a2/FW/fw_sr_p.c

```c
/*
 * Recursive implementation of the Floyd-Warshall algorithm.
 * command line arguments: N, B
 * N = size of graph
 * B = size of submatrix when recursion stops
 * works only for N, B = 2^k
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <omp.h>
#include "util.h"

inline int min(int a, int b);
void FW_SR (int **A, int arow, int acol,
            int **B, int brow, int bcol,
            int **C, int crow, int ccol,
            int myN, int bsize);

int main(int argc, char **argv)
{
    int **A;
    int i,j,k;
    struct timeval t1, t2;
    double time;
    int B=16;
    int N=1024;

    if (argc !=3){
        fprintf(stdout, "Usage %s N B \n", argv[0]);
        exit(0);
    }

    N=atoi(argv[1]);
    B=atoi(argv[2]);

    if ((N%B)!=0){
        fprintf(stdout, "N must be multiple of B\n");
        exit(0);
    }

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

    graph_init_random(A,-1,N,128*N);

//------------------------------------------------------------------------
    gettimeofday(&t1,0);

    #pragma omp parallel
    #pragma omp single
    {
    FW_SR(A,0,0, A,0,0,A,0,0,N,B);
    }

    gettimeofday(&t2,0);

    time=(double)((t2.tv_sec-t1.tv_sec)*1000000+t2.tv_usec-t1.tv_usec)/1000000;
    printf("FW_SR,%d,%d,%.4f\n", N, B, time);


//  for(i=0; i<N; i++)
//      for(j=0; j<N; j++) fprintf(stdout,"%d\n", A[i][j]);


    return 0;
}

inline int min(int a, int b)
{
    if(a<=b)return a;
    else return b;
}

void FW_SR (int **A, int arow, int acol,
            int **B, int brow, int bcol,
            int **C, int crow, int ccol,
            int myN, int bsize)
{
    int k,i,j;
    /*we use different task paral depending on the blocks A,B,C use.
    If they use same blocks therre may be future depedencies , else not*/

    //Arrays check
    if (A!=B || A!=C){
        printf("Different arrays not supported yet\n");
        exit (1);
    }
    //row check
    int RAB= arow==brow;
    int RAC= arow==crow;
    int RBC= brow==crow;
    //col check
    int CAB= acol==bcol;
    int CAC= acol==ccol;
    int CBC= bcol==ccol;
    //case check
    int case_id;
    if (RAB&&RAC&&CAB&&CAC) case_id=0; //A,B,C same block
    else if (RAB&&CAB) case_id=1; //A,B same block
    else if (RAC&&CAC) case_id=2; //A,C same block
    else case_id=3; //A separate from B and C

    /*
```

```c
     * The base case (when recursion stops) is not allowed to be edited!
     * What you can do is try different block sizes.
     */
    if(myN<=bsize)
        for(k=0; k<myN; k++)
            for(i=0; i<myN; i++)
                for(j=0; j<myN; j++)
                    A[arow+i][acol+j]=min(A[arow+i][acol+j], B[brow+i][bcol+k]+C[crow+k][ccol+j]);
    else {

        switch(case_id){
            case 0: //A,B,C same block
            {
                //call1
                FW_SR(A,arow, acol,B,brow, bcol,C,crow, ccol, myN/2, bsize);

                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call2
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                }
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call3
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol,C,crow, ccol, myN/2, bsize);
                }
                #pragma omp taskwait

                //call4
                FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol,C,crow, ccol+myN/2, myN/2, bsize);

                //call5
                FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);

                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call6
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
                }
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call7
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
                }
                #pragma omp taskwait

                //call8
                FW_SR(A,arow, acol,B,brow, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
            }
            break;
            case 1: //A,B same block
            {
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call1
                    FW_SR(A,arow, acol,B,brow, bcol,C,crow, ccol, myN/2, bsize);
                    //call2
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                    //call7
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
                    //call8
                    FW_SR(A,arow, acol,B,brow, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
                }
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call3
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol,C,crow, ccol, myN/2, bsize);
                    //call4
                    FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                    //call5
                    FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
                    //call6
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
                }

                #pragma omp taskwait
            }
            break;
            case 2: //A,C same block
            {
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call1
                    FW_SR(A,arow, acol,B,brow, bcol,C,crow, ccol, myN/2, bsize);
                    //call3
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol,C,crow, ccol, myN/2, bsize);
                    //call6
                    FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
                    //call8
                    FW_SR(A,arow, acol,B,brow, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
                }
                #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
                {
                    //call2
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                    //call4
                    FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                    //call5
                    FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
                    //call7
                    FW_SR(A,arow, acol+myN/2,B,brow, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
                }

                #pragma omp taskwait
            }
            break;
            case 3: //A separate from B and C
            #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
            {
                //call1
                FW_SR(A,arow, acol,B,brow, bcol,C,crow, ccol, myN/2, bsize);
                //call8
                FW_SR(A,arow, acol,B,brow, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);

            }
            #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
            {
                //call2
                FW_SR(A,arow, acol+myN/2,B,brow, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                //call7
                FW_SR(A,arow, acol+myN/2,B,brow, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
            }
            #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
            {
                //call3
                FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol,C,crow, ccol, myN/2, bsize);
                //call6
                FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
            }
            #pragma omp task firstprivate(arow,acol,brow,bcol,crow,ccol,myN,bsize) shared(A,B,C)
            {
                //call4
                FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol,C,crow, ccol+myN/2, myN/2, bsize);
                //call5
                FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
            }
            #pragma omp taskwait

            break;
        }
    }
}

/*
call1
        FW_SR(A,arow, acol,B,brow, bcol,C,crow, ccol, myN/2, bsize);
call2
        FW_SR(A,arow, acol+myN/2,B,brow, bcol,C,crow, ccol+myN/2, myN/2, bsize);
call3
        FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol,C,crow, ccol, myN/2, bsize);
call4
        FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol,C,crow, ccol+myN/2, myN/2, bsize);
call5
        FW_SR(A,arow+myN/2, acol+myN/2,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
call6
        FW_SR(A,arow+myN/2, acol,B,brow+myN/2, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
call7
        FW_SR(A,arow, acol+myN/2,B,brow, bcol+myN/2,C,crow+myN/2, ccol+myN/2, myN/2, bsize);
call8
        FW_SR(A,arow, acol,B,brow, bcol+myN/2,C,crow+myN/2, ccol, myN/2, bsize);
*/



```