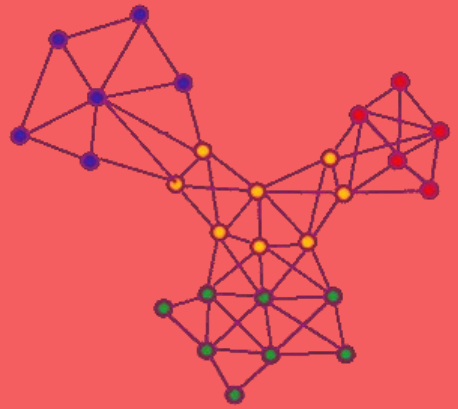
...a graph language

GIRAPHE



Dianya Jiang

Minh Truong

Tongyun Wu

Vince Pallone

Yoki Yuan

Introduction

Motivation

- Graphs appear naturally in many disciplines
- Solutions to graph problems can be extremely useful
- Writing your own graph library can be difficult and waste time
- Aiming for a language that can truly utilize and manipulate graphs with ease

Summary

- Mathematical style without the object-oriented fuss
- Native types for list, map, node, and graph built in
- Compiles to LLVM IR for cross-platform functionality
- Topological sort for scheduling and dependency checking
- Search graphs and find shortest path easily

Syntax

Comments

```
/*
    Multiple line
    comment
*/
```

Operators

```
+  -  *  /  %  <  >
== !=  <=  >=  =  !
      &&  ||
```

Keywords

```
if  else while  main
return int  bool
float  string  list
hashmap  node  edge
graph  null  void
```

Conditionals

```
if ( x == y ) {
  doSomething();
}


if ( i <= j ) {
  doSomething();
}else{
  doSomethingElse();
}
```

Loops

```
int i;
i = 0;
while( i < 3 ) {
  doSomething(i);
  i++;
}
```

Functions

```
int add(int a, int b) {
  return a + b;
}

void endl() {
  print("\n");
}

int main(){
  int x, y;
  x = 4;
  y = 12
  print( add( x, y ) );
  return 0;
}
```

List

```
int main(){
  list < int > stuff = [4];
  int i = 0;
  while (i < 10) {
    stuff.add( stuff, i );
  }

  print( stuff );
}
```

Map

```
int main(){
  hashmap < int > hash =
     {"cat" : 2, "dog" : 4};
  hash.put("mouse", 5);
  print( hash.get( "mouse" );
}
```

