SUBTREE ISOMORPHISM AND BIPARTITE PERFECT MATCHING ARE MUTUALLY NC REDUCIBLE

Andrzej Lingas
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden

Marek Karpinski
Department of Computer Science
Bonn University, 5300 Bonn 1, West Germany

Abstract : A simple NC reduction of the problem of subtree isomorphism to that of bipartite perfect matching is presented. The reduction implies the membership of the subtree isomorphism problem in random NC^3. It is also shown that the problem of perfect bipartite matching is NC^1 reducible to that of subtree isomorphism. Finally, it is observed that the latter problem is in NC if the first tree is of valence $O(\log n)$.

1. *Introduction*

The subtree isomorphism problem is to decide whether a tree is isomorphic to a subgraph of another tree. Analogously, we define the version of the problem for directed trees. The subtree isomorphism problem is one of the two known restrictions of the general subgraph isomorphism problem to a non-trivial graph family that are solvable in polynomial (sequential) time [GJ79]. The other example is the subgraph isomorphism problem for biconnected outerplanar graphs [Li86].

The subtree isomorphism problem can be solved in the undirected and directed case by a recursive reduction to the maximum bipartite matching problem in time $O(n^{2.5})$ [Ma78,Re77]. Remembering that the latter problem is in random NC [KUW85, MVV87], it is natural to ask whether such a reduction can be performed by a fast parallel algorithm using a polynomial number of processors.

The recursive depth of the reduction in the sequential algorithms of Matula and Reynolds [Ma78, Re77] is unfortunately proportional to the height of the input trees. To obtain an NC reduction, we need cut the input trees recursively to decrease their height. A straight-forward way of cutting by using a vertex "$1/3 - 2/3$" separator [LT77] in the first tree and guessing its image in the second tree can lead to unpolynomial number of considered components of the second tree. By using a random method, one can decrease the number of components to a polynomial one. This yields random NC reductions of the problem of subtree isomorphism to that of maximum bipartite matching discovered recently and independently by Miller, Karp and Smolenski, and Karpinski [K86].

The main result of the paper is a simple, deterministic NC reduction of subtree isomorphism to bipartite perfect matching. The reduction uses a tree cutting technique relying on two following observations:

In any isomorphism ϕ between a rooted tree T and a subtree of another rooted tree U that maps the root of T on the root of U, the path P_1 from a given vertex separator v to the root of

T is mapped on the path P_2 from $\phi(v)$ to the root of U.
If we cut T and U by respectively removing the path P_1 and P_2 with the adjacent edges then the resulting subtrees are full subtrees of T and U respectively rooted at the sons of vertices on the paths that themselves do not lie on the paths.

By the second observation and a straight-forward inductive argumentation, if we apply our tree cutting method recursively then the number of resulting components is not greater than the total number of full subtrees of the rooted trees T and U (the latter number is equal to the number of vertices in T and U).
Since the problem of subtree isomorphism is easily reducible to that for rooted trees, we obtain an NC^3 reduction of subtree isomorphism to maximum bipartite matching, and hence, to bipartite perfect matching. Next, since the general perfect matching problem is in random NC^2 [MVV87], we can conclude that the subtree isomorphism problem is in random NC^3.
Interestingly, we can also show that a reverse reduction can be done efficiently in parallel. The reverse reduction consists in a straight-forward construction of two trees for the input bipartite graph such that the subtree isomorphism problem for the trees is solvable if and only if the graph has a perfect matching. The construction of the trees can be easily done by NC^1 circuits. By slightly modifying the proofs of the two presented NC reductions, we could obtain also the corresponding NC reductions between the problem of constructing a subtree isomorphism and that of constructing a bipartite perfect matching.
Note that by our NC reductions, the intriguing problem of the membership of bipartite perfect matching [KUW85] in NC is equivalent to that of the membership of subtree isomorphism in NC.
In [Ru81], Ruzzo observes that the restriction of the subtree isomorphism problem to trees of valence $O(\log n)$ can be solved by an auxiliary non-deterministic PDA operating within $O(\log n)$ working space and polynomially-bounded pushdown store. Since such an automaton can be simulated by NC circuits [Ru81], Ruzzo concludes that the restriction of subtree isomorphism is in NC.
Combining Ruzzo's observation with the fact that a depth first search of a tree can be performed by NC circuits [Sm83], we conclude that the restriction of subtree isomorphism where only the first tree is required to be of valence $O(\log n)$ is also in NC. We also observe that Ruzzo's method yields the membership of the problem of subtree isomorphism for ordered trees and the problem of deciding whether a term is a subterm of another term in NC^2.

