GOBLAN

A Graphical Object Language

Yunsung Kim, Sean Garvey, Sameer Lal, Jee Hyun Wang

Table of Contents

Graph Programming in Conventional Languages

- **Conventional Languages** implementation of graph programming
  - Can be tedious, time consuming
  - Usually try to manipulate the graph as a whole, by using lists and arrays, and many loops.

```
dijkstra(graph *g, int start)           /* WAS prim(g,start) */
{
   int i,j;                             /* counters */
   bool intree[MAXV];                   /* is vertex in the tree yet? */
   int distance[MAXV];                  /* vertex distance from start */
   int v;                               /* current vertex to process */
   int w;                               /* candidate next vertex */
   int weight;                          /* edge weight */
   int dist;                            /* shortest current distance */

   for (i=1; i<=g->nvertices; i++) {
           intree[i] = FALSE;
           distance[i] = MAXINT;
           parent[i] = -1;
   }
   distance[start] = 0;
   v = start;

   while (intree[v] == FALSE) {
       intree[v] = TRUE;
       for (i=0; i<g->degree[v]; i++) {
            w = g->edges[v][i].v;
            weight = g->edges[v][i].weight;
/* CHANGED */    if (distance[w] > (distance[v]+weight)) {
/* CHANGED */            distance[w] = distance[v]+weight;
                         parent[w] = v;
                 }
       }
       v = 1;
       dist = MAXINT;
       for (i=2; i<=g->nvertices; i++)
            if ((intree[i]==FALSE) && (dist > distance[i])) {
                    dist = distance[i];
                    v = i;
            }
   }
}
```

Graph Programming in GOBLAN

- Domain specific language for constructing and manipulating complex structured graphs
- Introduces a new paradigm of graph programming
  - **Message Passing**
- Enables the communication between individual nodes

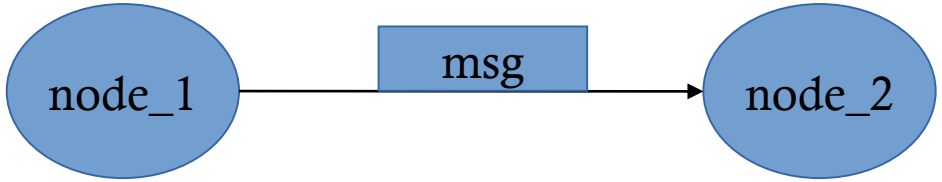
Message Passing

- Algorithms consists of two parts

  1) Communicating data between nodes through messages

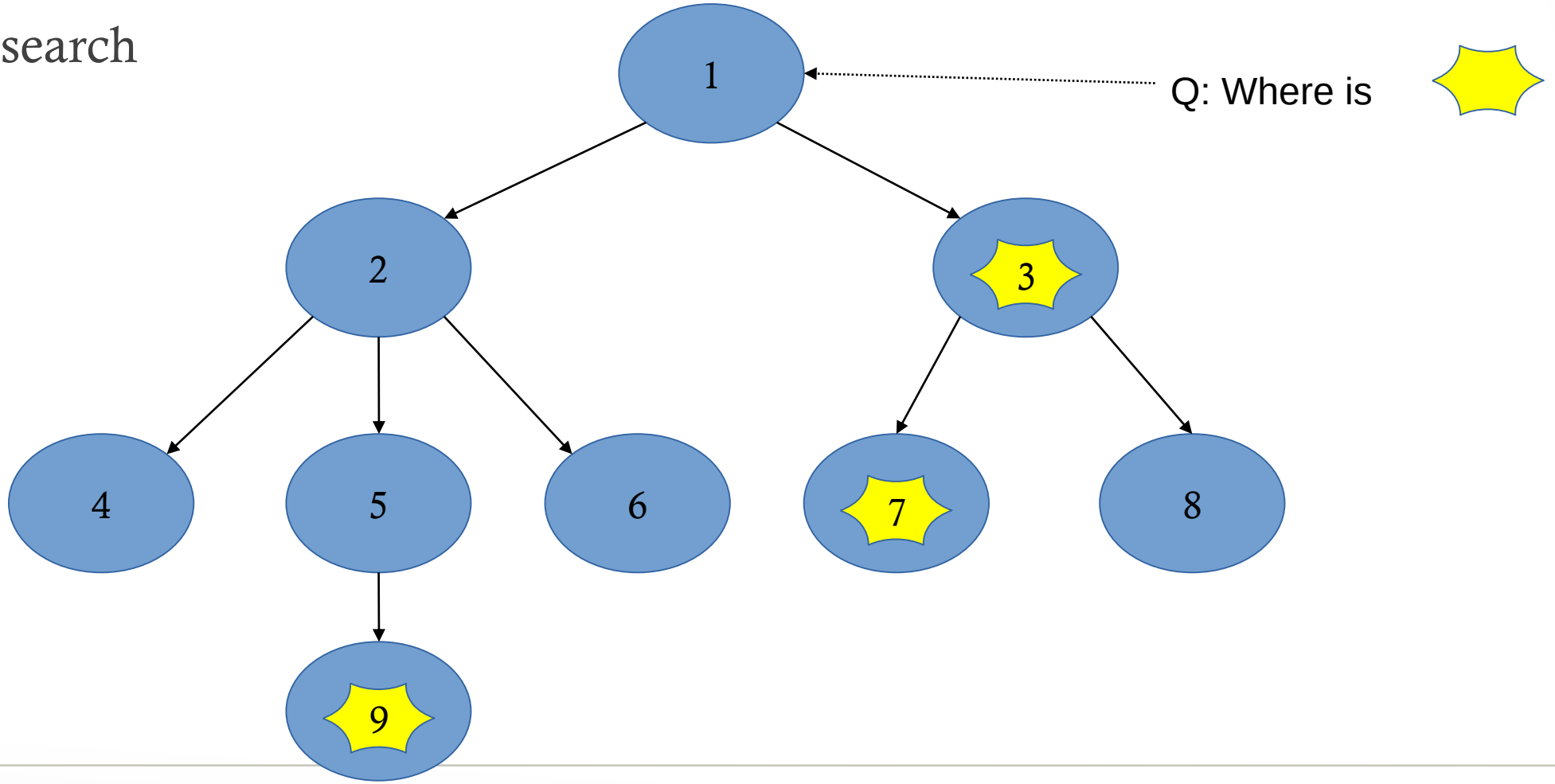
  2) Processing messages to update current node data

