CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

1. The ***incidence matrix*** of a directed graph $G = (V, E)$ with no self-loops is a $|V| \times |E|$ matrix $B = (b_{ij})$ such that

$$b_{ij} = \begin{cases} -1 & \text{if edge } j \text{ leaves vertex } i \text{ ,} \\ 1 & \text{if edge } j \text{ enters vertex } i \text{ ,} \\ 0 & \text{otherwise .} \end{cases}$$

Describe what the entries of the matrix product BB^{T} represent, where B^{T} is the transpose of B.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

1 Ans>

Let us assume that there is a graph with 3 vertices $[X, Y, Z]$, the edges for the graph are $[X \to Y, X \to Z]$

The incidence Matrix $B = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$

The transpose of $B = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}^T = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$

The product of $B * B^T = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$

$$= \begin{bmatrix} 1+0 & -1+0 & 0+0 \\ -1+0 & 1+1 & 0-1 \\ 0+0 & 0-1 & 0+1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

From the above matrix, we can derive following

$B.B^T(u\ v)$

→ matrix (0,0) = 1, X participates in 1 edge

→ matrix (0,1) = -1, as there is edge from $X \to Y$

→ matrix (0,2) = 0, no edges $X \to Z$ or $Z \to X$

→ matrix (1,0) = -1, one edge from $X \to Y$

→ matrix (1,1) = 2, 2 edges $X \to Y$ and $Y \to Z$

→ matrix (1,2) = -1, 1 edge from $Y \to Z$

→ matrix (2,0) = 0, no edges between $X \to Z$ or $Z \to X$

→ matrix (2,1) = -1, 1 edge between $Y \to Z$

→ matrix (2,2) = 1, 1 edge $Y \to Z$

For $B * B^T$, we can say that

$$BB^T(u,v) = \begin{cases} \text{indegree + outdegree} & (\text{when } u = v) \\ -(\text{no. of edges between } u \,\&\, v) & (\text{when } u \,!= v) \end{cases}$$

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

2.

Given a graph G and a minimum spanning tree T, suppose that we decrease the weight of one of the edges in T. Show that T is still a minimum spanning tree for G. More formally, let T be a minimum spanning tree for G with edge weights given by weight function w. Choose one edge $(x, y) \in T$ and a positive number k, and define the weight function w' by

$$w'(u,v) = \begin{cases} w(u,v) & \text{if } (u,v) \neq (x,y) , \\ w(x,y) - k & \text{if } (u,v) = (x,y) . \end{cases}$$

Show that T is a minimum spanning tree for G with edge weights given by w'.

2 Ans> According to Kruskals algorithm, a tree T is a minimum spanning tree of a graph G=(V,E) if it contains the top (|V|-1) minimum weighted edges (no cycles).

So, lets consider an edge (x,y) in tree T, say its weight is reduced. The Kruskals algorithm selects the minimum weighted edge among the left over edges, thereby selecting (x,y) becaus (x,y) is still considered by Kruskals algorithm as long as the weight of non-tree edges are same.

Therefore, we can say that decreasing the weights of each edge in minimum spanning tree does not change the edges in the solution as long as the weights of non-tree edges are same.

Hence, tree T will be a minimum spanning tree for edges with weights w'.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

3. Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Prim's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W?

3Ans> Implementation of Prims algorithm:

→ Firstly we initialize all keys to ∞ and first vertex to 0,

→ We select the minimum key and then picked vertex should not be in MST, Now we add the vertex to MST

→ Now we look for adjacent vertices and repeat the above process

→ By continuing this process, we find the minimum spanning tree.

EXTRACT-MIN : We get the vertex with minimum key value

DECREASE-KEY : After getting the minimum key, we update the key of its adjacent vertices, if new key is smaller update it in data structure

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

The runs time of Prims algorithm:

Initialization : $O(V)$

Extract Min : $O(V *$ time for EXTRACT-MIN$)$

Decrease-Key : $O(V *$ time for DECREASE-KEY$)$

The Van Emde Boas priority queue (range $1 \ldots |V|$) can speed up EXTRACT-MIN & DECREASE-KEY to $O(\log \log V)$.

$$\text{Total run time} = O(V \log \log V + E \log \log V)$$
$$= O(E \log \log V).$$

→ If range is from 1 to W, we can implement queue as array $[1 \ldots W+1]$, i^{th} slot has double linked list. The $(W+1)$ slot has ∞.

EXTRACT-MIN : $O(W) = O(1)$ (returns first element of list)

DECREASE-KEY : $O(1)$ (moving element from one slot to another)

4. Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles, let m be the maximum over all vertices $v \in V$ of the minimum number of edges in a shortest path from the source s to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m + 1$ passes, even if m is not known in advance.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

4Ans) We can implement the change to BELLMAN-FORD algorithm by remembering if V was relaxed or not. If V is relaxed, then we wait and check if V was updated (relaxed again). If V is not updated, we stop.

```
BellmanFord (G, s)
  Initialize (G,s)
  for i = 1 to |V| - 1
    for each edge (u, v) ∈ E
      relax(u, v)
  for each edge (u,v) ∈ E
      if d(v) > d[u] + w(u,v)
          return false
  return true
```

⇒ Here, we cannot change "for" statement to for i=1 to m+1, because we do not know the m value.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

But we can change the whole for loop to terminate after $m+1$ passes.
This can be proved if d-values are correct after m passes.
From path relaxation property:
→ If p is a shortest path from $s = v_0$ to v_k and the edges of p are relaxed in order $(v_0, v_1), (v_1, v_2) \dots (v_{k-1}, v_k)$ then $d[v_k] = \delta(s, v_k)$ (regardless of any other relaxations)
→ Each edge will be relaxed once in each pass over the edge set, k-iteration give correct set of d-values of v_k. We know that every vertex v lies at end of shortest s-v path containing m edges, Hence m iterations are enough to correcty populate d-values of all vertices in graph G.

5. Professor Gaedel has written a program that he claims implements Dijkstra's algorithm. The program produces $v.d$ and $v.\pi$ for each vertex $v \in V$. Give an $O(V+E)$-time algorithm to check the output of the professor's program. It should determine whether the d and π attributes match those of some shortest-paths tree. You may assume that all edge weights are nonnegative.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 6 – Graph Algorithms

Neha Goud Baddam
11519516

5Ans) Firstly we assume that all edges weighs are non-negative.

1. Verify
   - i) $S.d = 0$ and $S.\pi = NIL$
   - ii) $v.d = v.\pi.d + w(v.\pi, v)$ for all $v \neq S$
   - iii) $v.d = \infty$ if and only if $v.\pi = MIL$ for all $v \neq S$

2. i) If any of the above conditions fail, the output is declared as incorrect. Otherwise, run one pass of Bellman-Ford (relax each edge one time)

   ii) If any values of v.d changes, declare the output as incorrect. Otherwise output is declared as correct.