The main contribution of the paper is the new tree cutting technique used in the NC reduction of subtree isomorphism to bipartite perfect matching. Recently, the first author has combined an analogous technique with the ideas from [Li86] to design an NC reduction of the subgraph isomorphism for biconected outerplanar graphs to the problem of finding a simple path between a pair of vertices. Since the latter problem is in NC [Co83], the problem of subgraph isomorphism for biconnected outerplanar graphs is also in NC [Li86a].

2. *Preliminaries*

We shall adhere to a standard graph and set notation (see [AHU74], [H69]). Specifically, given a set S, the term $\mid S \mid$ will stand for the cardinality of S. Given a tree T, we shall often denote its set of vertices also by T. If T is a rooted tree and v is a vertex of T, then the term T_v will denote the full (i.e. largest) subtree of T rooted at v. By a vertex separator a tree T we shall mean a vertex v of T whose removal disconnects T into subtrees none of which has more than two thirds of the vertices of T. Recall that any tree has at least one vertex separator [LT77]. For the definitions of uniform circuit families, the classes NC^k, NC, their random versions RNC^k, RNC and the corresponding notions of reducibility, the reader is referred to [P79,Ru81,Co83].

The NC reductions between the problem of subtree isomorphism and that of bipartite perfect matching will be shown by proving corresponding reductions for simple modifications of these two problems. In this section, we shall define the modifications and prove them to be equivalent to the original problems in the sense of NC^1 reducibility.
The modification of the subtree isomorphism problem will be called the *root subtree isomorphism* problem.

Definition 2.1: The root subtree isomorphism problem is to decide whether there exists an isomorphism between a tree and a subgraph of another tree mapping the root of the first tree on the root of the second tree. Such an isomorphism will be called a *root imbedding* of the first tree in the other one.

Lemma 2.1: The subtree isomorphism problem for undirected trees as well as that for directed trees are NC^1 reducible to the root subtree isomorphism problem.
Proof Sketch: For directed trees, the reduction is obvious; the first tree is isomorphic to a subgraph of the other if and only if there is a root imbedding of the first tree in a full subtree of the other tree. In the undirected case, following [Ma78], for every edge (v, w) of the second tree U, we define the limb $U(v, w)$ as the maximum part of U reachable from v by simple paths passing through w, rooted at v. Analogously, we define the limb $U(w, v)$. Next, we identify the first tree T with its limb $T(x, y)$ where x is an arbitrary leaf of T. Now, it is easily seen that T is isomorphic to a subgraph of U if and only if there is a root imbedding of the limb $T(x, y)$ in a limb of U. It should be clear that in both cases, the simple many-one reductions can be done by NC^1 circuits. ∎

Lemma 2.2: The root subtree isomorphism problem is NC^1 reducible to the subtree isomorphism problem for undirected trees as well as to that for directed trees.
Proof Sketch: Let n be the number of vertices in the larger of the two input trees. It is enough to add to each root of the input trees n dummy sons. ∎

The modification of the bipartite perfect matching problem will be called the *bipartite partly perfect matching* problem.

Definition 2.2: Given a bipartite graph $G(V_1,\ V_2,\ E)$, a *partly perfect matching* of G is a matching of G whose cardinality is equal to $\min\{\mid V_1 \mid, \mid V_2 \mid\}$. The bipartite partly perfect matching problem is to decide whether a bipartite graph has a partly perfect matching.

Lemma 2.3: The problems of bipartite perfect matching and that of bipartite partly perfect matching are mutually NC^1 reducible.

Proof Sketch: Clearly, the former problem is NC^1 reducible to the latter. To obtain the reverse reduction, given a bipartite graph $G(V_1,\ V_2,\ E)$, where $|V_1| \leq |V_2|$, we extend V_1 by $|V_2|$-$|V_1|$ dummy vertices adjacent to all vertices in V_2. ∎

3. *Subtree isomorphism is NC^3 reducible to bipartite perfect matching*

In this section, we shall show that the problem of root subtree isomorphism is NC^3 reducible to that of bipartite partly perfect matching. By Lemma 2.1 and 2.3, this yields also an NC^3 reduction of subtree isomorphism to bipartite perfect matching.