- Many graph algorithms fall into the paradigm of message passing



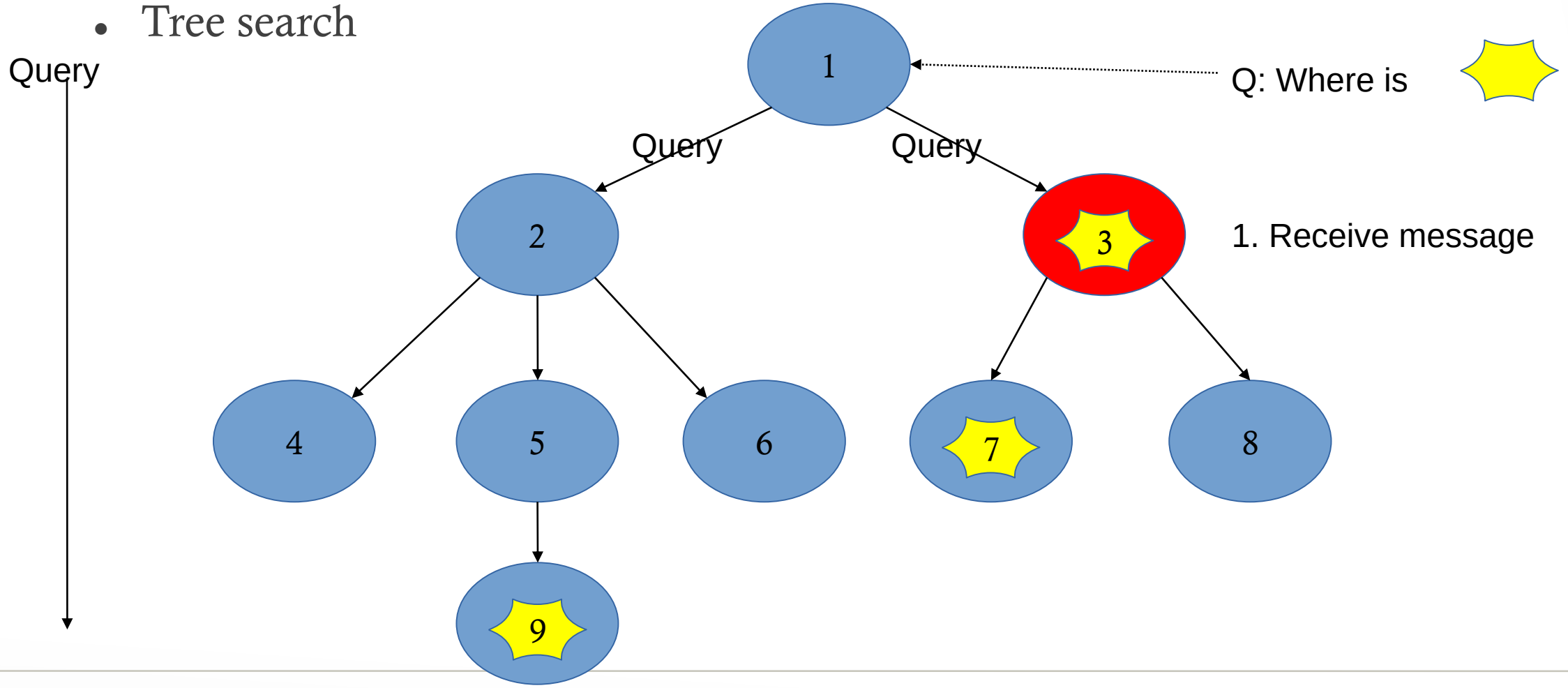
Message Passing

- Tree search



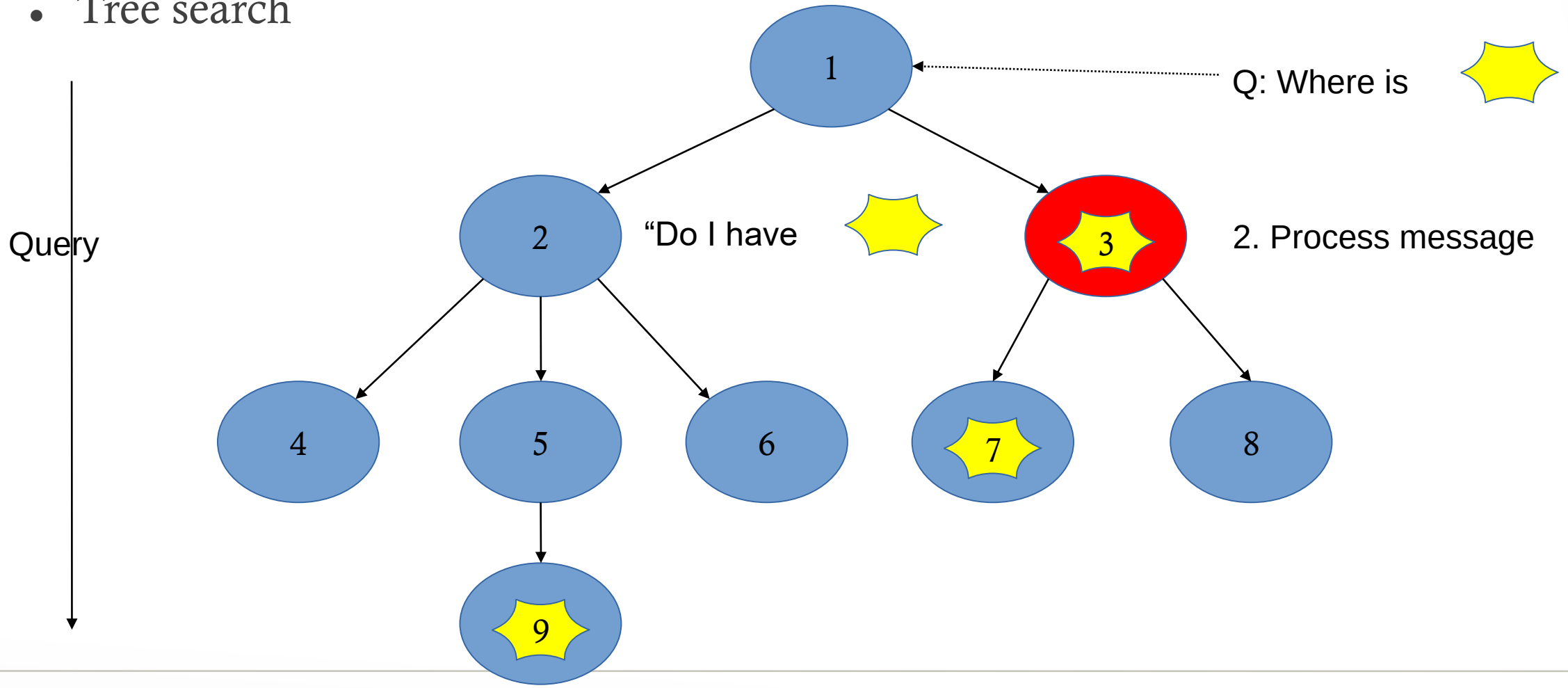
Message Passing

- Tree search



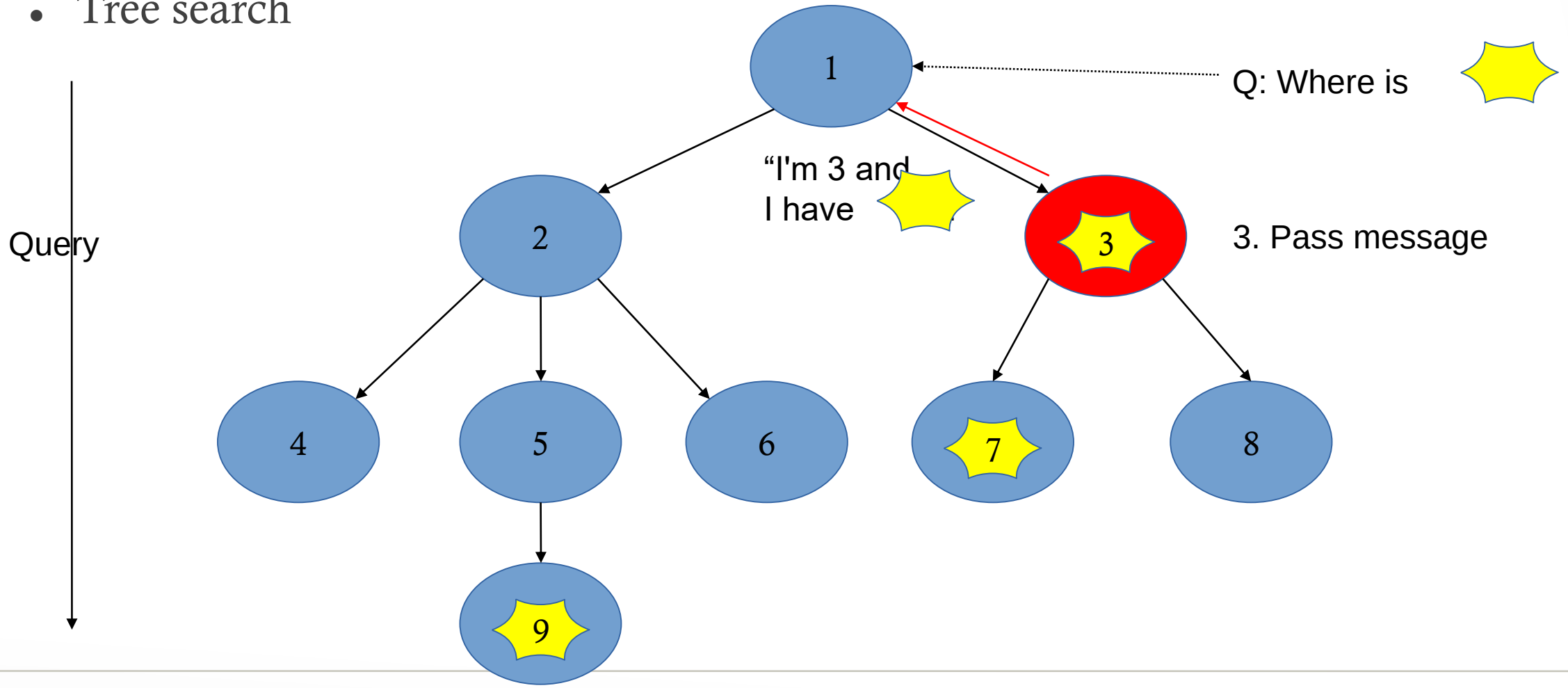
Message Passing

- Tree search