Implementation

Planning

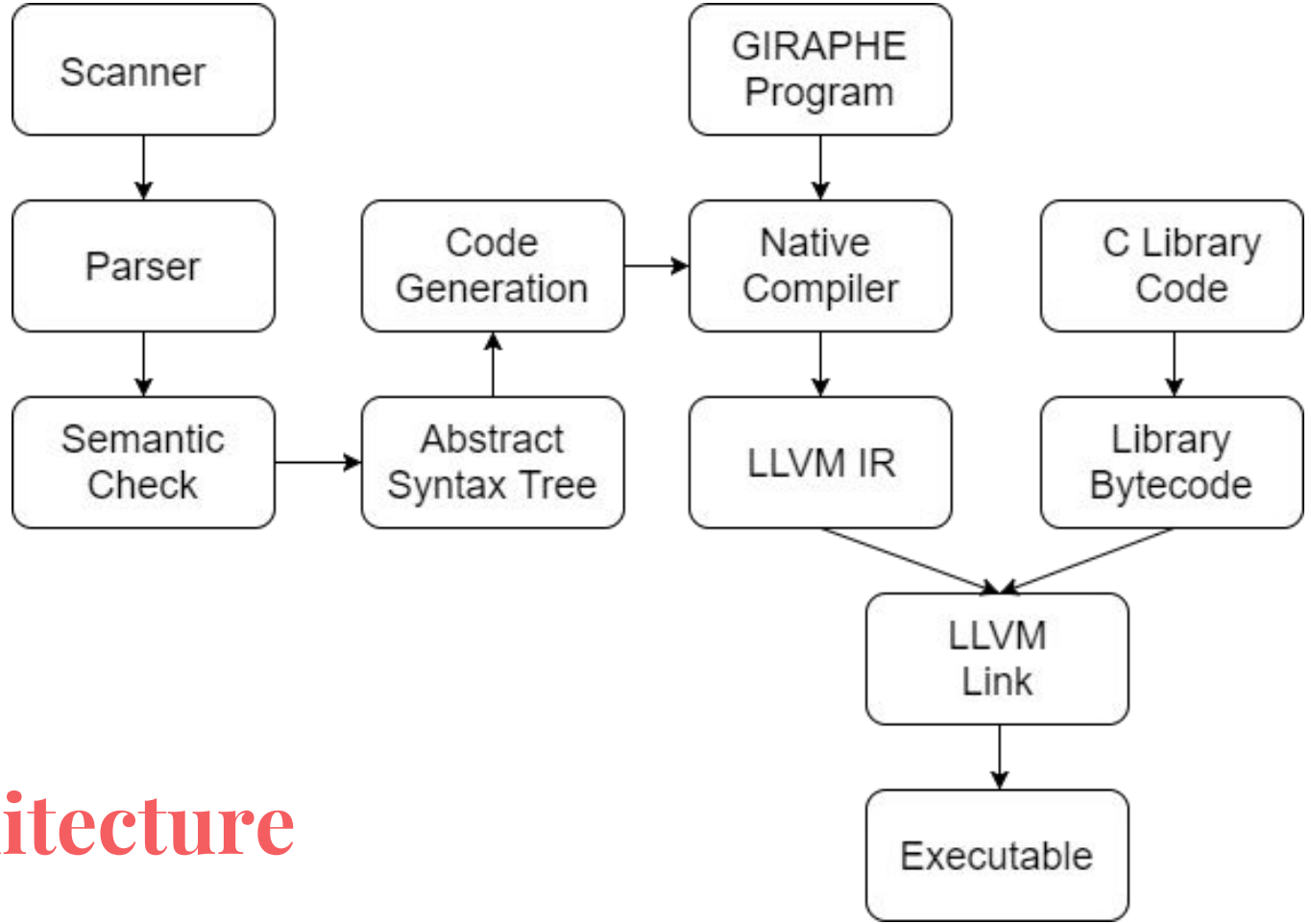
- Define exciting features we hope to implement
- Identify roles and distribute initial workloads
- Meet often to stay on the same page
- Create rough deadlines and push back unnecessary features
- Refine language usage and functionality

Roles

- Dianya Jiang: Project planning, Test case, Scanner, Parser
- Minh Truong: Scanner, Linking C Libraries, Code generation
- Tongyun Wu: AST, Parser, Code generation, Writing C Libraries
- Vince Pallone: Scanner, Linking C Libraries, Code generation
- Yoki Yuan: Semantics, Parser, Sast, Checker, Writing C Library