The main, recursive, reduction procedure RSI, using some preprocessing, is defined as follows.

input : two rooted trees T and U, and for every vertex t of T, the number $D(t)$ of the descendants of t in T;

output : *if* there is a root imbedding of T in U *then* 1 *else* 0;

data structures : a matrix $M(t,u)$, $t \in T$, $u \in U$; setting the entry $M(t,u)$ to 1 (respectively, 0) will denote that there is (respectively, there is no) a root imbedding of T_t in U_u.

```
procedure RSI(T, U, D)
distribute the value of | T | to all vertices of T;
select a vertex separator v of T;
find the path P1 from v to the root of T and its length | P1 |;
for i = 0,..,| P1 | do in parallel v_i ← the i-th vertex of P1;
for all vertices u of U do in parallel
begin
   find the path P2 from u to the root of U and its length | P2 |;
   if | P1 |=| P2 | then
      begin
         for i = 0,..,| P1 | do in parallel
         begin
            u_i ← the i-th vertex of P2;
            if v_i has more sons than u_i then
               begin
                  YES(i) ← 0;
                  go to A
               end;
            for all pairs (s1, s2) where s1 is a son of v_i not lying on P1 and s2 is a son of u_i
            not lying on P2 do in parallel M(s1, s2) ← RSI(T_s1, U_s2, D/T_s1);
            G_i(M) ← the bipartite graph induced by the matrix M restricted to the sons of
            v_i and u_i; (i.e. sons s1, s2 are adjacent in G_i(M) if and only if M(s1, s2) = 1);
            YES(i) ← if G_i(M) has a partly perfect matching then 1 else 0;
      A: end;
```

$YES(u) \leftarrow$ *if* $\bigwedge_{i=0}^{|P_1|} YES(i)$ *then* 1 *else* 0

end;

else $YES(u) \leftarrow 0$;

end

if $\bigvee_{u \in U} YES(u)$ *then* return 1 *else* return 0

The correctness of the procedure $RSI(T, U, D)$ immediately follows from the fact that in any root imbedding ϕ of T in U, the path P_1 in T from the vertex separator v to the root of T is mapped on the path P_2 from $\phi(v)$ to the root of U, and that for $i = 0, ..., | P_1 |$, each full subtree of T rooted at a son of the i-th vertex of P_1 not lying on P_1 is root imbedded in a unique full subtree of U rooted at a son of the i-th vertex of P_2 not lying on P_2.

We may assume $| T | \leq | U |$ without loss of generality. Let n denote $| U |$. To show that $RSI(T, U, D)$ can be implemented by NC circuits with oracle gates for bipartite partly perfect matching tests, we argue as follows.

a) The distribution of the value $| T |$ to the vertices of T can be done by a circuit of $O(\log n)$ depth and $O(n \log n)$ size. Then, we can decide, for each vertex t of T, whether t is a vertex separator of T by computing $| T | - D(t)$ and the maximum of $D(s)$ over the sons s of the vertex t in T. Clearly, it can be implemented by NC^1 circuits (see [Co83]). Finally, we can select a vertex v from these vertex separators using a circuit of depth $O(\log n)$ and size $O(n \log n)$.

b) The path P_1, and similarly, the path P_2, can be found by using a standard $O(\log n)$ method on a concurrent read exclusive write parallel RAM with $O(n^2)$ processors. In the j-th iteration of the method, we find, for each vertex v in the tree, the path from v to its ancestor in the distance 2^j by concatenating the path from v to its ancestor in the distance 2^{j-1} with the copied path between the two ancestors of v. By [SV84], the method can be implemented by (uniform) circuits of unbounded fan-in, $O(\log n)$ depth and polynomial size. Hence, it can be implemented by NC^2 circuits.

c) The total number of son pairs (s_1, s_2) over all $i = 0, 1, ..., | P_1 |$, is not greater than the product of $| T |$ and $| U |$ which is at most n^2.

d) By (a), (b) and (c), the body of $RSI(T, U, D)$ can be implemented by NC^2 circuits if we do not count the recursive calls and bipartite partly perfect matching tests.

