CS 4820, Spring 2018	**Homework 5, Problem 2**
Name: Rongguang Wang	
NetID: rw564	
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)	

(2) *(10 points)* Solve Exercise 5.7 in Kleinberg & Tardos. Suppose now that you're given an $n \times n$ grid graph G. (An $n \times n$ grid graph is just the adjacency graph of an $n \times n$ chessboard. To be completely precise, it is a graph whose node set is the set of all ordered pairs of natural numbers (i, j), where $1 \le i \le n$ and $1 \le j \le n$; the nodes (i, j) and (k, l) are joined by an edge if and only if $|i - k| + |j - l| = 1$.) We use some of the terminology of the previous question. Again, each node v is labeled by a real number x_v; you may assume that all these labels are distinct. Show how to find a local minimum of G using only $O(n)$ probes to the nodes of G. (Note that G has n^2 nodes.)

Solution:

The problem is aiming to design an algorithm which can find a local minimum of a grid G using only $O(n)$ probes to the nodes of G. The size of the grid G is $n \times n$ and the value of each node $v_{i,j}$ is $x_{i,j}$. All the value of the nodes are distinct. Let $v_{i,j}$ be the nodes which corresponds to the ordered pair (i, j) in the grid. Define B as the subset of nodes on the boundary of the grid, which means that subset B contains $B = \{e_{1,j} \forall j = 1 \dots n\} \cup \{e_{n,j} \forall j = 1 \dots n\} \cup \{e_{j,1} \forall j = 1 \dots n\} \cup \{e_{j,n} \forall j = 1 \dots n\}$. The neighbors of the elements in subset B can be divided into two catalogue: one node which is not in B and nodes in B. Let B_{min} be the minimum node in the boundary subset B and v be the node to which B_{min} connected to such that $v \notin B$. Node v can be called the inner neighbor of B_{min}. If $B_{min} < v$, B_{min} is the local minimum. Since all values of the node are distinct, B_{min} cannot equals to v. If $B_{min} > v$, there must be a local minimum within a reduced graph G with node subset $(G - B)$.

Now, define M as a subset of nodes on the boundary of the grid G plus the nodes on row $n/2$ and on column $n/2$, which is $M = \{e_{1,j} \forall j = 1 \dots n\} \cup \{e_{n,j} \forall j = 1 \dots n\} \cup \{e_{j,1} \forall j = 1 \dots n\} \cup \{e_{j,n} \forall j = 1 \dots n\} \cup \{e_{j,n/2} \forall j = 1 \dots n\} \cup \{e_{n/2,j} \forall j = 1 \dots n\}$. The grid is then be divided into four quadrants by the two central lines. Considering whether the central node $(n/2, n/2)$ of the grid is a local minimum by checking its four neighbors value. If the central node is less than any of its neighbors, it is the local minimum. Otherwise, continuously find the local minimum in subset M by searching the minimum node M_{min}. the minimum node may lies on either the outer boundary or the row $n/2$ or column $n/2$. If M_{min} lies on the outer boundary B, its inner grid neighbor (not on boundary B) x should lie in one quadrant Q of the four. If $x > M_{min}$, M_{min} is the local minimum. Otherwise, there must be a local minimum within the quadrant x lies in. Then, the local minimum can be found by apply the algorithm discussed above recursively on the sub-grid Q with size $n/2 \times n/2$. If M_{min} lies on row $n/2$ or column $n/2$ but not on the central node $(n/2, n/2)$ and the boundary nodes B, two neighbors x_i and x_j, which are not in subset M, of it should be lie in two different quadrant Q_i and Q_j. If $x_i > M_min$ and $x_j > M_{min}$, M_{min} is the local minimum. Otherwise, choose either quadrant Q_i or Q_j when x_i or x_j is smaller than M_{min}. There must be a local minimum within the quadrant selected. Then, the local minimum can be found by apply the algorithm discussed above recursively on the sub-grid Q with size $n/2 \times n/2$.

Algorithm:

```
Given grid G,

function LocalMin(G):

  find the minimum node B_min in boundary subset B

    if the neighbor node (not belongs to B) of B_min is larger than B_min
      return node B_min

    else if
    the neighbor nodes of the central node C (n/2, n/2) of G are all larger than C
      return node C

    else
      find the minimum node M_min in subset M (including B and row n/2 and column n/2)

        if M_min lies in the boundary subset B

          if the neighbor node x (not belongs to B) of M_min is larger than M_min
            return node M_min

          else
            call function LocalMin(subset of quadrant that x lies in)

        if M_min lies in (M - B) but not the central node C

          if the neighbor nodes x_i and x_j (not belongs to (M-B)) of M_min is
          both larger than M_min
            return node M_min

          else
            call function LocalMin(subset of quadrant that x_i or x_j lies in if
            x_i < M_min or x_j < M_min)

  return null
```

Algorithm's Correctness:

Lemma 1. *If $B_{min} > x$, there must be a local minimum within a reduced grid G with node subset G-B.*

Proof. Proof by induction.

Hypothesis:
The statement holds for any n.
Base Case:
When $n = 3$, any grid for $n < 3$ would not have any node inside if the boundary nodes were removed. Let B_{min} be the minimum node on the boundary B. Only node x left after remove the boundary B

which is grid G. If $x < B_min$, x is the local minimum since it is smaller than all its neighbors.
<u>Induction Step:</u>
Considering a grid G with size $(n+1) \times (n+1)$. Let B_{min} be the minimum node on the boundary B. Let x be the inner neighbor of B_min that has neighbors a, b and c addition to B_{min}. If x is less than a, b and c, x is the local minimum since $x < B_{min}$ was given. Therefore, the induction holds. □

Lemma 2. *Every grid G who has distinct values for its nodes would has a local minimum.*

Proof. If $B_{min} < x$, B_{min} is the local minimum. Otherwise, a local minimum can be find in grid G according to Lemma 1. □

Running Time Analysis:

There are n^2 nodes in the given grid G. The number of nodes along each edge is n. $O(n)$ time is taken for create the subset M and it costs $O(n)$ time in finding the M_{min} using the value probing. By using Divide and Conquer Algorithm, the problem of size $n \times n$ was divided into sub-problem of size $n/4 \times n/4$ since one quadrant was chosen out of four each iteration. Therefore, the recurrence is,

$$T(n) = T(\frac{n}{4}) + O(n)$$

which is $T(n) = O(n)$.