CS430 Homework 6

Chapter 7

Problem-1:

In the problem we are given a set of floor walls, fixtures, and switches. The switches are to be connected in such a way that the connection should not pass through the walls.

Give an algorithm to decide if a given floor plan is ergonomic.

Answer:

We can use bipartite graph to find perfect Matching to solve problem

So, define Graph G= (V, E) that summarizes floor plan

Set V is divided into 2 sets A and B, light switches ui ∈ A and light fixtures vi ∈ B.

Connect (ui, vi) with an edge if line drawn from ui to vi not intersect by any walls in the floor plan.

Each edge has capacity equal to 1.

Create source node s and destination node t and convert graph into flow Network.

Connect **s** to each switch nodes and connect all fixtures to sink node t

set capacity to 1 for all these edges.

Algorithm:

1. Start with initial flow as 0
2. While there is augmenting path from source s to destination t

   Add this path-flow to flow
3. If flow==n

   Print floor plan is Ergonomic

   Else

   Return False

Runtime Analysis: $O(n^3)$

Maximum capacity of Graph is n

There are n*n edges between fixture nodes and switch nodes

Running Time of Algorithm would take O(mf) = **$O(n^2)$** {in the worst case every light fixture is connected to every light switch} * **O(n)** {max flow} = **$O(n^3)$**

Problem-2

Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.

Answer:

As per the given scenario,

There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location.

Means, several people can be sent to one hospital so we can solve this problem by Maximum-flow Algorithm.

Algorithm:

1. Directed Graph G (V, E) where V to be as set of patients and hospitals
2. Set V is divided into 2 sets A and B, patients ui ∈ A and hospitals vi ∈ B.
3. we add an edge (ui, vi) from patient ui to hospital vi where patient can be evacuated.
4. Create source node s and destination node t convert graph into flow Network.
5. Connect source s to each patient node ui and each hospital vi to destination/sink node t
6. Set capacity equal to 1 for each edge from s to a patient node
7. Similarly, Set capacity equal to 1 , for each edge from a patient node to each hospital node where she/he is admitted to

8. Set capacity = [n/k] for each edge from hospital to destination node t
   // we have to choose a hospital in such a way that the load on the hospitals is balanced.
9. calculate Maximum flow by ford-Fulkerson algorithm
10. If value of flow==n

        return true

    //possible to evacuate patients to hospital in such a way that the load on the hospitals is balanced

    Else

        return false

    //balanced evacuation is not possible

Runtime Analysis: O(kn^2)

Max-flow algorithm that is O(mF) = O(kn2)

In worst case, where there are nk+n+k = nk edges

every person can be reached to every hospital and has the max flow of n. Therefore O(kn2).

Problem-3

a)

Answer:

We have given n people set A = {P1, P2, P3,…….Pn} and n nights set B ={d1,d2,d3…….dn).

For person pi, there is a set of nights Si ⊂ {d1, d2, d3 . . . , dn} when they are not to cook

For a feasible dinner schedule, each person Pi cooks on exactly one-night dj, there is someone cooking on each night.

As per the given information we can say that this is the matching problem

So, we can solve problem by bipartite Graph to find **perfect Matching**

A bipartite Graph G= (V, E),

Set V is divided into 2 sets A and B, People pi ∈ A and light night dj ∈ B.

Connect/add edge (Pi, dj) if person Pi is available to cook on night dj

// dj does not belong to Si

Set each edges capacity to 1.

Create source node S and destination node T.

Connect **S** to all each person Pi and sink node **T** connected to all nights dj.

Set capacity equal to 1 for all these edges.

If Max-Flow Algorithm for the above Graph G will return Maximum-Flow value equal to 1

Then we can say perfect matching available which gives a feasible dinner schedule.

Runtime Analysis: $O(n^3)$

Maximum capacity of Graph is n

There are n*n edges between fixture nodes and switch nodes

Running Time of Algorithm would take O(mf) = **$O(n^2)$** {in the worst case every light fixture is connected to every light switch} * **O(n)** {max flow} = **$O(n^3)$.**

b)

For each person pi, there's a set of nights Si ⊂ {d1, . . ., dn} when they are not able to cook.

For People, pi and pj, and the other two days, dk and dl

Both pi and pj assigned to cook on same night dk and

No one to cook on night dl

Answer:

So As per the Given details,

Night dk doesn't belongs to si or sj

CASE-I

If dl does not belong to Sj

Person Pj can cook on night dl

remove an edge from (pj, dk)

Add an edge (pj, dl) //print result and return true

CASE-II

If dl does not belong to Si

Person pi can cook on night dl

Remove an edge (pi, dk)

Add ad edge (pi, dl) print result and return true

CASE-III

Set E all incoming edges to night dl

For each source node x in E:

Let source node x is assigned to night dx in n-2 schedule

//means edge (px, dx) in n-2 schedule

If dx does not belongs to Si and dl does not belongs to Sx

//Means Px can work on night dl

Remove an edge (px, dx)

Add an edge (px, dl) and (pi, dx) //print result and return true

If dx does not belong to Sj and dl does not belong to Sx

//Means px can cook on night dl

Remove an edge (px, dx)

Add an edge (px, dl) and (pj, dx)// print result and return true

If we can find such edges as described in the previous 3 Cases, then it means that there exists a feasible dinner schedule else return false implied no such schedule can be formed

Runtime Analysis: $O(n^2)$

- For loop to iterate through all the incoming edges to night dl and inside the for loop we are using if statement to check the relationships takes n iteration.
- So, complexity of the algorithm is O(n*n).

Problem-4

Answer:

Ford Fulkerson Algorithm is used to solve this problem as firstly we need to find minimum S-T cut.

All edge of graph has assigned edge capacity Ce > 0

Then will check whether that cut is unique or not.

Algorithm:

1. Construct Residual Graph **G**′ for graph **G** by using Ford-Fulkerson Algorithm.
2. Find the set of vertices that are reachable from the source in graph **G**′
3. Compute Minimum s-t cut S1 of Graph G
   //All edges which are from a reachable vertex to non-reachable vertex
   //S1= {e1,e2,e3…….en}
4. Calculate capacity(volume) of minimum s-t cut S1 = **|S1|**
5. Increase capacity of each edge ek in S1 by 1
6. Compute Minimum s-t cut S_k in new graph
7. Calculate capacity of minimum cut S_k =**$|S_k|$**
8. If $|S1|=|S_k|$ for some k,
   //Means S_k is also minimum cut in original graph G
   and edge sets are different s1! = S_k

   Print Minimum - cut is **not Unique** cut

//multiple minimum-cut exist in original graph so total capacity is not increased by increasing the edge capacity.

9. Else

   Print Unique Minimum-Cut

10. This implies that for all k, if |S1|<|Sk| then the original graph has unique minimum cut.

Runtime Analysis:

if there are n edges in the min cut of Graph G then, the total runtime is at most O(n)*O(mF) (Ford- Fulkerson algorithm) which gives polynomial time complexity.