e) Note that the subtrees T_{s_1} and U_{s_2} are full subtrees of T and U respectively. Hence, by induction, all the subtrees occurring in the recursive calls of $RSI(T, U, D)$ are also full subtrees of T and U respectively, and the original values of the matrix D can be used there. It follows that the procedure $RSI(T, T, D)$ invokes at most $| U || T |$ different recursive calls. Hence, the whole procedure can be implemented by filling the entries of the matrix M in a bottom up manner. By (d), it can be done by NC circuits with oracle gates for the matching tests.

f) The recursive depth of $RSI(T, U, D)$ is $O(\log n)$. Hence, the depth of the circuits specified in (e) is $O(\log^3 n)$ by (d). However, the depth of the oracle gates can be shown to be only $O(\log^2 n)$ by the definition of $RSI(T, U, D)$.

To estimate the parallel complexity of the whole reduction, it remains to estimate the cost of preprocessing. We can find, for each vertex t of T, the total number $D(t)$ of descendants of t by using the so called Euler tour technique [TV85]. By applying this technique, the preprocessing can be done in time $O(\log n)$ time using $O(n)$ processors and $O(n)$ space on an exclusive-read exclusive write parallel RAM [TV85]. Hence, by [SV84], it can be done by (uniform) circuits of unbounded fan-in, $O(\log n)$ depth and polynomial size, and consequently, by NC^2 circuits. Thus, we obtain the following theorem.

Theorem 3.1: The problem of root subtree isomorphism is NC^3 reducible to that of bipartite partly perfect matching.

Combining Theorem 3.1 with Lemma 2.1 and 2.3, we obtain the main result of the paper.

Theorem 3.2: The problem of subtree isomorphism and that of directed subtree isomorphism are NC^3 reducible to the problem of bipartite perfect matching.

Combining Theorem 3.2 with the fact that the problem of bipartite perfect matching is in RN^2 [MVV87], we obtain also the following important theorem.

Theorem 3.3: The problem of subtree isomorphism and that of directed subtree isomorphism are in RNC^3.

4. *Bipartite perfect matching is NC^1 reducible to subtree isomorphism*

Let $G = (V_1, V_2, E)$ be a bipartite graph where $|V_1| \le |V_2|$. We shall construct rooted trees T_1 and T_2 using NC^1 circuits such that there is a root imbedding of T_1 in T_2 if and only if G has a partly perfect matching. By Lemma 2.2, 2.3, this will yield an NC^1 reduction of the problem of bipartite perfect matching to that of subtree isomorphism. The trees T_1 and T_2 are constructed as follows.

First, for each $v_j \in V_1$, $j = 1, ..., n$, we construct a tree S_j which consists of a directed line of length $\lfloor \log n \rfloor$ with additional single leaves attached to some vertices of the line but for its last vertex. precisely, we attach such a single leaf to the $i-th$ vertex in the line, $i = 1, ..., \lfloor \log n \rfloor$, if the $i-th$ digit in the binary $\lfloor \log n \rfloor$-bit representation of j is 1. We root S_j at the first vertex of the line. Note that the tree S_j is always of height $\lfloor \log n \rfloor$, and for $1 \le j' \le n$, $j \ne j'$, there is no root imbedding of S_j in $S_{j'}$.

Secondly, for each $v_j \in V_1$, $j = 1, ..., n$, we construct a tree $U(v_j)$ such that the root of $U(v_j)$ and the son of the root are of outdegree one, and the grandson of the root is the root of the tree S_j. By the properties of $S(j)$, the tree $U(v_j)$ is always of height $\lfloor logn \rfloor + 2$, and for $1 \le j' \le n$, $j \ne j'$, there is no root imbedding of $U(v_j)$ in $U(v_j')$.

Thirdly, for each $w_j \in V_2$, $j = 1, ..., n$, we construct a tree $T(w_j)$ whose root as well as its son are of outdegree one and the grandsons w_k of the root are in one-to-one correspondence with the vertices v_i in V_1 adjacent to w_j such that w_k is the root of the tree $S(i)$. By the properties of the trees $S(i)$, the tree $T(w_j)$ is always of height $\lfloor logn \rfloor + 2$. Next, by the construction of the trees $U(v_i)$, $i = 1, ..., n$, there is a root imbedding of $U(v_i)$ in the tree $T(w_j)$ if and only if v_i is adjacent to w_j in G. Note that since the roots and the sons of the roots in the above trees are of outdegree one, there are no two disjoint root imbeddings of two different trees $U(v_i)$ in $T(w_j)$.

