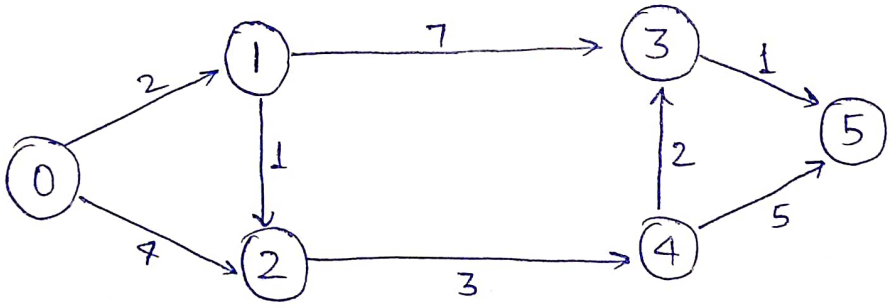
Objective :- Implement shortest path (Dijkstra's) algorithm

Approach :-

The algorithm computes for each vertex u the distance to u from the start vertex v, ie→ The weight of shortest path between v & u.

The algorithm keeps track of the set of vertices for which the distance has been computed. Every vertex has a label associated with it. The algorithm will update the D[u] value when it finds the shorter path from v to u. When a vertex u is added in the algorithm, its label is equal to the actual distance between the starting vertex & end vertex.

Dijkstra's algorithm creates the tree of the shortest paths from the starting source vertex from all other points in the graph.



Code :-

```
# include < iostream >
# include < bits/stdc++.h>
 using namespace std;
 int main() {
                vector< pair < int, int>>
graph[6] = {{{1,2},{2,4}},{{2,1},{3,7}},{{4,3}},{{5,1}},
            {{3,2},{5,5}}, {} };
```

Name : Abhishek Anand *Roll No. :* 57

Section : A *Year :* 3rd

```
priority_queue < pair < int, int >, vector < pair < int, int >> , greater
    < pair < int, int >>> pq ;

vector<int> dist ( 6, 100000000) ;
 dist [0] = 0 ;
 pq.push ( {0,0} );
 While ( !pq.empty()) {
        int a = pq.top().first ;
        int b = pq.top().second ;
         pq.pop() ;
        for (autoit : graph [a]) {
            if ( b+ it.second < dist [ it.first ]) {
                    dist [it.first ] = b+ it.second ;
              pq.push ({ it.first, dist [ it.first ]});
                }
            }
        }
    for (auto it : dist) {
                cout << it << "-->" ;
                }
            return 0 ;
        }
```

Output :

0 --> 2 --> 3 --> 8 -> 6 --> 9 -->

Discussion :-

For this problem, First I mark all vertex as unvisited vertex. Then I marked the source vertex as 0 & all other vertex as infinite. considered source vertex as current & calculate the path length of all the neighbouring vertex from current vertex by adding weight of the edge in the current vertex. Then, I repeated this process until all the vertex are marked as visited. In this way, I got the desired path.

Name : Abhishek Anand

Roll No. : 57

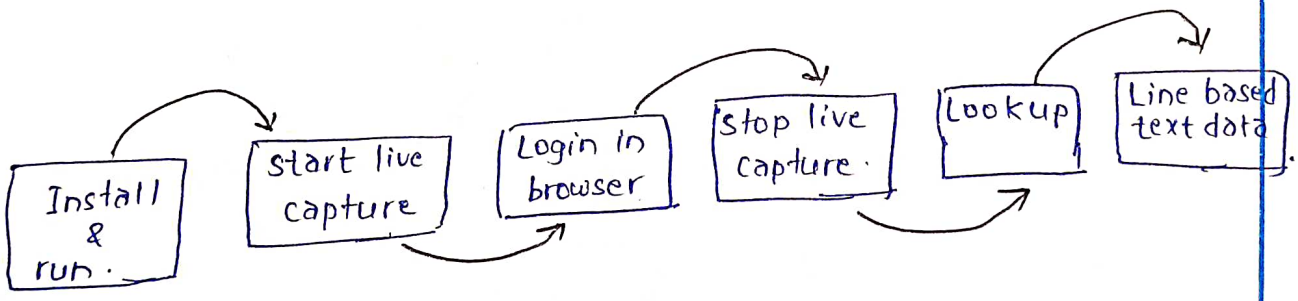
Section : A

Year : 3rd.