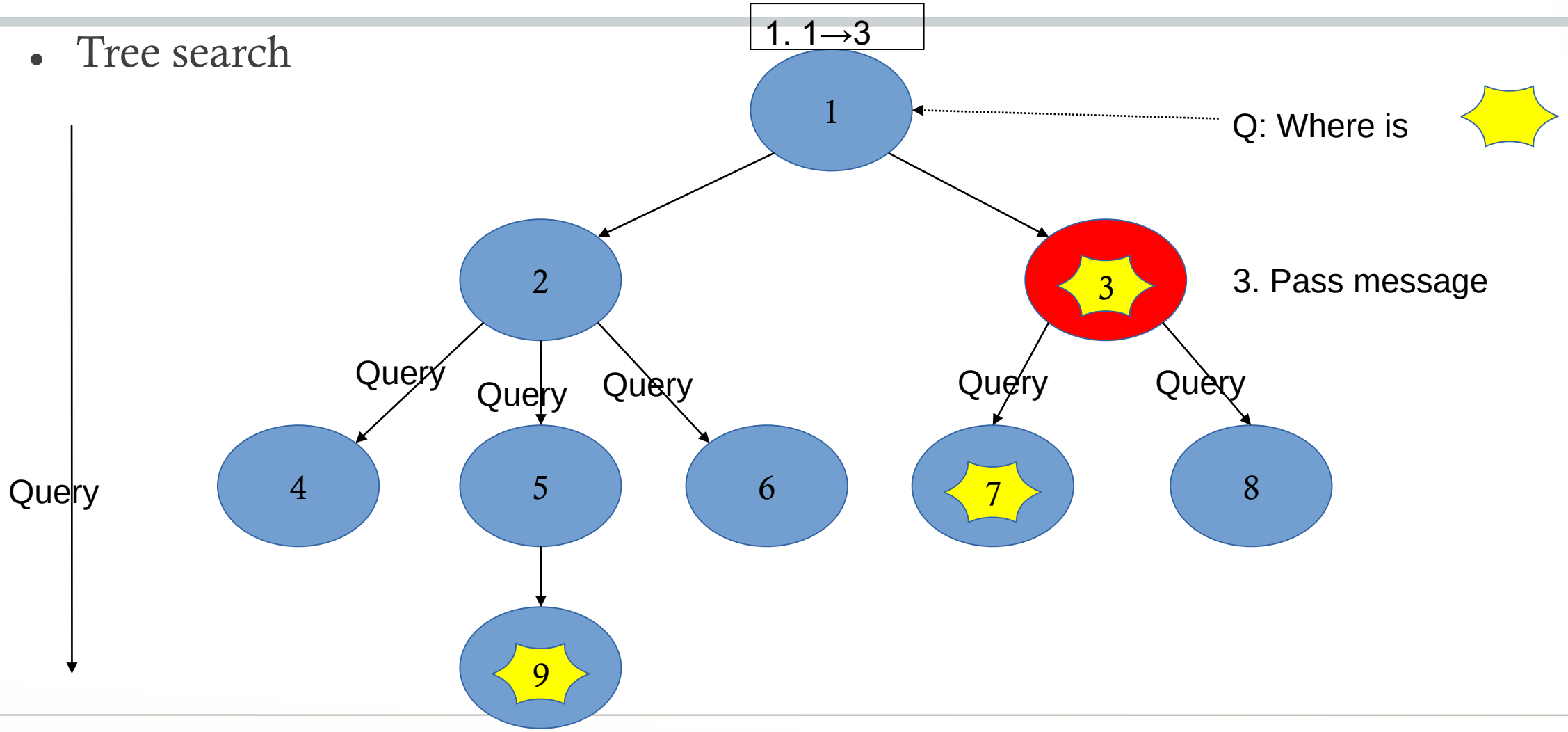
Message Passing

- Tree search



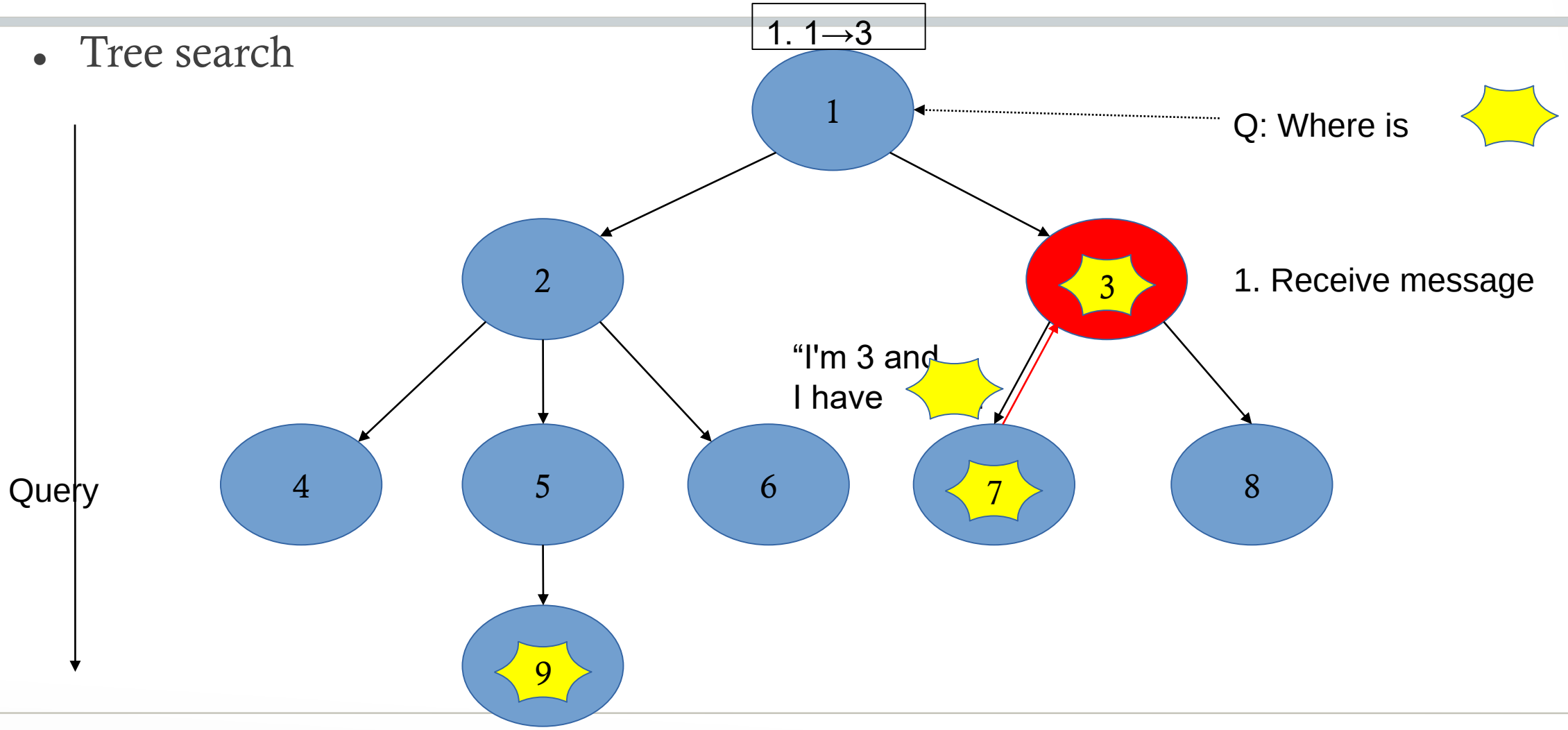
Message Passing

- Tree search



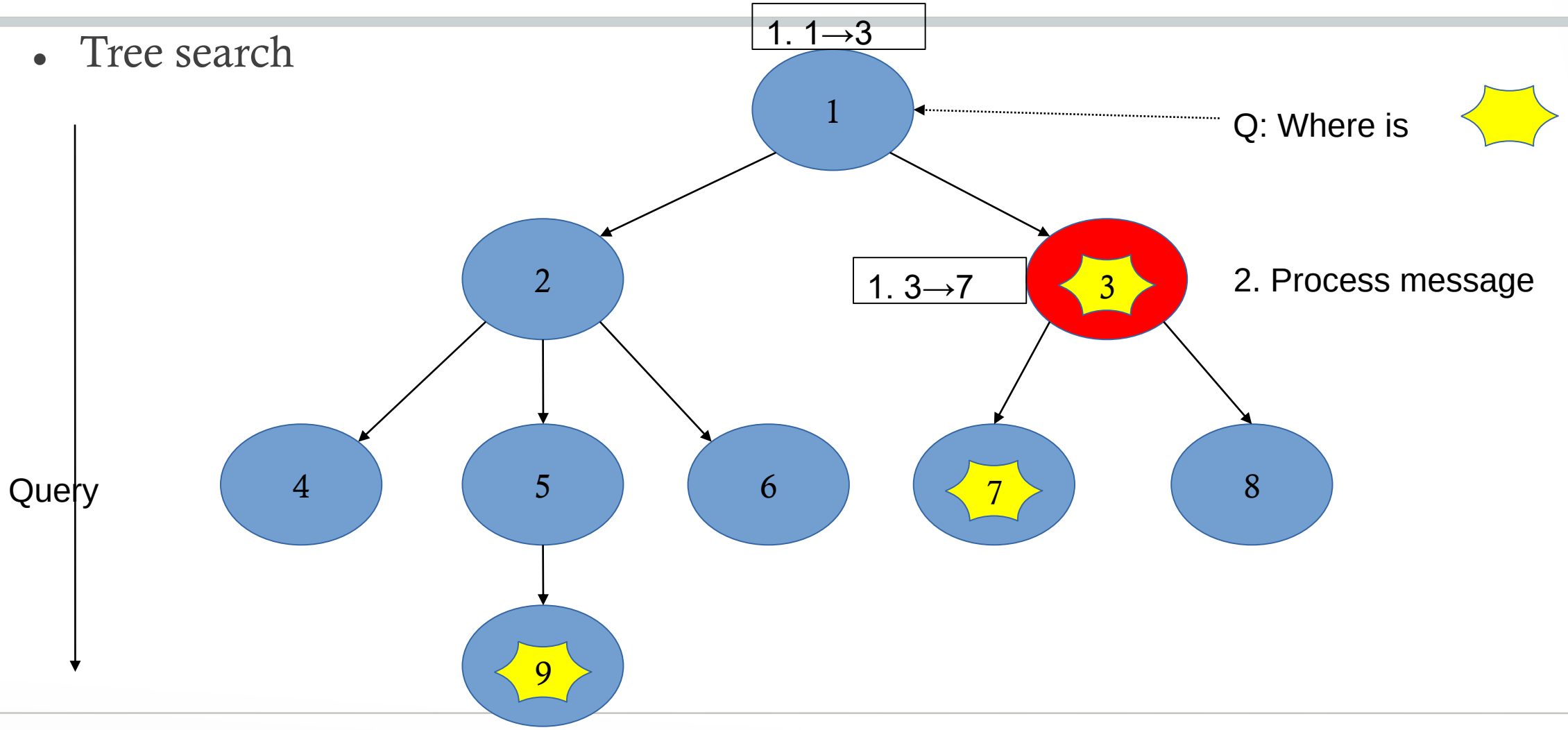
Message Passing

- Tree search



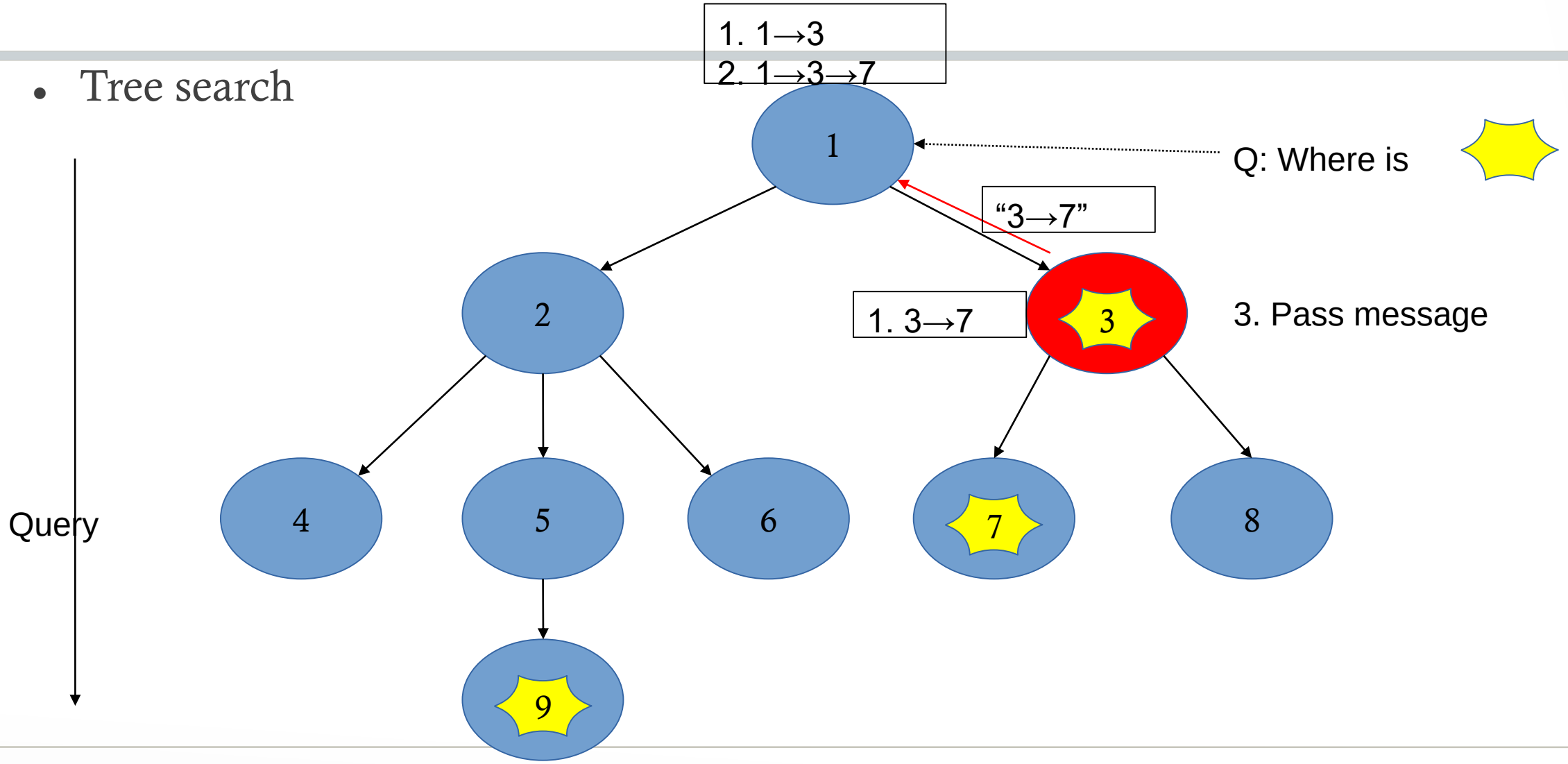
Message Passing

- Tree search



Message Passing

- Tree search



Strength of GOBLAN

1. Intuitive node declaration
2. Object function:

   run { node } (arg1, arg2, ….)
3. Message passing:

   pass pkt -> chld
4. Graph construction:

   new graph(node:A)[|…|]
5. Compiles to LLVM

Strength of GOBLAN

1. Intuitive node declaration

```
node:NodeType {
        data {
                /* data specification */
        }
        edge {
                /* edge attribute specification */
        }
        pack {
                /* message attribute specification */
        }
        type do (type arg1, type arg2, ...) {
                 /* asynchronous function definition
                (Can be called anywhere in the program)
*/
        }
        catch {
                 /* synchronous function definition
                (Invoked upon receiving a message) */
        }
}
```