Now, the construction of the trees T_1 and T_2 is obvious. The root of T_1 is of outdegree n and its sons are the roots of the trees $U(v_i)$, $i = 1, ..., n$, respectively. Similarly, the root of T_2 is of outdegree n and its sons are the roots of the trees $T(v_j)$, $i = 1, ..., n$, respectively. Note that the two trees are of height $\lfloor \log n \rfloor + 3$ each. Consequently, by the construction of T_1 and T_2, each of the subtrees $U(v_i)$ of T_1 is root imbedded in one of the subtrees $T(w_j)$ of T_2, in any root imbedding of T_1 in T_2. As we know, each of the above subtrees $T(w_j)$ is distinct. Moreover, by the construction of $T(w_j)$, there is a root imbedding of $U(v_i)$ in $T(w_j)$ if and only if the vertices v_i and w_j are adjacent in G. Thus, if there is a root imbedding of T_1 in T_2 then the graph G has a partly perfect matching. Conversely, if G has a partly perfect matching then we can root imbed the subtrees $U(v_i)$ of T_1 in the subtrees $T(w_j)$ of T_2 in one-to-one manner and additionally map the root of T_1 on the root of T_2 to a obtain a full root imbedding of T_1 in T_2. The construction of the basic parts of the trees T_1 and T_2 which are the subtrees $S(j), j = 1, ..., n$, can be easily performed in $O(logn)$ working space of a Turing machine since the subtrees are in the one-to-one, trivial correspondence with the binary strings of length $\lfloor \log n \rfloor$. Then, the subtrees $U(v_i)$, $T(v_j)$ and finally the trees T_1 and T_2 can be easily assembled using $O(logn)$ working space by running the procedure for constructing the trees $S(j)$ several times. Instead of generating the subtrees $S(j)$, $U(v_i)$, and finally the trees T_1, T_2 within $O(logn)$ working space, one can easily generate circuits of $O(logn)$ depth that generate respectively the above subtrees, using the circuits for smaller subtrees to assemble the circuits for the larger subtrees. It is not difficult to see that the circuits can be generated using $O(logn)$ space. Thus, we have the following theorem.

Theorem 4.1: The problem of bipartite partly perfect matching is NC^1 reducible to the problem of root subtree isomorphism.

Combining Theorem 4.1 with Lemma 2.2 and 2.3, we obtain the main result of this section.

Theorem 4.1: The problem of bipartite perfect matching is NC^1 reducible to the problems of subtree isomorphism and directed subtree isomorphism.

5. *Subtree isomorphism is in NC if the first tree is of valence $O(logn)$*

In [Ru81], Ruzzo has observed that the subtree isomorphism problem constrained to trees of valence $O(\log n)$ can be solved by a non-deterministic, $\log n$ space, auxiliary PDA with polynomially-bounded pushdown store which implies the membership of so constrained subtree isomorphism in NC^2. The PDA performs a depth-first search of the second tree, i.e. this into which we want to imbed the other, while in parallel traversing the first tree, non-deterministically choosing an ordering of the descendants of each node.

We can slightly generalize the above observation of Ruzzo to allow the second tree to be of any valence by using the fact that the problem of depth first search of trees is in NC [SM83] (i.e. the vertices of a tree can be listed in a depth first order by NC circuits).

To use the above fact, we equip a a non-deterministic, $\log n$ space, auxiliary PDA with an oracle tape. When the PDA non-deterministically writes a binary word on the oracle tape, the oracle answers yes if the word is an encoding of the consecutive vertex of the second tree in a given depth search order. The pushdown store is used only to perform a depth first search of the first tree in one-to-one, non-deterministically guessed correspondence to the given depth first search of the second tree on a part of the second tree. As in the case of Ruzzo, but only for the first tree, $O(\log n)$ bit vectors are used to keep track which sons of a vertex to which we shall backtrack have been already visited. Since the problem of depth first search for trees is in NC, we may assume that the oracle set is NC. On the other hand, we have the following lemma.

