Distributed and Parallel (Extended) - Assignment 2

Group 10

Mason Cusack (1414624), Vadim Makarenkov (1566767), Callum Tolley (1552710), and David Somers (1567268)

We developed a program that achieves all the goals set in the assignment.

It also takes two optional arguments:
`--verbose`: reports more output about the algorithm
`--breadth-first`: performs the breadth-first algorithm, where the token is sent through each link twice.

In its original implementation the code was more complex. For example, the initiator and normal nodes were different functions. Now, there is one entry function, `init_node`, which waits for the initial broadcast of neighbour PIDs, and a recursive main function that the nodes run named `node`.

We combined the node functions as they shared much of the same code: now the only difference is that an initiator node starts out with a parent, wheras a normal node doesn't (i.e. it starts out with the special atom `none`). This is possible because the parent is only set on a node if it doesn't have a parent already.

It was decided that every message would be of a uniform format: it would be a list, where the first element was always an `atom` that formed the ID of the message, and the rest were arguments.

ID	Args	Notes
`neighbour_ids`	`[Neighbours]`	Sent initially to inform nodes about the PIDs and names of it's neighbours. `Neighbours` is an array of tuples of the form `{Pid, Name}`.
`token`	`[Token, From]`	Main message used in the algorithm to distribute the token around. `Token` is the token, but reversed for efficiency's sake (prepending is more efficient than appending in Erlang). `From` contains which PID the message is from.
`exit`	*N/A*	Signals the node to exit.

Sample Output

Normal output

```
$ escript tarry.erl < input.txt
p q t r t s u s t q p
```

Breadth first (non-extended part)

```
$ escript tarry.erl --breadth-first < input.txt
p s t q s q p q r t u s u t r q t s p
```

`input_matrix.txt`

`input_matrix.txt` is a 3x3 matrix of nodes labelled from 'a' to 'i', with adjacent nodes (including diagonals) connected.

```
$ escript tarry.erl < input_matrix.txt
e a d h f i f b c b f h g h d a e
$ escript tarry.erl --breadth-first < input_matrix.txt
e g h f c b e c e a d g d e d b f e b d h i f b a b c f i e i h d a e h e f h g e
```

Depth first (Verbose)

```
$ escript tarry.erl --verbose < input.txt
Nodes: [{"p",["q","s"]},
        {"q",["p","s","t","r"]},
        {"r",["q","t"]},
        {"s",["p","q","t","u"]},
        {"t",["q","r","u","s"]},
        {"u",["s","t"]}]
Initiator: "p"
<0.32.0> p: Waiting for neighbour IDs...
<0.33.0> q: Waiting for neighbour IDs...
<0.34.0> r: Waiting for neighbour IDs...
<0.35.0> s: Waiting for neighbour IDs...
<0.36.0> t: Waiting for neighbour IDs...
<0.37.0> u: Waiting for neighbour IDs...
<0.32.0> p: Received neighbours: q s
<0.33.0> q: Received neighbours: p s t r
<0.34.0> r: Received neighbours: q t
<0.35.0> s: Received neighbours: p q t u
<0.36.0> t: Received neighbours: q r u s
<0.37.0> u: Received neighbours: s t
<0.32.0> p: remaining neighbours: [{<0.35.0>,"s"}]
<0.32.0> p: got token, sending to neighbour q (<0.33.0>):
<0.33.0> q: remaining neighbours: [{<0.36.0>,"t"},{<0.35.0>,"s"}]
<0.33.0> q: got token, sending to neighbour r (<0.34.0>): p
<0.34.0> r: remaining neighbours: []
<0.34.0> r: got token, sending to neighbour t (<0.36.0>): q p
```

```
<0.36.0> t: remaining neighbours: [{<0.37.0>,"u"}]
<0.36.0> t: got token, sending to neighbour s (<0.35.0>): r q p
<0.35.0> s: remaining neighbours: []
<0.35.0> s: got token, sending to neighbour u (<0.37.0>): t r q p
<0.37.0> u: no neighbours, returning token to parent (<0.35.0>): s t r q p
<0.35.0> s: no neighbours, returning token to parent (<0.36.0>): u s t r q p
<0.36.0> t: no neighbours, returning token to parent (<0.34.0>): s u s t r q p
<0.34.0> r: no neighbours, returning token to parent (<0.33.0>): t s u s t r q p
<0.33.0> q: no neighbours, returning token to parent (<0.32.0>): r t s u s t r q p
<0.32.0> p: no neighbours, returning token to parent (<0.2.0>): q r t s u s t r q p
p q r t s u s t r q p
```

