Yongzuan Wu
wu68
cs573
HW 5
Problem 4
reference: http://en.wikipedia.org/wiki/Maximum$_flow_problem$
(a)
Proof: Here is a counter example
For the input array $M[1..n, 1..n]$,

$$M[i][j] = \begin{cases} 1 & \text{if } (i = 1 \text{ and } j \in [1..n-1])\text{or}(i = n \text{ and } j \in [2..n]) \\ 0 & \text{otherwise} \end{cases}$$

The greedy algorithm find a good path $M[1,1] \to M[1,2] \to M[1,3] \to M[n,3] \to M[n,n]$, which covers n marked cells. Yet the remaining configuration needs at least two more path to cover, since both $M[n,2]$ and $M[1,4]$ are uncovered.

(b)
We reduce the problem to a bipartite graph matching.

Given the array $M[1..n, 1..n]$, first we find the set $V = \{v_{ij} \,|M[i,j] = 1\}$. Then We construct a directed graph $G = (K, E)$, where

$$E = \{(u, w) \,|u = v_{ij}, w = v_{i'j'}, u \neq v, i' \geq i, j' \geq j\}$$

In G, a vertex represent a marked cell, and an edge (u, w) represents a monotone path from cell u to cell w. It's esay to see that the problem is equivalent to finding a collection of paths in G that covers all vertices.

We can reduce this problem to bipartite graph matching.

Let $G' = (V \cup V', E')$ be a bipartite graph, where $V' = \{v'_{ij} \,|v_{ij} \in V\}$, $E = \{(v_{ij}, v'_{i'j'}) \,|(v_{ij}, v_{i'j'}) \in E\}$. Let M be a matching of G. Every edge in M corresponds to an edge in G. Then a vertex $v'_{ij} \in V'$ that is not covered by M indicates that v_{ij} has no succedent in its path, i.e. it is the end point in its path. Thus the number of paths in the collection defined by M is $n - |M|$. Therefore the smallest number of paths covering G is given by ($n-$the maximum matching of G). Thus the algorithm follows,

UncoverCell($M[1..n, 1..n]$)
construct G' as described above
compute the maximum matching M of G'
return $n - |M|$

Analysis: We could use the Ford-Fulkerson algorithm to compute the maximun matching. The algortihm runs in $O(VE)$. By construction, $|V| = O(n^2)$, $|E| = O(n^4)$, thus the running time is $O(n^6)$