;
    if(i<N-1){
        A[i][i+1]=1.0;
        A[i+1][i]=1.0;
    }
}
iters=Jacobi(N,A,x,q,&res,10e-10);

printf("iters=%d, residual norm=%lf\n",iters,res);

for(i=0;i<N;i++) free(A[i]);
free(A);
free(x);
free(q);
return 0;
}

```

2. 순차코드[Fortran]

```
PROGRAM JACOBI_PROBLEM
IMPLICIT NONE
INTEGER(KIND=4)::I
INTEGER(KIND=8)::ITERS
INTEGER(KIND=8),PARAMETER::N=500
REAL(KIND=8)::A(N,N),X(N),Q(N)
REAL(kind=8)::RES=10d10

A=0.0
Q=0.0;   X=0.0

DO I=1,N
  Q(I)=I
  A(I,I)=-2.0
  IF(I<N)THEN
    A(I,I+1)=1.0;
    A(I+1,I)=1.0;
  END IF
END DO

CALL JACOBI(N,A,X,Q,RES,ITERS,10d-10)
PRINT*, 'ITERS=',ITERS,'RESIDUAL NORM=',RES

END PROGRAM JACOBI_PROBLEM

SUBROUTINE JACOBI(N,A,X,B,RES,ITERS,ABSTOL)
IMPLICIT NONE
INTEGER(KIND=8)::N
INTEGER(KIND=8)::ITERS
REAL(KIND=8)::A(N,N),X(N),B(N)
REAL(KIND=8)::RES
REAL(KIND=8)::ABSTOL

INTEGER(KIND=8)::I,J,K
INTEGER(KIND=8),PARAMETER::MAXITERS=10000000
!INTEGER(KIND=4),PARAMETER::MAXITERS=10000000
REAL(KIND=8)::SUM1, SUM2
REAL(KIND=8)::XOLD(N)
LOGICAL::CONVERGE=.FALSE.
XOLD=1.0
```



```

DO K=1,MAXITERS
  DO I=1,N
    SUM1=0.0;    SUM2=0.0
    DO J=1,I-1
      SUM1=SUM1+A(I,J)*XOLD(J)
    END DO
    DO J=I+1,N
      SUM2=SUM2+A(I,J)*XOLD(J)
    END DO
    X(I)=(-SUM1-SUM2+B(I))/A(I,I)
  END DO

  RES=0.0
  DO I=1,N
    RES=RES+(X(I)-XOLD(I))*(X(I)-XOLD(I))
  END DO

  IF(SQRT(RES)<=ABSTOL)THEN
    ITERS=K
    CONVERGE=.TRUE.
    RES=SQRT(RES)
    EXIT
  END IF
  XOLD=X
END DO
IF(CONVERGE==.FALSE.)THEN
  PRINT*,"Jacobi: Maximum Number of Iterations Reached Without Convergence"
  ITERS=MAXITERS
  RES=SQRT(RES)
END IF
END SUBROUTINE JACOBI

```