Homework 2 Graph Algorithms - 23 November, 2020

Constantinescu M.G. George-Gabriel, group E3, year II
Bogatu D. Daniel-Marian, group E3, year II

1 Problem 1

proof: (a)

Let X_1 and X_2 two bipartion classes.

Let dg_1 and dg_2 be the degree of the vertices from X_1 and X_2.

We suppose that $|X_1| \leq |X_2|$. By this , we can say that the number of edges in the graph is equal with the number of vertices multiplied by the degree of the vertices from the same bipartition.

$$|E(G)| = |X_1| \cdot dg_2$$
$$\text{(or)}$$
$$|E(G)| = |X_2| \cdot dg_1$$

$$\text{Hence, } dg_1 \geq dg_2 \ .$$

Let M be a subset of the class X_1 and G' the induce bipartite subgraph with the bipartition class M and the projection of M , $N_g(M)$.

We can find that the number of edges in G', $|E(G')|$ is equal with the number of edges in $N_g(M)$ times the degree of the vertices dg_2 from the bipartition class X_2 and also is equal with the number of vertices of M , $|M|$ times the degree of the vertices dg_1 from the bipartition class X_1.(all the vertices are incident with the edges from G')

$$|E(G')| = |N_g(M)| \cdot dg_2$$
$$\text{(or)}$$
$$|E(G')| = |M| \cdot dg_1$$

Hence, there exists a vertex from $N_g(M)$ such that $d_G(u) \geq \frac{|E(G')|}{|N_G(M)|}$, where u is a vertex from the subset M.

Hence, we obtain the next relation :

$$dg_2 \geq \frac{dg_1 \cdot |M|}{|N_G(M)|}$$

From $dg_1 \geq dg_2$ and $dg_2 \geq \frac{dg_1 \cdot |M|}{|N_G(M)|}$ we obtain this new relation: $dg_1 \geq \frac{dg_1 \cdot |M|}{|N_G(M)|}$. After we simplify the relation , we found that: $|N_g(M)| \geq |M|$.

All in all ,$\forall M$ from X_1 we have a matching which saturates all the vertices from at least one bipartition class.

In the same way , we can do this for $|X_2| \leq |X_1|$.

All things considered, using the Hall's theorem , there exists a matching which saturates at least one of the bipartition classes from graph G.

proof: (b)

We choose 6 cards randomly from the deck : C_1,C_2,C_3,C_4,C_5,C_6.

We select a pair of cards (C_i,C_j),i≠j such that both cards C_i and C_j have the same symbol.

We put the card with the highest value from the pair with the face down and the other one with face up. If the difference between them is greater than 6 we choose the card with the smallest value of the remaining 4 and we put it as the last card and we add 6 to the card who is with the face up.

After this the remaining 3 cards C_a and C_b and C_c,we order them in a way such that we can apply one of the following rules:

- Low-Medium-High : +1
- Low-High-Medium : +2
- Medium-Low-High : +3
- Medium-High-Low : +4
- High-Low-Medium : +5
- High-Medium-Low : +6

After all, the card which is with the face down will have the colour of it's neighbour with the number which is the sum of the value of it's neighbour and the value we obtained using the set of rules ahead.

Observation: If the difference between the 2 cards is not grater than 6, we follow the exact same steps without adding 6 in the first way.

As an example, we have the following cards: 3 of red hearts, 2 of clubs, 5 of diamonds,10 of red hearts, 8 of clubs and 9 of black hearts.

We choose the first 2 cards : 3 of red hearts and 10 of red hearts(they have the same symbol). Now we put 10 of red heart with the face down and 3 of red heart with the face up. We calculate the difference between them, which is 7. We see that this value is greater than 6 , so we choose from the remaining cards the one with the smallest value : 2 of clubs as the last card and then we add 6 to that value of the second card.(3+6=9). Now we have to sort our remaining 3 cards in the way that we'll get our result(value 10). In order to get value 10 , we have to add +1 to the value we calculated before(9). In this case, we sort them in this following way : 5 of diamonds, 8 of clubs and 9 of black hearts. And now we can deduce the value of the card with the face down , being the symbol of it's neighbour(red hearts) and the value equal to 10(3+6+1).

Hence,we can agree on a system which always makes this possible.

2 Problem 2

proof: (a)

We have this property at a) : $\forall$ pair of servers (X_i,X_j),$i \neq j$ where the distance between them is 2, the servers X_i and X_j are directly connected. Using this property we can say that a connected graph composed only by servers is a complete graph.

Let G be a connected graph with 3 servers X_1,X_2 and X_3 with the edges:X_1X_2 and X_2X_3 .Using the property ahead we can say that there exists a edge X_1X_3 such that $d_G(X_1) = d_G(X_2) = d_G(X_3) = 2$.Hence, the graph is complete.

Now, we add a new server X_4 to the graph G which is connected to one of the other servers which already exists in the graph. We know that the graph G without X_4 is a complete graph, so after we add X_4 to the graph the distance between X_4 and other server that is not directly connected with X_4 is 2. Hence, if we apply again the property ahead we can say that the new graph G with the new server X_4 is a complete graph.

In general : Let G' be a complete graph with n-1 vertices(or servers). We add to the graph G' a new vertex X_n connected through one edge to G'. The distance between the new server S_n and any other server besides the one who is connected with in G' is 2 , hence we can say by applying the property ahead that we obtain a complete graph.

Using the general property we can say a connected graph is a complete graph. After we erase a vertex from a complete graph , the resulted graph will be also connected.We know that a client is connected to at least 2 servers.