Lemma (Fact ?) 5.1: Given a non-deterministic, $\log n$ space auxiliary PDA with polynomially-bounded pushdown store and an oracle in NC, the language recognized by the PDA is in NC.
Proof Sketch: Consider a non-deterministic, $\log n$ space auxiliary PDA that has a two part input, the first part corresponds to the input of the original PDA, the second part consists of a polynomial in the length of the first part sequence of answers YES or NO. The new PDA acts as the original PDA on the first part of input treating the second part as answers to consecutive oracle queries. Moreover, it prints the queries on the output tape instead of an oracle tape. It is easy to show that the function computable by the new PDA is NC-computable. Now, given NC circuits computing the above function, we can easily connect them with the NC-circuits for the oracle set to get NC-circuits recognizing the language accepted by the original PDA. ∎

By the above lemma and the properties of the oracle of our PDA, we obtain the following theorem.

Theorem 5.1: The subtree isomorphism problem where the first tree is of valence $O(logn)$ is in NC.

Marginally, let us observe that Ruzzo's method can be directly applied to test ordered trees [H69] for subgraph isomorphism (comparing with the definition of non-ordered subtree isomorphism, the sub-isomorphism here is additionally required to be monotone with respect to the tree orderings). To perform the corresponding depth-first search of both trees, it is sufficient to keep the paths from the tree roots to currently visited vertices, and for each of the vertices on the paths, the number of its sons already visited, using the polynomially-bounded pushdown store of the auxiliary non-deterministic PDA operating within $O(\log n)$ space. Hence, by [Ru81], we obtain the following remark.

Remark 5.1: The subgraph isomorphism problem for ordered trees is in NC^2.

Similarly, using Ruzzo's method, we can obtain the following remark.

Remark 5.2: The problem of deciding whether a term is a subterm of another term is in NC^2 (the terms are over a given, finite alphabet).

References

[AHU74] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Massachusetts, 1974).

[Co83] S.A. Cook, *The Classification of Problems which have Fast Parallel Algorithm*, (Proc. Foundations of Computation Theory, LNC 158, Borgholm, Sweden, 1983).

[GJ79] M.R. Garey, D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-completeness* (Freeman, San Francisco, 1979).

[H69] F. Harary, *Graph Theory* (Addison-Wesley, Reading, Massachusetts, 1969).

[K86] R.M. Karp, *personal communication.*

[KUW83] R.M. Karp, E. Upfal, and A. Wigderson, *Constructing a Maximum Matching is in random NC* (Proc. 17th STOC, 1985, pp. 22-31).

[Li86] A. Lingas, *Subgraph Isomorphism for Biconnected Outerplanar Graphs in Cubic Time* (Proc. of the 3rd Symposium on Theoretical Aspects of Computer Science, 1986, France, LNC 210).

[Li86a] A. Lingas, Subgraph Isomorphism for Biconnected Outerplanar Graphs is in NC (in preparation).

[LT77] R.J. Lipton and R.E. Tarjan,*A separator theorem for planar graphs* (Proc. of Waterloo Conference on Theoretical Computer Science, 1975).

[Ma78] D. W. Matula, *Subtree isomorphism in* $O(n^{5/2})$ (Annals of Discrete Mathematics 2 (1978) 91-106).

[MVV86] K. Mulmuley, U.V. Vazirani and V.V. Vazirani, *A Parallel Algorithm for Matching* (manuscript, 1986).

[P79] N. Pippenger, *On simultaneous resource bounds* (Proc. 20th IEEE FOCS, 307-311).

[Re77] S. W. Reyner, *An analysis of a good algorithm for the subtree problem* , (SIAM J. Comput. 6 (1977), 730-732).

[Ru81] W.L. Ruzzo, *On uniform circuit complexity* (J. CSS 22, pp. 365-383).

[Sm83] J. R. Smith, *Parallel Algorithms for Depth First Searches: 1. Planar Graphs* (International Conference on Parallel Processing, 1984. To appear in SIAM Journal of Computing).

[SV84] L. Stockmeyer and U. Vishkin, *Simulation of Parallel Random Access Machines by Circuits* (SIAM J. Comput. 13 (1984), 409-422).

[TV85] R.E. Tarjan and U. Vishkin, *Finding Bi-connected Components and Computing Tree Functions in Logarithmic Parallel Time* (Proceedings 25th IEEE FOCS 1984 pp. 12-20).