Branch: master ▾

Find file Copy path

cs-35l / assignment9 / topo_order_commits.py

prithvikannan updated lab9

f09e7f6 2 days ago

1 contributor

Raw Blame History

174 lines (160 sloc) 5.65 KB

● Code navigation is available for this repository but data for this commit does not exist. Learn more or give us feedback

```python
#!/usr/bin/python3

# Prithvi Kannan
# UID: 405110096

import os
import zlib
import sys

class CommitNode:
    def __init__(self, commit_hash):
        """
        :type commit_hash: str
        """
        self.commit_hash = commit_hash
        self.parents = set()
        self.children = set()

    def __str__(self):
        return 'Commit Hash: ' + self.commit_hash

def getObjectDir():
    top_level = find_root(os.getcwd())
    object_dir = top_level + '/.git/objects/'
    return object_dir

def find_root(test, dirs=(".git",), default=None):
    import os
    prev, test = None, os.path.abspath(test)
    while prev != test:
        if any(os.path.isdir(os.path.join(test, d)) for d in dirs):
            return test
        prev, test = test, os.path.abspath(os.path.join(test, os.pardir))
    sys.stderr.write('Not inside a Git repository')
    exit(1)

def get_parents_of(hash):
    parent_hashes = []
    path = getObjectDir() + hash[:2] + '/' + hash[2:]
    contents = zlib.decompress(open(path, 'rb').read())
    if (contents[:6] == b'commit'):
        contents = contents.decode().split('\n')
        for line in sorted(contents):
            if(line[:6] == 'parent'):
                parent_hash = line[7:]
                parent_hashes.append(parent_hash)
    return parent_hashes

def print_graph(nodes):
    for hash in sorted(nodes.keys()):
        node = nodes[hash]
        print('node - ' + node.commit_hash)
        for children in sorted(node.children):
            print('child - ', end='')
            print(children)
        for parent in sorted(node.parents):
            print('parent - ', end='')
            print(parent)
        print()

def build_graph(branch_hash):
    nodes = {}
    for hash in sorted(branch_hash.keys()):
        file_name = getObjectDir() + hash[:2] + '/' + hash[2:]
        contents = zlib.decompress(open(file_name, 'rb').read())
        if (contents[:6] == b'commit'):
            stack = [hash]
            while(len(stack) != 0):
                curr = stack.pop()
                if curr not in nodes:
                    curr_node = CommitNode(curr)
                else:
                    curr_node = nodes[curr]
                parents = get_parents_of(curr)
                for parent in sorted(parents):
                    curr_node.parents.add(parent)
                    if parent not in nodes:
                        stack.append(parent)
                        parent_node = CommitNode(parent)
                    else:
                        parent_node = nodes[parent]
                    parent_node.children.add(curr)
                    nodes[parent] = parent_node
                nodes[curr] = curr_node
    return nodes

def DFS_topo(nodes):
    visited = set()
    order = []
    sources = []
    for hash in sorted(nodes):
        if len(nodes[hash].parents) == 0:
            sources.append(hash)
    for source in sources:
        if source not in visited:
            stack = [source]
        while len(stack) != 0:
            curr = stack.pop()
            if curr not in visited:
                if len(nodes[curr].parents) > 1:
                    path = []
                    new_visited = []
                    for parent in sorted(nodes[curr].parents):
                        if parent not in visited:
                            path = [parent]
                            visited.add(parent)
                            while len(path) != 0:
                                new_curr = path.pop()
                                for parent in sorted(nodes[new_curr].parents):
                                    if parent not in visited:
                                        path.append(parent)
                                new_visited.append(new_curr)
                                visited.add(new_curr)
                    order.extend(new_visited[::-1])
                for c in sorted(nodes[curr].children):
                    if c not in visited:
                        stack.append(c)
                order.append(curr)
                visited.add(curr)
    return order

def get_branches(top_level):
    branch_hash = {}
    branches = os.listdir(top_level + '/.git/refs/heads/')
    for b in sorted(branches):
        hash = open(top_level + '/.git/refs/heads/' +
                    b, 'r').read().strip('\n')
        if hash not in branch_hash:
            temp = set()
        else:
            temp = branch_hash[hash]
        temp.add(b)
        branch_hash[hash] = temp
    return branch_hash

def print_topo_order(nodes, order, branch_hash):
    i = 0
    sticky = False
    while i < len(order):
        curr_id = order[i]
        curr_node = nodes[curr_id]
        if sticky:
            sticky = False
            sticky_start = "="
            for child in sorted(curr_node.children):
                sticky_start += f'{child} '
            sticky_start = sticky_start.rstrip()
            print(sticky_start)
        print(curr_id, end='')
        if curr_id in branch_hash:
            for b in sorted(branch_hash[curr_id]):
                print(' ' + b, end='')
        print()
        if i != len(order) - 1:
            next_id = order[i+1]
            next_node = nodes[next_id]
            if curr_id not in next_node.children:
                end = ""
                for parent in sorted(curr_node.parents):
                    end += f'{parent} '
                print(end.strip()+'=')
                print()
                sticky = True
        i += 1

def topo_order_commits():
    top_level = find_root(os.getcwd())
    branch_hash = get_branches(top_level)
    nodes = build_graph(branch_hash)
    order = DFS_topo(nodes)[::-1]
    return print_topo_order(nodes, order, branch_hash)

if __name__ == '__main__':
    topo_order_commits()
```