Strength of GOBLAN

2. Object function: run { node } (arg1, arg2, ….)

- Asynchronous object function
- Allows simple invocation of graph algorithms
- Invoked anywhere in the program

Strength of GOBLAN

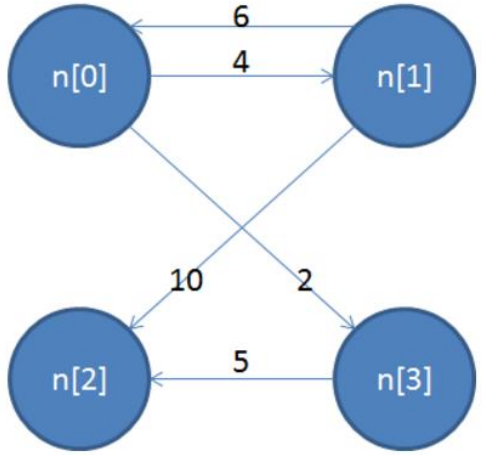
3. Message passing: pass pkt -> chld

- Simple keyword : chld | prnt |prnt_chld | chld_prnt
- High level encapsulation of intricate recursion

Strength of GOBLAN

4. Graph construction: new graph(node:A)[|…|]

- Interconnecting nodes through statements that are easy to understand



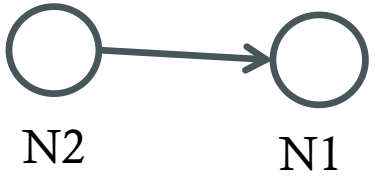

```
list node:A n;
int i;

n = new node:A[| |];
for (i = 0; i < 4; i = i + 1)
        n += new node:A(...);

new graph(node:A)[|
        edge[n[0]->n[1]](4);
        edge[n[0]->n[3]](2);
        edge[n[1]->n[0]](6);
        edge[n[1]->n[2]](10);
        edge[n[3]->n[2]](5);
|];
```

