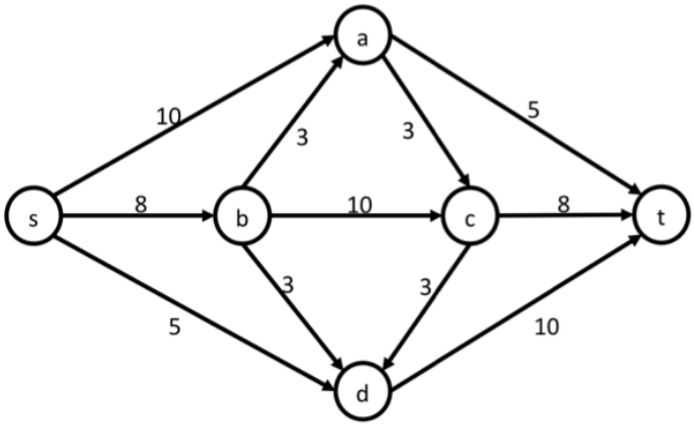
There are 2 questions for a total of 10 points.

---

1. (3 points) Consider the flow network below and the mapping f from edges to numbers:

$$f(s,a)=5, \quad f(s,b)=8, \quad f(s,d)=5, \quad f(a,c)=0,$$
$$f(a,t)=5, \quad f(b,a)=0, \quad f(b,c)=8, \quad f(b,d)=0,$$
$$f(c,d)=0, \quad f(c,t)=8, \quad f(d,t)=5$$



(a) f is an s-t flow.

Answer: True

(b) f is an s-t flow with maximum value.

Answer: False

(c) Give an s-t cut (A, B) with minimum capacity. You can specify a minimum cut by giving sets A and B.

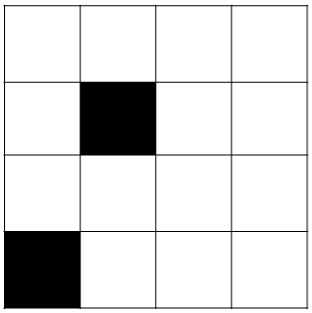
Answer: $A = \{s, a\}, B = \{b, c, d, t\}$

(d) What is the capacity of minimum-cut?

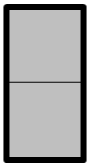
Answer: 21

2. (7 points) There is an $n \times n$ grid in which some cells are empty and some are filled. The empty/filled cells are given by an $n \times n$, 0/1 matrix F. Cell (i, j) is empty iff $F[i, j] = 0$. You have unbounded supply of 2×1 tiles (called dominoes). Each domino could be placed on the empty cells of the grid in horizontal and vertical manner (note that you need two consecutive empty cells on the grid for doing this). The problem is to determine if the grid can be covered by placing these 2×1 dominos such that each empty cell is covered by exactly one domino.

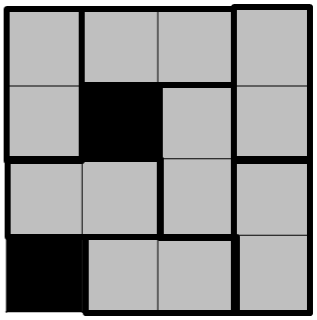
(The figure below shows an example of a 4×4 grid that can be covered using 2×1 domino tiles.)



$n \times n$ Grid



2×1 Domino tile



Grid cover using dominoes

Design an algorithm for this problem. Your algorithm should output "yes" if it is possible to cover the grid and "no" otherwise. Give running time analysis and proof of correctness.

[Note that you will receive full points if your algorithm has polynomial running time. There are no bonus points for this question.]

Solution:

We call a placement of dominos a *tiling* if the conditions mentioned in the problem are satisfied.

I will reduce this problem to finding a perfect matching in a graph.

Consider the graph G constructed as follows:

Let S be the set of all unfilled cells (i, j) which have $i + j$ odd, and let T be the set of all unfilled cells (i, j) which have $i + j$ even. Add an edge between two adjacent cells iff both are unfilled.

Claim 0: This graph is bipartite.

Proof: Note that we have an edge if and only if the cells are adjacent and empty. Note that for any adjacent cells, the sum of coordinates are $s, s + 1$ for some integer s, and hence have opposite parities. Hence one of the endpoints of the edge is in S and the other is in T. Hence we have shown that for each edge, one endpoint is in S and the other in T, so no edge has both endpoints in exactly one of the sets S and T, showing that the graph is bipartite.

Claim 1: There is a perfect matching in this graph $\implies$ there is a tiling.

