15-826 Project Phase1

Silun Wang
silunw@andrew.cmu.edu
Yuwei Zhang
yuweiz1@andrew.cmu.edu

```
#Task 0: kcore
def gm_kcore ():
    # compute coreness of each node
    print "Computing kcore..."

    cur = db_conn.cursor()
    GM_TABLE_DUP = "GM_TABLE_DUP"
    GM_KCORE_TMP = "GM_KCORE_TMP"
    gm_sql_table_drop_create(db_conn, GM_KCORE, "node_id integer, \
                             coreness integer")
    gm_sql_table_drop_create(db_conn, GM_TABLE_DUP, "src_id integer, dst_id integer")

    cur.execute("insert into %s" % GM_TABLE_DUP + " select src_id, dst_id from %s"
        %GM_TABLE)
    db_conn.commit()

    cur.execute("create index on %s (src_id, dst_id)" % GM_TABLE_DUP)
    db_conn.commit()

    k = 1
    while True:
        # each time we pick out elements with less than k neighbors and output them with
            coreness k
        # delete from the original table whose neighbor are those elements
        # if we get no such elements, we increase k
        # until we get no elements left in the orginal table

        gm_sql_table_drop_create(db_conn, GM_KCORE_TMP, "src_id integer, \
                             neighbor integer")
        cur.execute ("insert into %s" % GM_KCORE_TMP +
                    " select src_id, count(*) as neighbor from %s" % GM_TABLE_DUP +
                    " group by src_id having count(*) <= %d" % k)
        cur.execute("create index on %s (src_id)" % GM_KCORE_TMP)
        db_conn.commit()
        cur.execute("select count(*) from %s" % GM_KCORE_TMP)
        val = cur.fetchone()[0]
        if val == 0:
            k += 1
            continue

        cur.execute ("INSERT INTO %s" % GM_KCORE +
                             " SELECT src_id , %d" % k + " as coreness from %s"
                                %GM_KCORE_TMP)
        db_conn.commit()

        cur.execute("delete from %s"%GM_TABLE_DUP + " where src_id in (select src_id from
            %s)"%GM_KCORE_TMP)
        cur.execute("delete from %s"%GM_TABLE_DUP + " where dst_id in (select src_id from
```

```
            %s)"%GM_KCORE_TMP)
        db_conn.commit()

        cur.execute("select count(*) from %s"%GM_TABLE_DUP)
        val = cur.fetchone()[0]
        if val == 0:
            break

    gm_sql_table_drop(db_conn, GM_TABLE_DUP)
    gm_sql_table_drop(db_conn, GM_KCORE_TMP)

    cur.close()
```