School of Computing

FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

<Full title of Project>

<Full Name of Author>

Submitted in accordance with the requirements for the degree of
<Name of Degree>

<Session>

The candidate confirms that the following have been submitted.
<As an example>

Items	Format	Recipient(s) and Date
Beliverable 1, 2, 3	Report	SSO (DD/MM/YY)
Participant consent forms	Signed forms in envelop	SSO (DD/MM/YY)
Deliverable 4	Software codes or URL	Supervisor, Assessor (DD/MM/YY)
Deliverable 5	User manuals	Client, Supervisor (DD/MM/YY)

Type of project: ____________________________________

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) ____________________

Summary

<Concise statement of the problem you intended to solve and main achievements (no more than one A4 page)>

Acknowledgements

<The page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by other to the project.>
Note that it is not acceptable to solicit assistance on ‘proof reading’ which is defined as the “the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the test”;
see http://www.leeds.ac.uk/gat/documents/policy/Proof-reading-policy.pdf.

Contents

Chapter 1

Something "W" This Way Comes!

1.1 W Diameter Detector

The algorithm for computing the "near" to maximum W in a given height tree is based on an algorithm for finding the longest path in tree - it's diameter. It consist of running a standard Breadth First Search (BFS) from any vertex in the tree and recording the leaf that is furthest from that vertex. It can be show [ref] that the furthest leaf is a starting vertex of a diameter of the tree. To obtain the actual diameter then one can run another BFS rooted at that leaf.

Similarly this new algorithm runs a BFS from any vertex in the graph and records the leaf that is farthest in terms of w-length (or number of kinks on the path). This furthest leaf if guaranteed to be either the endpoint of W diameter of the tree or some path that has w-length at least -2 of the actual W diameter of the tree.

The algorithms works as follows:

Algorithm 1 Computing the W Diameter of a Height Tree.

```
function W_BFS(T, root)
    root.d = 0
    root.π = root
    furthest = root
    Q = ∅
    Enqueue(Q, root)
    while Q ≠ ∅ do
        u = Dequeue(Q)
        if u.d ≥ furthest.d then
            furthest = u
        for all v ∈ T.Adj[u] do
            if v.π == ∅ then
                v.π = u
                if u ∉ (v, u.π) then
                    v.d = u.d + 1
                else
                    v.d = u.d
                Enqueue(Q, v)
    Return furthest
function CALCULATE_W_DIAMETER(T)
    s = <any vertex>
    u = W_BFS(T, s)
    v = W_BFS(T, u)
    return b.d
```

Before proving the correctness of the algorithm we must first establish two useful properties that related the w length of to it's subpaths. Let $a \rightsquigarrow b$ be a path from a to b in the tree. The we will denote as $w(a, b)$ the number of kinks in that path (which is unique in the tree).

Definition 1 - Subpath Property.

Let $a \rightsquigarrow b$ be a subpath of $c \rightsquigarrow d$. Then $w(a, b) \leq w(c, d)$.

This property follows from the fact that all kinks of the path from a to b are also kinks of the path from c to d.

Definition 2 - Path Decomposition Property Property

Let $a \rightsquigarrow b$ be a path and t be a vertex on that path. Then $w(a, b) = w(a, t) + w(t, b) + x$, where $x \in \{0, 1\}$ depending on whether t is a kink in the path from a to b.

Indeed t can be a kink in the path from a to b, but it cannot be a kink in the paths from a to t and from t to b because it is an endpoint of both. All other kinks are preserved in either $w(a, t)$ or $w(t, b)$. To account for this we must consider additional cases for both possible values of x when using this property.

Theorem 1 - The Algorithm produces the endpoints of a path who is at most 2 kinks shy of being the kinkiest path in the tree.

Proof. After running the BFS twice we obtain two vertices u and v such that:

$$w(s, u) \geq w(s, t), \forall t \in V(T) \tag{1.1}$$

$$w(u, v) \geq w(u, t), \forall t \in V(T) \tag{1.2}$$

Furthermore let a and b be two leaves that are the endpoints of a path that is a W diameter. For any such pair we know that:

$$w(a, b) \geq w(c, d), \forall c, d \in V(T) \tag{1.3}$$

By this equation we have that $w(a, b) \geq w(u, v)$. Our goal in this proof will be to give a formal lower bound on $w(u, v)$ terms of $w(a, b)$. To this end let t be the first vertex in the path between a and b that the first BFS starting at s discovers. From this description it is clear that t cannot be a or b unless s is equal to a or b.

The proof can then be split into several cases.

Case 1. When the path from a to b does not share any vertices with the path from s to u.

Case 1.1. When the path from u to t goes through s.