Proof: Suppose a perfect matching exists. Consider a perfect matching. For each edge (u, v) in the graph, since (u, v) is an edge in G, u and v must be adjacent cells of the grid which are both unfilled. Hence, we can place a domino covering these two tiles. Now to show that this tiling is valid, note that if there were two dominos intersecting at a cell, then in the perfect matching, we must have had two edges sharing an endpoint, which is impossible due to it being a valid matching. To show that it covers all the unfilled cells, note that since the perfect matching matches all the vertices of the bipartite graph (i.e., for each vertex, there is an edge whose endpoint it is), for each unfilled cell, we must have a domino covering it. Hence we have shown that for a perfect matching in the graph, there is a tiling.

Claim 2: There is a tiling $\implies$ there is a perfect matching in this graph.

Proof: For any domino, note that it covers two adjacent squares. Either they have consecutive i's and the same j or consecutive j's and the same i. In either case, one of the cells is in S and the other one is in T. Add the edge between these two cells into a set X (and do this for all dominos). Note that since no two dominos overlap, no two edges share a vertex, and thus these edges form a matching. Also note that since all cells in S and T are covered by the dominos by the definition of a tiling, this set X of edges is a perfect matching, as required.

Hence we have shown that our problem is equivalent to finding a perfect matching in the bipartite graph G. However, we have done this problem in the class, and I will use the routine IsThereAPerfectMatching(G) for the algorithm (the details will be summarized in the runtime analysis, nevertheless).

Algorithm:

S will be represented implicitly as a list of size equal to the number of unfilled cells at odd sum of coordinates, consisting of elements from 1 to $|S|$, each representing a cell.

T will be represented implicitly as a list of size equal to the number of unfilled cells at even sum of coordinates, consisting of elements from $|S|+1$ to $|S|+|T|$, each representing a cell.

function IsThereATiling($a[1..n, 1..n]$)
  let $b[1..n, 1..n]$:= array filled with -1 initially ▷ $b[i,j]$ shall store the vertex number for each cell (i, j) that is unfilled, and -1 if it is filled.
  let $currentIndex := 1$
  let $size_S := 0$
  for each unfilled cell $(i,j) \in G$ **do**
    if $i+j$ is odd **then**
      $b[i,j] := currentIndex$
      $currentIndex++$
    end if
  end for
  $size_S := currentIndex - 1$
  for each unfilled cell $(i,j) \in G$ **do**
    if $i+j$ is even **then**
      $b[i,j] := currentIndex$
      $currentIndex++$
    end if
  end for
  $totalVertices := currentIndex - 1$ ▷ $size_T = totalVertices - size_S$
  let G := graph with vertices $1..totalVertices$
  for $i \in \{1 \dots n\}$ **do**
    for $j \in \{1 \dots n\}$ **do**
      if $a[i,j] \neq 0$ **then**
        continue
      end if
      for $(i',j') \in \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$ **do**
        if $1 \leq i' \leq n \wedge 1 \leq j' \leq n \wedge a[i',j'] = 0$ **then**
          add edge between $b[i,j]$ and $b[i',j']$ in G
        end if
      end for
    end for
  end for
  if IsThereAPerfectMatching($G, \{1 \dots size_S\}, \{size_S + 1 \dots totalVertices\}$) **then**
    output "yes"
  else
    output "no"
  end if
end function

Runtime Analysis: Creation of b takes $O(n^2)$ time. The next two loops take $O(n^2)$ time. Then while constructing the graph, in the innermost loop, we do $O(1)$ work (checks and adding to the end of adjacency lists etc), and hence we do $O(n^2)$ work in creating the graph.

Internally the perfect matching routine finds a bipartite matching and checks if the sizes of the two parts are the same and equal to the size of the max matching of the graph. This is done by the Ford Fulkerson algorithm after creating a network with the same vertices and two additional vertices for the source and the sink. Since the size of the max matching equals the max flow in the built network, and since the size of the max matching is at most n^2, the max flow is at most n^2.

The number of edges in this graph is at most $O(n^2)$ as well since each vertex in the original graph has degree bounded above by 4, and the number of edges in the network is the number of edges in the original graph + size of S + size of T (due to connections between the source and S, and T and the sink). The number of vertices in the network is at most $n^2 + 2$ as well, and hence in $O(n^2)$. Hence the complexity is $O((V + E)C)$ which is in $O(n^2 \times n^2) = O(n^4)$, and the overall complexity is also $O(n^4)$.