Objective :- Sniffing for passwords with wireshark.

Approach :-

- First of all, opened the terminal & installed Wireshark using the command,
  $ sudo apt-get install wireshark
- To Run the Wireshark, I wrote the command,
  $ sudo wireshark.
- Now, On clicking the Interface list. I activated the capture interfaces mode.
- After that I opened a webpage gogo6.com & signed up using test gmail & password!
- Then gone back to wireshark & stop the live capture
- Filter for the HTTP protocol results only using the filter textbox.
- Located the info column & look for entries with the HTTP verb POST and clicked on it.
- There I obtained the encoded Email & password that I used during sign in.



Discussion -

On doing the above mentioned steps, I learned how to sniffing passwords using wireshark. Some of the screenshots I have uploaded in the google classroom. related to it.

Name : Abhishek Anand
Roll No. : 57
Section : A
Year : 3rd.

Objective → IPC using Message Queues (Inter Process communication)

Introduction →

A message queue is a linked list of messages stored within the kernel and identified by a message queue identifier. A new queue is created or an existing queue opened by msgget(). New queue messages are added to the end of a queue by msgsnd(). Every message has a positive long integer type field, a non-negative length and the actual data bytes, all of which are specified to msgsnd() when the message is added to a queue. Messages are fetched from a queue by msgrcv().

All process can exchange information through access to a common system message queue. The sending ~~places~~ process places a message onto a queue which can be read by another process.

System calls used for message queues –

ftok() – used to generate unique keys.

msgget() – either returns the message key identifier for a newly created message queue or returns the identifier for a queue which exist same key.

msgsnd() – Data is placed on a message queue

msgrcv() – messages are retrieved.

msgctl() – It is used to destroy message queue

Code →

```
// message queue for writer process.
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
```

Name: Abhishek Anand Roll No.: 57

Section: A Year: 3rd.

```
struct mesg_buffer {
        long mesg_type;
        char mesg_text[100];
        } message;

int main() {
        key_t key;
        int msgid;
        key = ftok("progfile", 65);
        msgid = msgget(key, 0666 | IPC_CREAT);
        message.mesg_type = 1;
        printf("Write Data : ");
        gets(message.mesg_text);
        msgsnd(msgid, &message, sizeof(message), 0);
        printf("Data send is : %s \n", message.mesg_text);
        return 0;
        }
```

// Message Queue for Reader process

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
struct mesg_buffer {
        long mesg_type;
        char mesg_text[100];
        } message;

int main() {
        key_t key;
        int msgid;
        key = ftok("progfile", 65);
        msgid = msgget(key, 0666 | IPC_CREAT);
        msgrcv(msgid, &message, sizeof(message), 1, 0);
```

Name : Abhishek Anand

Roll No. : 57

Section : A

Year : 3rd.

```
Printf (" Data Received is : %s \n", message.mesg_text);
msgctl (msgid, IPC_RMID, NULL);
return 0;
}
```

Output :

Writer process

Write Data : Abhishek Anand

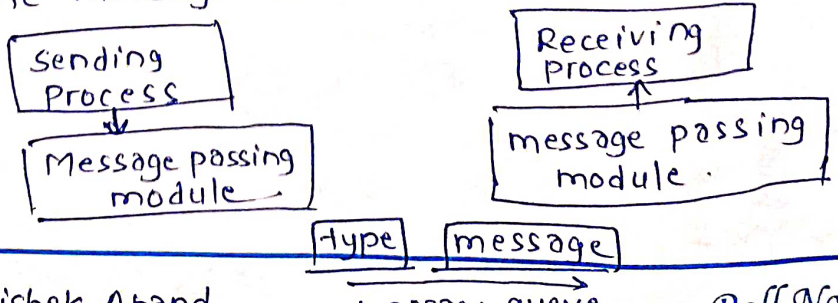
Data send is : Abhishek Anand

Reader process

Data Received is : Abhishek Anand.

Discussion :

For IPC, first I written & implemented the code for writer process, In which I firstly written the structure for message queue. then using ftok to generate unique key then used msgget to create a message queue then msgsnd to send message. & For the reader process I did the same in starting, Firstly I written the structure for message queue, then ftok to generate unique key, then msgget to create message queue, after that I used msgrcv to receive the message and atlast I used msgctl to destroy the message queue after displaying the message which is received.



Name : Abhishek Anand Roll No. : 57

Section : A Year : 3rd.