PD109-1 hw5-1

Given a two dimension array map, which store the number of people living on location (x, y)

Given (n, m) as the boundary of (x, y) in the map

Given r as the range of coverage of a built hospital

```
// main function
int findMaxCover($n$, $m$, $r$, $map$)
    initialize the number of maximum cover as $maxCover$ to be 0
    for $u$ from 0 to $n$
        for $v$ from 0 to $m$
            set $numCover$ = cover($n$, $m$, $u$, $v$, $r$, $map$)
            if $numCover > maxCover$
                update $maxCover$ to be $numCover$
    return $maxCover$
```

For the function cover(), we have two algorithms as following:

Algorithm 1

```
int cover($n$, $m$, $u$, $v$, $r$, $map$)
    initialize the number of people covered by the hospital on $(u, v)$ as
    $numOfCovered$ to be 0
    for $i$ from 0 to $n$
        for $j$ from 0 to $m$
            set $distance = |i - u| + |j - v|$
            if $distance \leq r$
                update $numOfCovered$ to $numOfCovered + map[i][j]$
    return $numOfCovered$
```

Algorithm 2

```
int cover($n,\ m,\ u,\ v,\ r,\ map$)
    initialize the number of people covered by the hospital on $(u, v)$ as
    $numOfCovered$ to be 0
    for $i$ from $u - r$ to $u + r$
        set $yRange$ as $r - |i - u|$
        for $j$ from $v - yRange$ to $v + yRange$
            update $numOfCovered$ to $numOfCovered + map[i][j]$
    return $numOfCovered$
```

In algorithm 1, for each location, we have to compute $(n+1) * (m+1)$ times to compute the number of covered people. In algorithm 2, if $r = 1$, then we have to compute 1+3+1 times; if $r = 2$, then we have to compute 1 + 3 + 5 + 3 + 1 times. Thus, for a given range r, we have to compute $2\sum_{i=1}^{r}(2i-1) + (2r+1) = 2r(r+1) + 1$ times. Theoretically, we can formulate the inequality as following to find the condition such that one of the algorithm is more efficient than the other one:

$$2r(r+1) + 1 \leq (n+1)(m+1)$$

However, in practice, we only need to consider those villages inside of the map. When we implement these two algorithms, obviously the range to compute in algorithm 2 is less or equal than the range to compute in algorithm 1. Thus, algorithm 2 is more efficient.