People are good at programming with collections.

(at least, better than with streams)

Gist: collection-oriented programming language,
the system manages changes to collections.

```
fn your_prog: [D] -> [R] = /* .. */;

for t in times {
    let output[t] = your_prog(input[t]);
}
```


d_input: Stream<(Data, Time, isize)>


d_output: Stream<(Data, Time, isize)>

```
let nodes = /* pairs (node, bool) */;
let edges = /* pairs (node, node) */;
```

```
nodes.join(edges)     // one hop neighbors
     .concat(nodes)   // plus original nodes
     .distinct()      // extended neighborhood
```

```
for t in times {
    nodes.insert(..);
    edges.insert(..);
}
```

People are good at programming with collections.

(at least, better than with streams)

Gist: collection-oriented programming language,
the system manages changes to collections.

```
fn your_prog: [D] -> [R] = /* .. */;

for t in times {
    let output[t] = your_prog(input[t]);
}
```

d_output: Stream<(Data, Time, isize)>