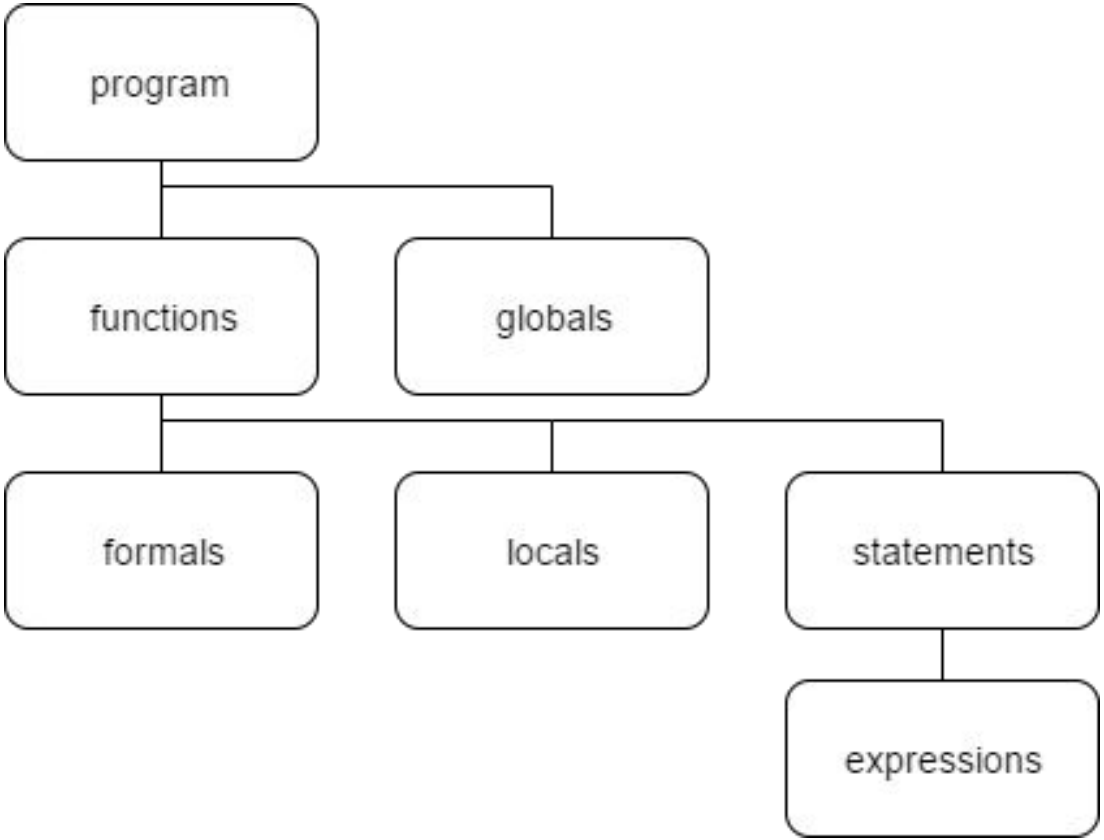
Tools

- Slack: Group communication
- GitHub: Public code repo and version control kept us sane
- Ubuntu: Consistent operating system for testing
- VirtualBox: Software for running virtual environment
- LLVM: The only reason we have a language



Architecture

Abstract Syntax Tree



Lessons Learned

- Start early with something small that works
- Use branches and add only one feature at a time
- Start with tests, then make them work
- Aim for a small set of highly focused features

Features

C Library - List and Map

- List
  - Initialize List, Add, Get, Set, Contains, Remove Data, Push, Get Size, Print List
  - *Special: Concat List -> [1,2,3] + [4,5,6] = [1,2,3,4,5,6]
  - Remove Data -> [1,2,3,4,5] - 5 = [1,2,3,4]
- HashMap
  - Put, Get, Contains, Remove, Keys,

C Library - Queue and Minheap

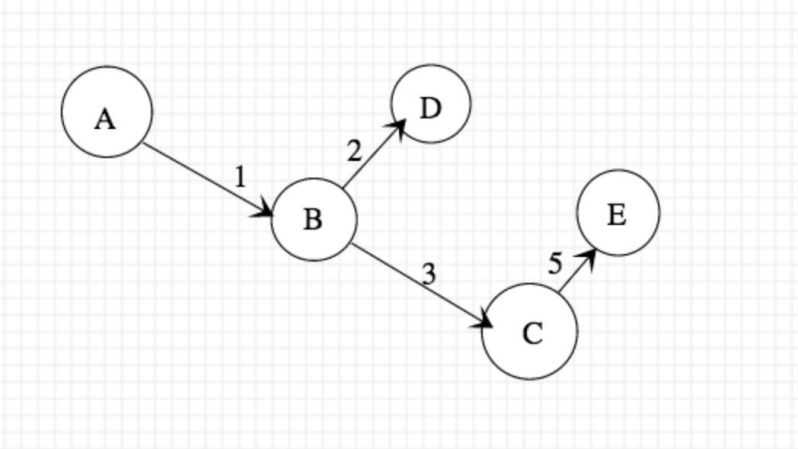
- Queue
    - Initialize, Push Back, Pop Front, get Size, print
- Minheap
    - Initialize, Swap, Compare, Heapify, Insert, Get Min Value, Decrease Priority, Print

