**P2**

---

**Prompt:**

Given a square n x n table, design an algorithm that runs in time polynomial in n and outputs the minimum number of extra squares to remove from the table such that we cannot put any domino in the remaining table.

**Algorithm:**

```
Input: A square n x n table T meeting the given description
      with "X"s on some of the cells
Output: The minimum number of squares that can be removed from T
       such that no domino can be placed on T
MinSquares(T):
    Let X be an empty set
    Let Y be an empty set
    Let E be an empty set
    Color the cells of T in black and white checkerboard pattern
    For each cell c in T do
        If c is black then
            Add a vertex x corresponding to c to X
        Else then
            Add a vertex y corresponding to c to Y
        EndIf
    EndFor
    For each cell c in T do
        If there is no "X" on c then
            For each cell n adjacent to c do
                If there is no "X" on n then
                    Let v be the vertex from X or Y corresponding to c
                    Let u be the vertex from Y or X corresponding to n
                    If there isn't an edge in E between v and u
                        Let e be an edge between v and u
                        Add e to E
                    EndIf
                EndIf
            EndFor
        EndIf
    EndFor
    Let graph G = (X, Y, E)
    Let flow network H = FlowNetwork(G)
    Let min s-t cut C = FFA(H)
    Let min vertex cover S = VertexCover(H, C)
    Output |S|
```

---

**Claim:**

The algorithm terminates in time polynomial in n.

**Proof:**

We know from HW8 P1 that FlowNetwork, FFA, and VertexCover all terminate in polynomial time. Therefore, that MinSquares terminates is evident given that every for loop iteration over the cells of T is over finite and monotonically decreasing sets. The first outer for loop iterates over each cell for $O(n^2)$ runtime while the second outer for loop iterates over each cell times each cell's 4 neighbors for $O(n^2)$ runtime. $O(n^2)$ is the dominant runtime factor. Thus, MinSquares terminates in time polynomial in n.

---

**Claim:**

Given a table T, the algorithm outputs the minimum number of squares to remove from T such that no domino can be placed on T.

**Proof:**

```
Let T be colored in black and white checkerboard pattern

The graph G = (X, Y, E) is bipartite
    G is divided into sets X and Y
        every vertex x in X corresponds to a black cell in T and
        every vertex y in Y correspondes to a white cell in T
        there is no vertex v in both X and Y
        thus G is divided into sets X and Y
    Every edge e is between some x in X and some y in Y
        let v be a vertex in X or Y corresponding to a cell c in T
        let u be a vertex in X or Y corresponding a cell n adjacent to c
        let e be between v and u
        since no two black cells nor two white cells are adjacent
        there are no edges between any xi and xj in X
        and there are no edges between any yi and yj in Y
        thus every edge in E is between some x in X and some y in Y
    Thus G is bipartite

Every edge in G represents a valid domino placement in T
    an edge is added to G iff there are two available adjacent cells in T
    a valid domino placement requires two adjacent cells in T
    Thus an edge in G represents a valid domino placement in T
```

```
Let flow network H = FlowNetwork(G)
H is a flow network of G
    We've shown in lecture 23 that H is a flow network of G

Let min s-t cut C = FFA(H)
C is the min s-t cut of H
    We've shown in lecture 23 that Ford-Fulkerson produces min s-t cut

Let min vertex cover S = VertexCover(H, C)
S is the min vertex cover of H
    We've shown in P1 that VertexCover produces the min vertex cover of H

G \ S removes all edges from G
    for any vertex v in G
    removing v from G removes the edges incident to v from G
    S is a vertex cover
    therefore every edge in G has at least one vertex in S
    thus G \ S removes all edges from G

|S| is a number of cells such that no domino can be placed on T
    Every edge in G represents a valid domino placement in T
    G \ S removes all edges from G
    So G \ S represents T with no valid domino placements
    Thus |S| is a number of cells such that no domino can be placed on T

Let K be the minimum set of cells that can be removed from T
such that no domino can be placed on T

K corresponds to a vertex cover
    Every cell c in T is represented in X or Y
    Every valid domino placement in T is represented as an edge in E
    every k in K is a cell c in T
    so every k in K
        is represented as a vertex in X or Y and
        represents removing ≥ 1 valid domino placement from T
        and so represents removing ≥ 1 edge from E
    By definition
        K is the minimum set of cells to remove all valid domino positions
        so T \ K removes all valid domino positions from T
    So T \ K is equivalent to G \ S
    Thus K corresponds to a vertex cover

|S| = |K|
    Let |min number cells| denote the optimum solution to T

    |min vertex cover| ≤ |K|
        suppose S is the minimum vertex cover of H
        then |S| = |min vertex cover|
        Then there are two options
            either |S| = |K|
            or |S| != |K|
```

```
    If |S| = |K| then
        |K| = |min vertex cover|
        and since K is the number of cells to remove from T
        |min vertex cover| = |min number cells|
    Else if |S| != |K| then
        since K represent number of cells to remove from T
        it must be that |S| < |K|
            since it can't be that |min vertex cover| > |K|
            so then |min vertex cover| < |K|
    So since |S| ≤ |K|
    and since |S| = |min vertex cover|
    Thus |min vertex cover| ≤ |K|

|min number cells| ≤ |S|
    suppose K is the minimum number of cells to remove from T
    then |K| = |min number cells|
    Then there are two options
        either |K| = |S|
        or |K| != |S|
    If |K| = |S| then
        |S| = |min number cells|
        and since S is the vertex cover of G
        |min number cells| = |min vertex cover|
    Else if |K| != |S| then
        since S represents the vertex cover of G
        it must be that |K| < |S|
            since it can't be that |min number cells| > |S|
            so then |min number cells| < |S|
    So since |K| ≤ |S|
    and since |K| = |min number cells|
    Thus |min number cells| ≤ |S|

So |S| = |min vertex cover| and
|K| = |min number cells| and
|min vertex cover| ≤ |K| and
|min number cells| ≤ |S|
Then it must be that
    |min vertex cover| = |min number cells| and
    |S| = |K|
Thus |S| = |K|
```