According to the paragraph above, if a server becomes unavailable the clients can still communicate with all the remaining servers.

proof: (b)

Let C_1 and C_2 two circuits with a common edge called m.

We can remove 2 edge in the following ways:

- edge m and another edge from C_1;
- edge m and another edge from C_2;
- one edge from C_1 and one edge from C_2;
- two edges from C_1 or two edges from C_2;

If we remove an edge from a circuit, the circuit will remain connected. By this, we can say that the graph remains connected in the first 3 cases.

In the 4th case(two edges from C_1 or two edges from C_2) there are 2 possibilities :

the edges e_1 and e_2 are adjacent
(or)
the edges e_1 and e_2 are not adjacent;

In the first case, when the edges e_1 and e_2 are adjacent, there must exist another circuit C_3 because of the property at b) : any direct connection between two servers is contained in two cycles intersecting only in e. In this way, the graph will remain connected.

In the second case, when the edges e_1 and e_2 are not adjacent, there must exists two circuits C_3 and C_4 for keeping the connectivity safe.

In conclusion, the network remains connected even if any two cables between server fail.

3 Problem 3

proof: a)

We can say that if we have 2 different b-subgraphs and their intersection has multiple vertices than they can form by uniting themselves another b-subgraph which is practically pointless (because the unification of the 2 b-subgraphs could make graph G which should be only connected,not 2-connected).Hence, the intersection between 2 b-subgraphs should have only one vertex.

Let X_1 and X_2 two b-subgraphs in G. The intersection between X_1 and X_2 is a vertex x from the set of vertices of graph G,V(G).

If we remove x , both subgraphs X_1 and X_2 would break apart from each other and the property of G being connected will be lost.Hence, the intersection between the two b-subgraphs is a vertex-cut(x is the vertex cut).

Therefore, we can say that the intersection between two b-subgraphs is a cut vertex,so any vertex-cut is the point of the intersection betweeen two b-subgraphs. All in all, we proved that if two b-subgraphs of G have common vertices, then their intersection is a cut vertex of G and any cut-vertex is the intersection of at least two b-subgraphs.

proof: (b)

Let X_1,X_2,X_3 and X_4 four b-subgraphs such that the intersection between X_1 and X_2 is equal with x, where x is a vertex from V(G) and there is a path from X_1 to X_2 and also a path from X_3 to X_4.

Using point a), we know that between two b-subgraphs is a cut-vertex. In our case, we can say that we can't have a path between C and D because there would be a contradiction.Hence, no circuits exists between b-subgraphs. So, we can say that our meta-graph $\mathcal{G}$ is acyclic. **(1)**

We also know that $\mathcal{G}$ is connected because it is a bipartite graph. **(2)**

Considering **(1)** and **(2)**, $\mathcal{G}$ is a tree.

proof: (c)

We know from the implementation of a tree, if we remove the root of the tree(obs:tree has at least 2 children), the tree will disconnect and we can say that the graph will not be connected.Also, if we remove a vertex from the tree where it have at least 1 child, we can say also that the graph will not be connected. **(1)**

We can consider the graph G as follows: all the vertices of graph G can be b-subgraphs and the edges can play the role of the paths between all the "b-subgraphs" of the graph G. **(2)**

In conclusion, using **(1)** and **(2)** , if a graph G is a tree where every vertex can be a b-subgraph,eliminating the root of the tree will make the graph G be not connected or eliminating a vertex from the tree which has at least 1 child,it will also make the graph G not connected. So,we can say that the above algorithm is correct.

4 Problem 4

proof: (a)

Everytime we enter the while loop, we have $p \geq 2$, the number of connected components.

So, in this case we have to divide the problem in two separate cases : Worst Case Scenario(WCS) and Best Case Scenario(BCS).

In WCS, every edge is selected twice and we remain with $\frac{p}{2}$. An edge ab can be selected twice if it's the least choice for T_i and T_j, with vertex $a \in V(T_i)$ and vertex $b \in V(T_j)$.

In BCS, we choose the minimum edges such that at the end of the iteration will be only one connected component.

All things considered before, we can say that after each while iteration the number of connected components of T is at least halved.

proof: (b)

We can see that a cycle can be composed only when,during an iteration in the algorithm, between 2 connected components we choose 2 edges ab and cd with $ab \neq cd$. But here is a contradiction because in the algorithm, we choose everytime the minimum cost edge(let $e_i \in E$ such that $c(e_i) = \min...$).

So, we can say that after each while iteration T is acyclic.

proof: (c)

Everytime we enter the while loop, the edges who are chosen are of cost minimum. **(1)**

We can say that T is a tree because after the algorithm ends, T is connected and doesn't contain cycles(from point b). **(2)**

Considering **(1)** and **(2)** , T is a minimum cost spanning tree of G.

proof: (d)

Using point a), we know that everytime we are in the while loop we are stuck in the WCS(Worst Case Scenario) with half of the remaining connected components.So, everytime we enter the while loop the number of connected components will be halved.For example : From n we go to $\frac{n}{2}$ and then to $\frac{n}{4}$ and so on).

Hence, for $n = 2^k$ vertices, the while loop will repeat k times , so the number of the while iterations is $\mathcal{O}(\log n)$. **(1)**

We can see that in while loop we have a for instruction where it's perform a search through all the edges of the graph.

Observation: We can see in the algorithm that all the p components contains all the vertices of the graph(T is initially formed only with the vertices of the graph). Hence, each while iteration takes $\mathcal{O}(m)$ time. **(2)**

Considering **(1)** and **(2)**, the overall time complexity of the algorithm is $\mathcal{O}(m \log n)$.