Solution 1.

We can view the problem as a Bipartite problem of finding the maximum matches, that will correspond to the maximum number of dominos that can be placed over the chessboard. As on chessboard a 2X1 domino will always occupy a black and a white position, we first divide the chessboard into two groups, white cells and black cells, for every white cell we check all the adjacent black cells and connect if there is a 0(connection means we can put a domino over a pair of adjacent white black cell) and vice versa. we connect white to black cells with forward edges.

Then we add a source with forward edges to all white cells and sink with all black edges having a forward edge to sink.
Then we ran the Bipartite algorithm to find the maximum match possible.
If the value of max match*2 is less than the number of zeros, we have some uncovered position left and the algorithm outputs NO.
If it is equal the algorithm outputs YES.(obviously it cannot output more).

Running time would be $O((nm)^3)$ as the graph will have nm vertices and maximum of (n-1)m + (m-1)n or 2mn edges
Now running the Edmond karp's to get the maximum match of the bipartite graph is $O((V+E)(VE)) = O((nm+nm)(n^2m^2))= O(n^3m^3)$.

Algorithm:

1. We partition Black and white positions and connect adjacent white black positions(with 0 at their positions) with a weight 1.
2. Make a source and connect source to all white cells with a forward edge and weight 1.
3. Make a sink and connect all black cells to sink with weight=1.
4. Call the graph thus obtained as G,c as all1's,s=source,t=sink.

5. Run Edmonds-Karp (G=(V,E), c, s, t)
   1. Initialize flow f to 0
   2. While exists augmented path p (check with BFS) do
   3. Augment flow f along p
   Return f

6. total number of zeros in original matrix = k.
7. if (k>2*f)
8. print (“No”)
9. else
10. print(“Yes”)

Solution 2

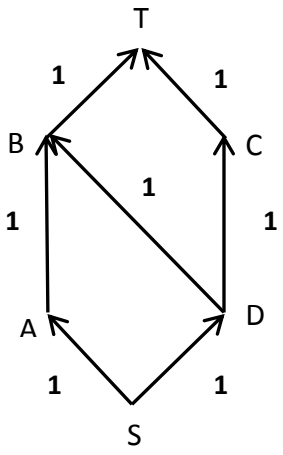

Fig 1.

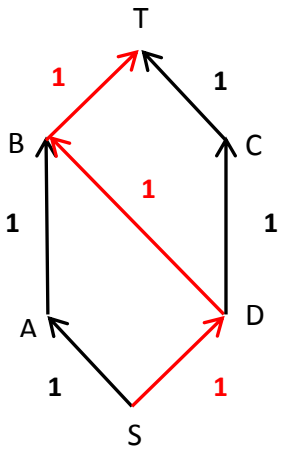

Fig 2.

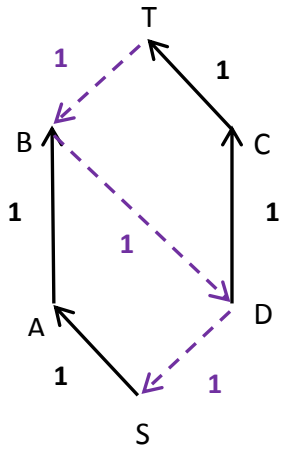

Fig 3.

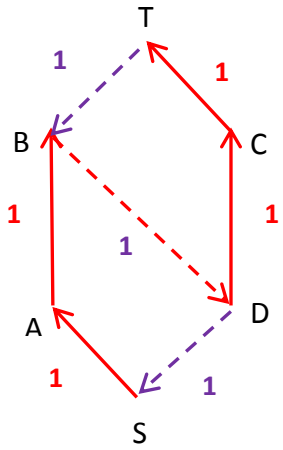

Fig 4.

Fig 1. is the graph we are considering for the example.
Fig 2. Shows the 1st augmented path chosen by the BFS being the shortest path. Thus 1 travels from the source to target via path S-D-B-T
Fig 3. Shows the residual graph with backward edges.
Fig 4. Shows how the next shortest path chosen by the BFS will have to include the backward edge BD to get the next augmented path.

This example is similar to the one given in the slides. (1000 – 1 –> slide 10) here the edge weights could have been similar with BD being 1 but for simplicity we have chosen all edges to be of unit length.

Solution 3.

(a) We input the graph of Hamiltonian Cycle as G= (V,E).

As the input to a TSP problem is a graph, Graph= (V’,V’XV’) we change the input of our Hamiltonian Cycle by converting it into a graph G’ as follows G’ = (V,VXV)

For any vertices (u, v)

If there exists an edge between u and v then weight of u, v is equal to 1, else weight is 2

- ⇨ Weight(u, v) = 1(if edge(u, v) exists in Hamiltonian input graph)
- ⇨ Weight(u, v) = 2(if edge (u, v) does not exist in our Hamiltonian input)

And the value of k =n

If there exist a Hamiltonian path, it will take the minimum path where all edges would be of weight 1 and the TSP box would give the answer YES as sum of total path would be n.

However , if there is no cycle, TSP box will take any cycle with 2 weight at least once, that would make the total cost > n and hence the answer will be NO.

Also we know TSP will traverse all the vertices just once so minimum we will have weight as n.

(b)The answer to Black box of TSP(or G') is the answer to our original graph G that is ,

if the answer given by black box is Yes, there exists a Hamiltonian cycle

If the answer given by black box is NO, there exists no Hamiltonian cycle.