In this case $s \rightsquigarrow u$ is a subpath of $t \rightsquigarrow u$, which in turn means that $w(t, u) \geq w(s, u)$. By equation 1.2 we also have that $w(s, u) \geq w(s, a)$. We can therefore conclude that $w(t, u) \geq w(a, t)$ as $s \rightsquigarrow a$ is a subpath of $t \rightsquigarrow a$.

Now via path decomposition of $a \rightsquigarrow b$ and $u \rightsquigarrow b$ at t have that:

$$w(a,b) = w(b,t) + w(t,a) + x$$

$$w(u,b) = w(b,t) + w(t,u) + y.$$

Where $x, y \in \{0,1\}$ depending on whether there is a kink at t for the path from a to b and from u to b respectively. As $w(t,u) \geq w(a,t)$ we can show that:

$$w(u,b) \geq w(b,t) + w(t,a) + y$$

$$w(u,b) \geq w(b,t) + w(t,a) + x - x + y$$

$$w(u,b) \geq w(a,b) - x + y$$

$$w(u,b) \geq w(a,b) + (y - x)$$

But as $w(u,v) \geq w(u,b)$ (by equation 1.2) we obtain that:

$$w(u,v) \geq w(a,b) + (y - x)$$

Considering all possible values that x and y can take, we can see that the minimum value for the right hand side of the equation is at $y = 0$ and $x = 1$. The final conclusion we may draw is that $w(u,v) \geq w(a,b) - 1$.

Case 1.2. When the path from u to t does not go through s.

If the path from u to t does not go through s then the paths $s \rightsquigarrow t$ and $s \rightsquigarrow u$ have a common subpath. Let s' be the last vertex in that subpath. We will be able to reduce this case to the previous one by using s' in the place of s. We must only account for a situation where s' is a kink for one of the paths and not the other. We know that $w(t,u) \geq w(s',u)$ (as a subpath) and through path decomposition we obtain that:

$$w(s,a) = w(s,s') + w(s',a) + x$$

$$w(s,u) = w(s,s') + w(s',u) + y$$

We know that $w(s,u) \geq w(s,a)$ and therefore:

$$w(s,s') + w(s',u) + y \geq w(s,s') + w(s',a) + x$$

$$w(s',u) + y \geq w(s',a) + x$$

$$w(s',u) \geq w(s',a) + (x - y)$$

Since $w(t,u) \geq w(s',u)$ we can further conclude that:

$$w(t,u) \geq w(s',a) + (x - y)$$

From the fact that $t \rightsquigarrow a$ is a subpath of $s' \rightsquigarrow a$ it follows that $w(s',a) \geq w(t,a)$. This lets us obtain that:

$$w(t,u) \geq w(t,a) + (x - y)$$

Now we are ready to proceed in a similar fashion as the previous case. We will decompose the paths from b to a and from b to u at the vertex t as follows:

$$w(b,a) = w(b,t) + w(t,a) + z$$

$$w(b,u) = w(b,t) + w(t,u) + w$$

$$w(b,u) \geq w(b,t) + w(t,a) + (x - y) + w$$

$$w(b,u) \geq w(b,t) + w(t,a) + z - z + (x - y) + w$$

$$w(b,u) \geq w(a,b) - z + (x - y) + w$$

$$w(b,u) \geq w(a,b) + (x - y) + (w - z)$$

The minimum value for the right hand side of this equation is at $x, w = 0$ and $y, z = 1$. Now as $w(u,v) \geq w(u,b)$ we finally obtain that:

$$w(u,v) \geq w(a,b) - 2$$

Case 2. When the path from a to b shares at least one vertex with the path from s to u. We can do a path decomposition as follows:

$$w(s,u) = w(s,t) + w(t,u) + x$$

$$w(s,a) = w(s,t) + w(t,a) + y$$

As $w(s,u) \geq w(s,a)$ (by equation 1.2)we obtain that:

$$w(t,u) \geq w(t,a) + (y - x)$$

This again leads us to:

$$w(b,a) = w(b,t) + w(t,a) + z$$

$$w(b,u) = w(b,t) + w(t,u) + w$$

$$w(b,u) \geq w(b,t) + w(t,a) + (x - y) + w$$

$$w(b,u) \geq (w(b,t) + w(t,a) + z) - z + (x - y) + w$$

$$w(b,u) \geq w(a,b) - z(x - y) + w$$

$$w(b,u) \geq w(a,b) + (x - y) + (w - z)$$

Where similarly to the previous case we obtain that:

$$w(u, v) \geq w(a, b) - 2$$

Based on these cases we can conclude that for any input tree the algorithm will produce a path that is at most -2 away from the actual maximum path.

$\square$

[1]

Chapter 2

Chapter 2 Title

2.1 Section 1

References

[1] D. Parikh, N. Ahmed, and S. Stearns. An adaptive lattice algorithm for recursive filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(1):110–111, 1980.

Appendices

Appendix A

External Material

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Appendix B

Ethical Issues Addressed