Breadth first (Verbose)

```
$ escript tarry.erl --breadth-first --verbose < input.txt
Nodes: [{"p",["q","s"]},
        {"q",["p","s","t","r"]},
        {"r",["q","t"]},
        {"s",["p","q","t","u"]},
        {"t",["q","r","u","s"]},
        {"u",["s","t"]}]
Initiator: "p"
<0.32.0> p: Waiting for neighbour IDs...
<0.33.0> q: Waiting for neighbour IDs...
<0.34.0> r: Waiting for neighbour IDs...
<0.35.0> s: Waiting for neighbour IDs...
<0.36.0> t: Waiting for neighbour IDs...
<0.37.0> u: Waiting for neighbour IDs...
<0.32.0> p: Received neighbours: q s
<0.33.0> q: Received neighbours: p s t r
<0.34.0> r: Received neighbours: q t
<0.35.0> s: Received neighbours: p q t u
<0.36.0> t: Received neighbours: q r u s
<0.37.0> u: Received neighbours: s t
<0.32.0> p: remaining neighbours: [{<0.33.0>,"q"}]
<0.32.0> p: got token, sending to neighbour s (<0.35.0>):
<0.35.0> s: remaining neighbours: [{<0.33.0>,"q"},{<0.37.0>,"u"}]
<0.35.0> s: got token, sending to neighbour t (<0.36.0>): p
<0.36.0> t: remaining neighbours: [{<0.37.0>,"u"},{<0.34.0>,"r"}]
<0.36.0> t: got token, sending to neighbour q (<0.33.0>): s p
<0.33.0> q: remaining neighbours: [{<0.32.0>,"p"},{<0.34.0>,"r"}]
<0.33.0> q: got token, sending to neighbour s (<0.35.0>): t s p
<0.35.0> s: remaining neighbours: [{<0.37.0>,"u"}]
<0.35.0> s: got token, sending to neighbour q (<0.33.0>): q t s p
<0.33.0> q: remaining neighbours: [{<0.34.0>,"r"}]
<0.33.0> q: got token, sending to neighbour p (<0.32.0>): s q t s p
<0.32.0> p: remaining neighbours: []
<0.32.0> p: got token, sending to neighbour q (<0.33.0>): q s q t s p
<0.33.0> q: remaining neighbours: []
<0.33.0> q: got token, sending to neighbour r (<0.34.0>): p q s q t s p
<0.34.0> r: remaining neighbours: []
<0.34.0> r: got token, sending to neighbour t (<0.36.0>): q p q s q t s p
```

```
<0.36.0> t: remaining neighbours: [{<0.34.0>,"r"}]
<0.36.0> t: got token, sending to neighbour u (<0.37.0>): r q p q s q t s p
<0.37.0> u: remaining neighbours: []
<0.37.0> u: got token, sending to neighbour s (<0.35.0>): t r q p q s q t s p
<0.35.0> s: remaining neighbours: []
<0.35.0> s: got token, sending to neighbour u (<0.37.0>): u t r q p q s q t s p
<0.37.0> u: no neighbours, returning token to parent (<0.36.0>): s u t r q p q s q
t s p
<0.36.0> t: remaining neighbours: []
<0.36.0> t: got token, sending to neighbour r (<0.34.0>): u s u t r q p q s q t s p
<0.34.0> r: no neighbours, returning token to parent (<0.33.0>): t u s u t r q p q
s q t s p
<0.33.0> q: no neighbours, returning token to parent (<0.36.0>): r t u s u t r q p
q s q t s p
<0.36.0> t: no neighbours, returning token to parent (<0.35.0>): q r t u s u t r q
p q s q t s p
<0.35.0> s: no neighbours, returning token to parent (<0.32.0>): t q r t u s u t r
q p q s q t s p
<0.32.0> p: no neighbours, returning token to parent (<0.2.0>): s t q r t u s u t r
 q p q s q t s p
p s t q s q p q r t u s u t r q t s p
```