C Library - Node and Edge

- Node
  - Create Node , Print Node, Get Node Value,* Set Visited, * Get Visited
  - Add Node, Has Node, Remove Node
- Edge
  - Create Edge, Print Edge, Get Edge Weight
  - Add Edge, Contains Edge, Remove Edge

C Library - Graph

- Graph
  - Graph Define: a->1$b->3$c + b->2$d + c -> 5$e
  - Graph Manipulation: Link Graph, Split Graph, Copy Graph, Get All Nodes, Set All Node Unvisited
- * Cool Functions
  - Breadth First Search, Depth First Search
  - Dijkstra Algorithm



Node & Edge Functions

4 FEATURES

```
node a = node("A");
node b = node("B");
node c = node("C");
node d = node("D");
node e = node("E");
node f = node("F");

node m = node("M");
node h = node("H");

node x = node("X");
node y = node("Y");
node z = node("Z");


print(" A -> E + A -> B -> D + A -> C ->F ");

graph g = a -> 1$e + a -> 2$b -> 3$d + a -> 4$c -> 5$f;
print(g);

print("Get Graph Size:");
print(g.size());

print("Add Nodes to Graph");
g.addNode(m);
print(g.size());
print(g);

print("Judge whether Nodes exist");
print(g.hasNode(x));
print(g.hasNode(a));

print("Add Edge to Graph, Support new Nodes");
g.addEdge(e, f, 5);
g.addEdge(a, z, 10);
print(g.size());
list<node> nodes = g.getAllNodes();
print(nodes);
print(g);

print("Judge whether Edges exist between two Nodes");
print(g.hasEdge(c,d));
print(g.hasEdge(a,b));
```

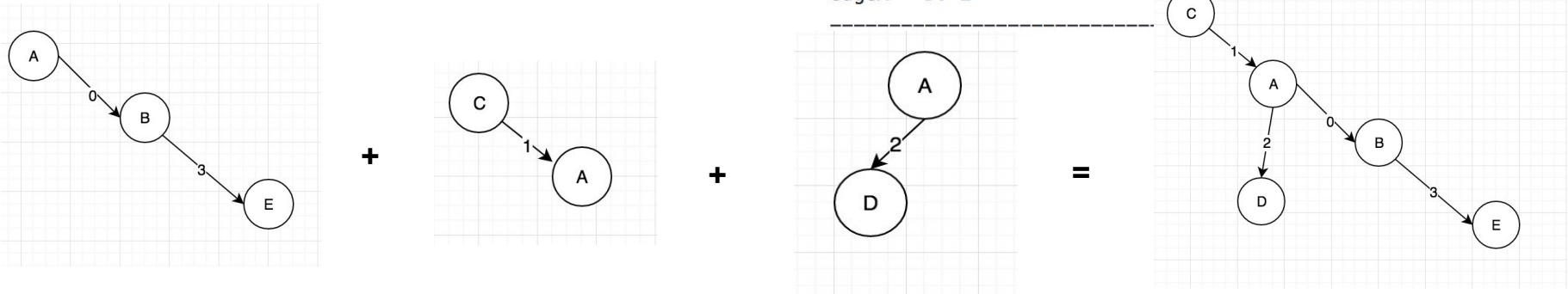
```
 A -> E + A -> B -> D + A -> C ->F
----------------------------------------
Nodes:
node A
node E
node B
node D
node C
node F

Edges:
edgeA ->E: 1
edgeB ->D: 3
edgeA ->B: 2
edgeC ->F: 5
edgeA ->C: 4
----------------------------------------
Get Graph Size:
6
Add Nodes to Graph
7
----------------------------------------
Nodes:
node A
node E
node B
node D
node C
node F
node M

Edges:
edgeA ->E: 1
edgeB ->D: 3
edgeA ->B: 2
edgeC ->F: 5
edgeA ->C: 4
----------------------------------------
```

```
----------------------------------------
Judge whether Nodes exist
false
true
Add Edge to Graph, Support new Nodes
8
[node A
node E
node B
node D
node C
node F
node M
node Z
]
----------------------------------------
Nodes:
node A
node E
node B
node D
node C
node F
node M
node Z

Edges:
edgeA ->E: 1
edgeB ->D: 3
edgeA ->B: 2
edgeC ->F: 5
edgeA ->C: 4
edgeE ->F: 5
edgeA ->Z: 10
----------------------------------------
Judge whether Edges exist between two Nodes
false
true
```