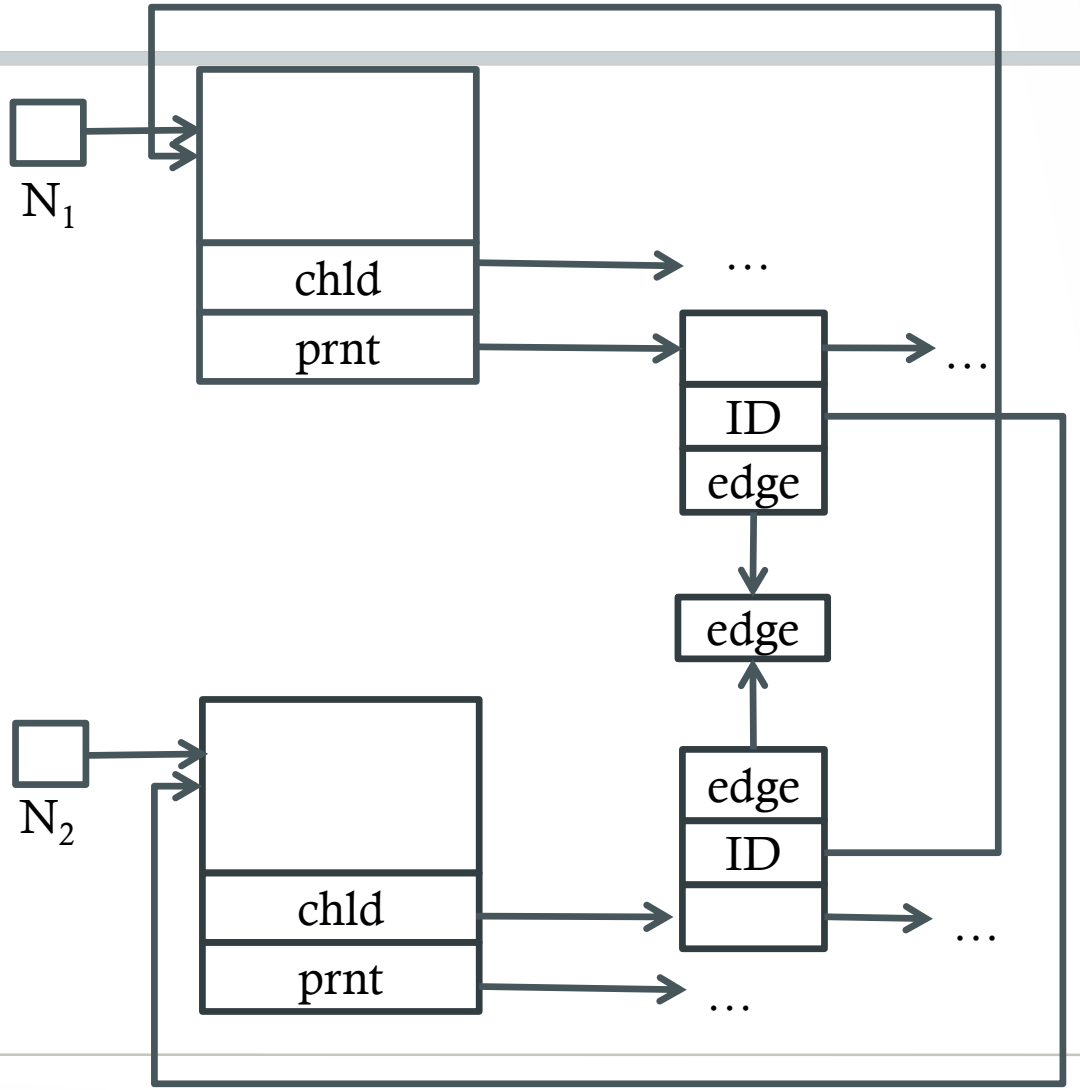
Strength of GOBLAN

5. Compiles to LLVM

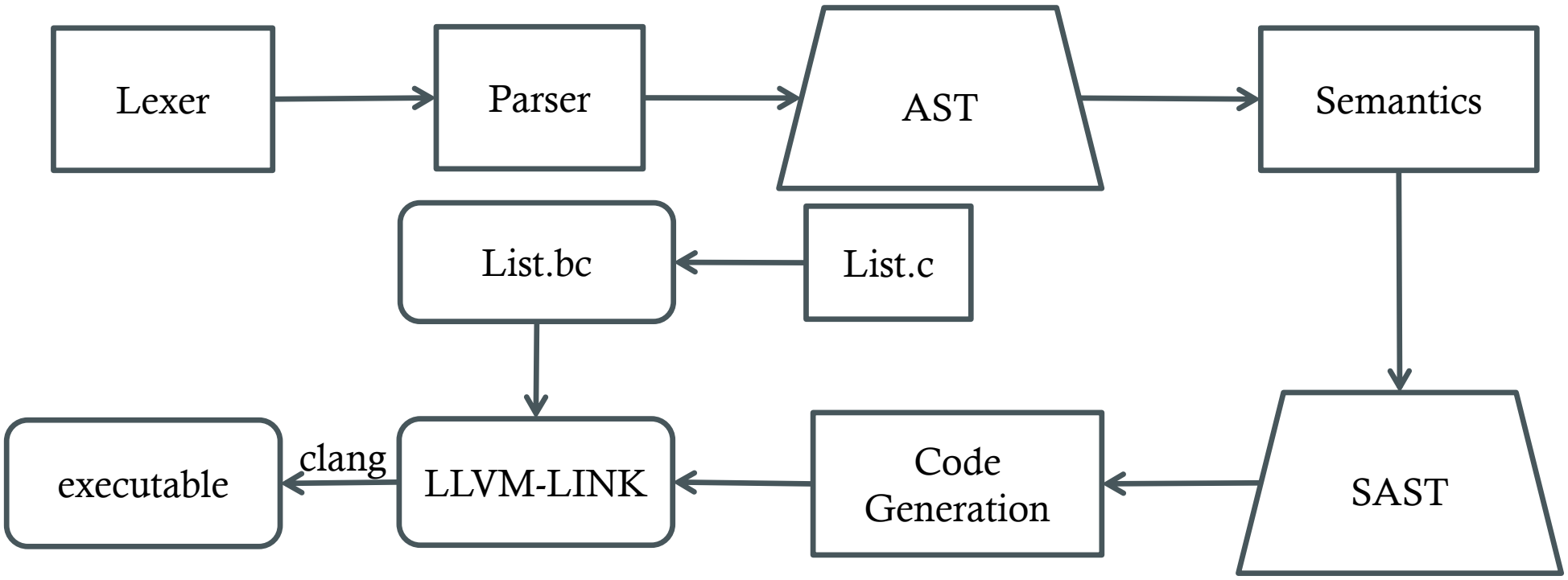
Structure of GOBLAN

- How graphs are represented





Structure of GOBLAN (compiling)

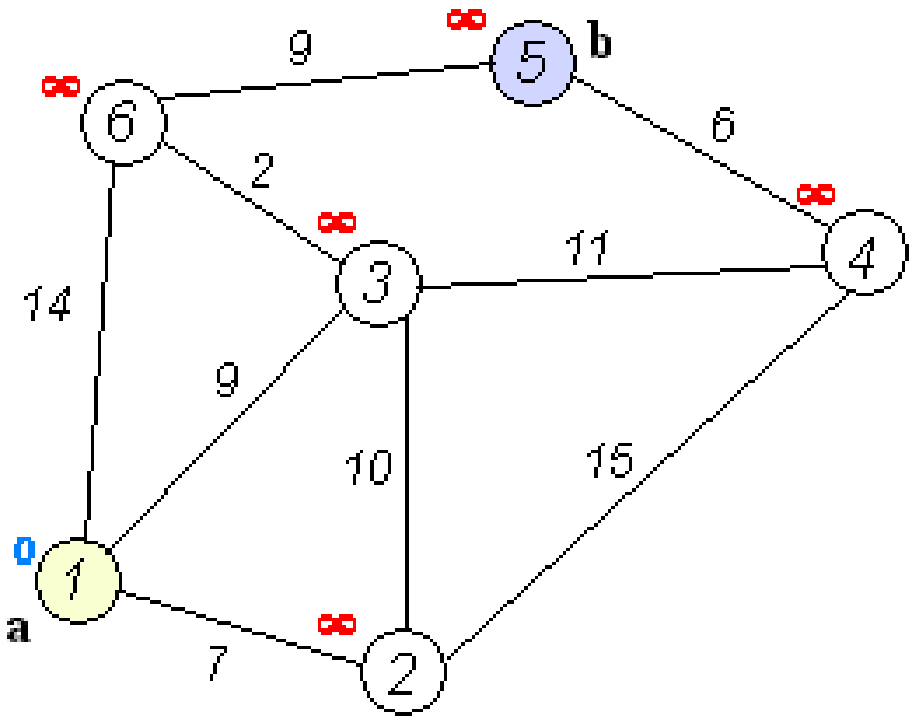


DEMO

- Dijkstra's algorithm
- Tree Search

DEMO

- **Dijkstra's algorithm**



DEMO

- **Tree Search**