

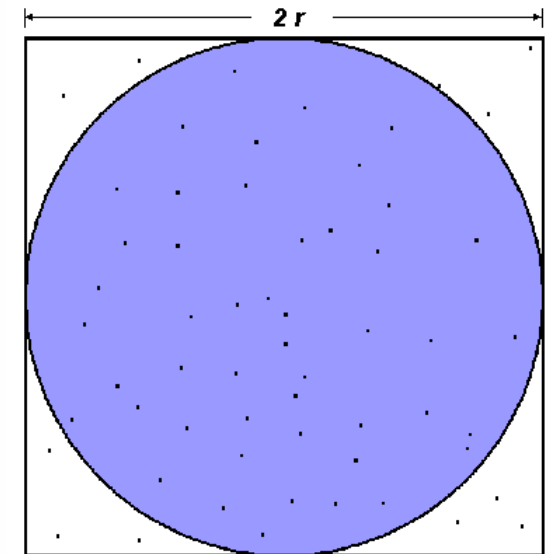


# Esercizi MPI

Algoritmi e Calcolo Parallelo

- Implementare in MPI una soluzione parallela del seguente algoritmo per approssimare PI

```
scanf("%d",&npoints); count = 0;
for(j=0, j<npoints; j++) {
    x = random();
    y = random();
    if (inCircle(x, y))
        count++;
}
PI = 4.0*count/npoints
```



$$A_S = (2r)^2 = 4r^2$$
$$A_C = \pi r^2$$
$$\pi = 4 \times \frac{A_C}{A_S}$$

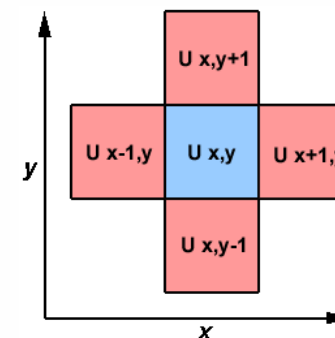
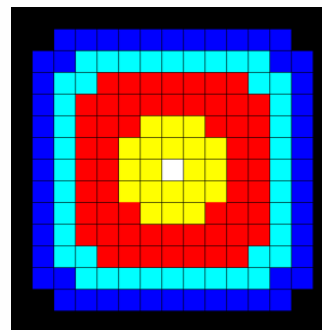
- Non è necessario implementare la funzione `inCircle()`
- Si implementi
  - ▶ una soluzione senza utilizzare le funzioni di comunicazione collettive
  - ▶ una soluzione in cui vengono utilizzate le funzioni di comunicazione collettive

- ❑ Parallelizzare in MPI il seguente algoritmo che
  - ▶ conta i numeri primi inferiori a LIMIT
  - ▶ ritorna il numero primo più grande trovato

```
pc=4;    /* 2,3,5,7 are counted here */
for (n=11; n<=LIMIT; n=n+2) {
    if (isprime(n)) {
        pc++;
        foundone = n;
    }
}
printf("found= %d, largest=%d\n", pc, foundone);
```

- Proporre una soluzione MPI per parallelizzare su una macchina con M processori il calcolo di una semplice equazione di calore su una matrice NxN:

$$\begin{aligned}
 U_{x,y} &= U_{x,y} \\
 &+ C_x * (U_{x+1,y} + U_{x-1,y} - 2 * U_{xy}) \\
 &+ C_y * (U_{x,y+1} + U_{x,y-1} - 2 * U_{x,y})
 \end{aligned}$$



- La soluzione seriale del problema può essere schematizzata come:

```

for (x=1; x < N-1; x++)
  for (y=1; y < N-1; y++)
    u2[x][y] = u[x][y] + cx*(u[x+1][y] + u[x-1][y]
    - 2* u[x][y]) + cy*(u[x][y+1] + u[x][y-1]
    - 2* u[x][y])
  
```