Graph Link

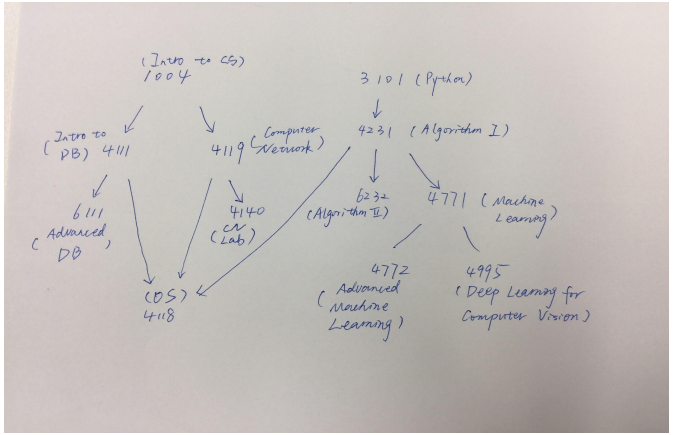
```
node a = node("A");
node b = node("B");
node c = node("C");
node d = node("D");
node e = node("E");

print("Using + to link graph");

print("Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D");
print("Shared nodes: A");

graph g = a -> 0$b ->3$e +  c -> 1$a + a -> 2$d;

print(g);
```

```
Using + to link graph
Link graphs: A -> 0$B ->3$E +  C -> 1$A + A -> 2$D
Shared nodes: A
-------------------------------------
Nodes:
node B
node E
node A
node C
node D

Edges:
edgeB ->E: 3
edgeA ->B: 0
edgeC ->A: 1
edgeA ->D: 2
-------------------------------------
```



Topological Sort

```
node a = node("COMS 1004: Introduction to Computer Science and Programming in Java");
node b = node("COMS 4111: Introduction to Databases");
node c = node("COMS 6111: Advanced Database Systems");
node d = node("CSEE 4119: Computer Networks");
node e = node("CSEE 4140: Networking Laboratory");
node f = node("COMS 4118: Operating Systems");
node g = node("COMS 3101: Programming Language Python");
node h = node("CSOR 4231: Analysis of Algorithms I");
node i1 = node("COMS 6232: Analysis of Algorithms II");
node j = node("COMS 4771: Machine Learning");
node k = node("COMS 4772: Advanced Machine Learning");
node l = node("COMS 4995: Deep Learning for Computer Vision");

graph ga = a -> b -> f + b -> c + a -> d -> f + d -> e + h -> f + g -> h -> j -> l + h -> i1 + j -> k;

int i=0;
list<node> n = ga.getAllNodes();
list<node> tmp = n;
list<node> res = n;
int stmp = 0;
ga.setAllUnvisited();

while(i<ga.size()){
        if(n.get(i).isVisited()){
                i = i + 1;
        }else{
                tmp = ga.dfs(n.get(i));
                stmp = tmp.size();
                while(stmp > 0) {
                        res.add(tmp.get(stmp-1));
                        stmp = stmp - 1;
                }
                i = i + 1;
        }
}

while(i > 0) {
        print(res.get(i-1+ga.size()));
        i=i-1;
}
```

