CS 4820, Spring 2017 Homework 7, Problem 1

Name: Yuxiang Peng

NetId: yp344

Collaborator: jl3455, zl542

1a). We show that this problem could be reduced to a max-flow problem in polynomial time.

n: as defined in problem
Problem reduction:

1. For each non-boundary node(i,j), for each one of its four neighboring nodes, create an edge of capacity c from (i,j) to the neighbor. Time: 4*(n-1)^2
2. Create source node S, for S and each non-boundary node (i,j), create an edge with capacity p(i,j) from S to (i,j). Time: (n-1)^2
3. Create sink node T, for T and each boundary node (i,j), create an edge with capacity c+1(actually infinity capacity) from (i,j) to T. Time: 4*(n-1)+4

Use Edmondes-Karp algorithm to solve the resulting max-flow problem. If all edges leaving S reach its capacity(checking time: (n-1)^2), the evacuation is possible, output the flow on the edges except for the edges connecting S and T as the number of people traversing that edge in the specified direction, if both the flows on E(i,j) and E(j,i) are positive and f(i,j)>f(j,i), f(i,j)=f(i,j)-f(j,i), f(j,i)=0. This way, we ensure the edge capacity constraint of the original problem is conserved by subtracting a cycle from the max-flow solution which is still a max-flow solution. (assignment time: 4*(n-1)^2).

Runtime:
Number of vertices: (n+1)^2, number of edges: 4*(n-1)^2+(n-1)^2+4*(n-1)+4.
Edmondes-Karp: O(n^6)
Problem reduction: O(n^2)
Solution recovery: O(n^2)
So it is a polynomial algorithm.

Correctness:
We could transform the original problem to a max-flow problem and recover a solution to the original problem from the max-flow problem solution. If the max- flow solution has an unsaturated edge leaving S, it means we can't evacuate all people at that corresponding node, therefore we can't evacuate all people of the city. Otherwise, we could evacuate all people, the edge capacity constraint of the original problem is conserved in the max-flow problem through the reduced problem construction and the recovery of the original problem as explained in the algorithm.

b)
The hot spot algorithm is correct.

We prove if the evacuation is not possible, the min-cut of the corresponding max-flow problem corresponds to some hot spot. Therefore if we could not find a hot spot, the evacuation is possible.

For a city with an impossible evacuation plan, say the min cut of the corresponding max-flow problem is set A, B. Set A contains the source S. Set A could only has only non-boundary nodes as boundary nodes have edges with infinity capacity connecting the sink. Let the inner nodes in A be X, let the inner nodes in B be Y. Let o(X) be the total number of outlets in X, n(Y) be the total population in Y, n be the total number of people. Since evacuation is impossible, o(X)<n-n(Y)=>o(X)<n(X).

Set X might have disjoint components, we now prove that one of the components say Z also has the property o(Z)<n(Z). Say X has disjoint components $x_1 \ldots x_n$. Different components could not share outlets as that would make those components one joint component. So each outlet of each disjoint component is one outlet of X, thus
$\sum_i o(x_i) \leq o(X),\ \sum_i n(x_i) = n(X)$
Therefore $o(x_i) <= o(x_i)$ for some i.

For this x_i, find the smallest rectangle R that includes it, n(R)>=n(x_i), o(R)<=o(x_i), as each outlet in R has a corresponding outlet in x_i since if we traverse the outlet in R horizontally or vertically we will find an outlet in x_i. Therefore o(R)<n(R). We found a hotspot.