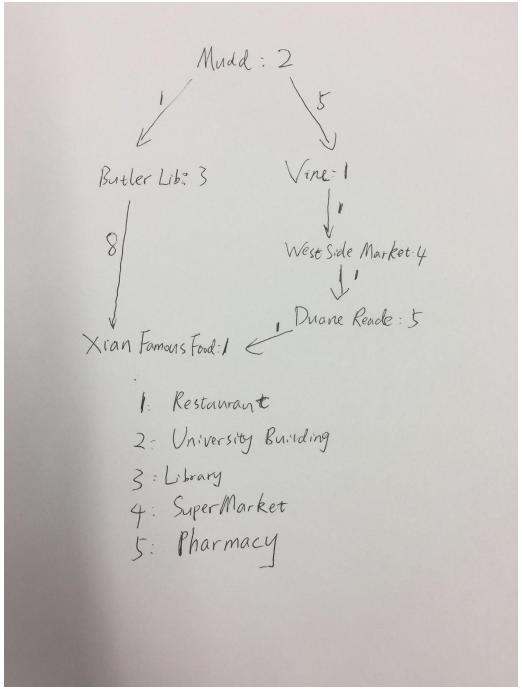



```
node COMS 3101: Programming Language Python
node CSOR 4231: Analysis of Algorithms I
node COMS 6232: Analysis of Algorithms II
node COMS 4771: Machine Learning
node COMS 4772: Advanced Machine Learning
node COMS 4995: Deep Learning for Computer Vision
node COMS 1004: Introduction to Computer Science and Programming in Java
node CSEE 4119: Computer Networks
node CSEE 4140: Networking Laboratory
node COMS 4111: Introduction to Databases
node COMS 6111: Advanced Database Systems
node COMS 4118: Operating Systems
```

Filter

- Find specific types of locations by using our filter
- Will generate out the shortest path between the source and those specific locations

```
node a = node("Mudd");
node b = node("Vine");
node c = node("Xian Famous Food");
node d = node("Butler Library");
node e = node("WestSide Market");
node f = node("Duane Reade");


graph gh = a -> 1$d ->8$c + a -> 1$d -> 2$b -> 1$e -> 1$f -> 1$c + a ->5$b;
list<node> path = [a];
node cur = a;
int curVal = 0;
print("1 : Restaurant");
print("2 : Unversity Building");
print("3 : Library");
print("4 : SuperMarket");
print("5 : Pharmacy");
print("");
hashmap<int> hmap = { a : 2, b : 1, c : 1, d : 3, e : 4, f : 5};

void filter(node sour, int target) {
    print("-- try to find places of target type --");
    print(target);
    list<node> nodes = gh.getAllNodes();
    int size = nodes.size();
    int i = 0;
    while (i < size) {
        cur = nodes.get(i);
        curVal = hmap.get(cur);
        if (curVal == target) {
            print("");
            print("--------- have found target ------------");
            print("---------     target      --------------");
            print(cur);
            print("--------- shortest path to target  -----");
            path = gh.dijkstra(sour, cur);
            print(path);
            print("");
            print("");
        }
        i = i + 1;
    }
}
filter(a, 1);
```




```
1 : Restaurant
2 : Unversity Building
3 : Library
4 : SuperMarket
5 : Pharmacy

-- try to find places of target type --
1

--------- have found target ------------
---------     target      --------------
node Xian Famous Food
--------- shortest path to target  -----
[node Mudd
node Butler Library
node Vine
node WestSide Market
node Duane Reade
node Xian Famous Food
]


--------- have found target ------------
---------     target      --------------
node Vine
--------- shortest path to target  -----
[node Mudd
node Butler Library
node Vine
]
```

"DO OR DO NOT.

THERE IS NO TRY."

THANK YOU