ITERATIVE ALGORITHMS FOR GLOBAL FLOW ANALYSIS

by

Robert Endre Tarjan

STAN-CS-76-547
MARCH 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Iterative Algorithms for Global Flow Analysis

Robert Endre Tarjan */

Computer Science Department
Stanford University
Stanford, California 94305

February 1976

Abstract. This paper studies iterative methods for the global flow analysis of computer programs. We define a hierarchy of global flow problem classes, each solvable by an appropriate generalization of the "node listing" method of Kennedy. We show that each of these generalized methods is optimum, among all iterative algorithms, for solving problems within its class. We give lower bounds on the time required by iterative algorithms for each of the problem classes.

Keywords: computational complexity, flow graph reducibility, global flow analysis, graph theory, iterative algorithm, lower time bound, node listing.

*/ Research partially supported by National Science Foundation grant MCS 75-22870.

1. Introduction.

A problem extensively studied in recent years [2,3,5,7,8,9,12,13,14, 15,27,28,29,30] is that of globally analyzing computer programs; that is, collecting information which is distributed throughout a computer program, generally for the purpose of optimizing the program. Roughly speaking, global flow analysis requires the determination, for each program block*/, of a property known to hold on entry to the block, independent of the path taken to reach the block.

A widely used approach to global flow analysis is to model the set of possible properties by a semi-lattice (we desire the "maximum" property for each block), to model the control structure of the program by a directed graph with one vertex for each program block, and to specify, for each branch from block to block, the function by which that branch transforms the set of properties. If the semi-lattice of properties and the set of transforming functions satisfy certain axioms, efficient methods are available for producing a maximum property for each block [1,7,8,12,15, 27,30]. In essence, finding such properties involves solving a set of linearequations.

The known algorithms are of two types: "iterative" algorithms, which use only the semi-lattice meet operation and function application [9,12, 14,15], and "elimination" algorithms, which use in addition function composition and meet extended to functions [3,5,7,8,27,28,30]. Most of the elimination algorithms are refinements of the "interval analysis" method of Cocke and Allen [3,5], which requires that the program flow graph have a special property, called "reducibility". However, it is

*/ A block is a set of statements with a single entry and a single exit point.

possible to formulate a general elimination method, related to Gaussian elimination, which applies to all graphs and is particularly efficient on reducible or almost-reducible graphs [27]. This algorithm requires only $O(n\ \alpha(n,n))$ time on an n vertex reducible program flow graph ($\alpha(n,n)$ is a very slowly growing function related to a functional inverse of Ackermann's function).

The elimination algorithms apply to very general global flow problems and are asymptotically efficient, but they are rather complicated to program: There are several iterative algorithms which are much simpler but which work only on restricted kinds of global flow problems. These include the propagation algorithm of Kildall [15], studied by Hecht and Ullman [9] and related to early work by Vyssotsky [29], and the "node listing" algorithm of Kennedy [14]. Kam and Ullman [13] have derived a necessary and sufficient condition for global flow problems to be efficiently solvable by Hecht and Ullman's "depth-first" version of Kildall's algorithm. Aho and Ullman [1], by giving an algorithm for constructing short node listings, have shown that Kennedy's algorithm can be implemented to run in $O(n \log n)$ time on an n vertex reducible program flow graph.

This paper extends the results of Kennedy [14] and Kam and Ullman [13]. We present a hierarchy of global flow problem classes, each solvable by an appropriate generalization of Kennedy's algorithm. One of the classes is the one considered by Kam and Ullman. We show that each of the generalized algorithms is optimum, among all iterative algorithms, for solving problems in its class. We give lower bounds on the time required by iterative algorithms for each of the problem classes.

The paper contains five sections. Section 2 contains the necessary graph theory, including lemmas'needed to derive the lower bounds. Section 3 gives an abstract framework for global flow analysis, defines the hierarchy of problem classes, and presents the corresponding hierarchy of algorithms. Section 4 shows the optimality of the algorithms and provides a lower bound on the time required for each problem class. Section 5 contains further remarks.

2. Directed Graphs and Iteration Sequences.

A directed graph $G = (V,E)$ is a finite set of $n = |V|$ elements called vertices and a finite set E of $m = |E|$ elements called edges. Each edge (v,w) is an ordered pair of distinct vertices. The edge (v,w) leaves v and enters w; we say v is a predecessor of w. The in-degree of a vertex v is the number of edges entering v; the out-degree of v is the number of edges leaving v. The reverse of a graph is formed by reversing the direction of all its edges.

A path p of length k from v to w is a sequence of edges $p = (v_1,v_2),(v_2,v_3),\ldots,(v_k,v_{k+1})$ with $v_1 = v$ and $v_{k+1} = w$. The path p contains vertices $v_1,\ldots,v_{k+1}$ and edges $(v_1,v_2),\ldots,(v_k,v_{k+1})$ and avoids all other vertices and edges. There is a path of no edges from every vertex to itself. A vertex w is reachable from a vertex v if there is a path in G from v to w.

A triple $G = (V,E,r)$ is a flow graph if (V,E) is a directed graph, $r \in V$, and every vertex is reachable from r. G is a program flow graph if the out-degree of every vertex is at most two. Every program flow graph has $m \leq 2n$. A flow graph $G = (V,E,r)$ is reducible if it can be reduced to the flow graph $(\{r\},\emptyset,r)$ by applying a sequence of transformations of the following form.

T: Let $w \neq r$ be a vertex with exactly one entering edge (v,w). Replace (V,E,s) by (V',E',s), where $V' = V-\{w\}$;
$E' = \{(x,y) \in E \mid w \notin \{x,y\}\} \cup \{(v,y) \mid (w,y) \in E \text{ and } y \notin \{v,w\}\}$.

Cocke and Allen introduced reducible graphs [3,5]; the definition above is Hecht and Ullman's [10,11], modified to avoid the creation of loops (edges of the form (v,v)). There is an $O(m\ \alpha(m,n))$ time algorithm to test reducibility and to construct a reducing sequence of transformations for any reducible flow graph [24,25].

Let $k \geq 1$. A k-path in a flow graph $G = (V,E,r)$ is a path $p = (v_1,v_2)(v_2,v_3) \ldots (v_l,v_{l+1})$ such that no vertex appears more than k times among $v_2,\ldots,v_{l+1}$. A ks-path is a k-path which begins at r and contains r no more than k times. A k-sequence for $G = (V,E,r)$ is a sequence of edges which contains every k-path of G as a subsequence. A ks-sequence is a sequence of edges which contains every ks-path of G as a subsequence. Our lower bound proofs require the following results concerning lengths of k-sequences and ks-sequences. The first two lemmas are immediate corollaries of results in [20]. The third lemma is new.

Lemma 1 [20]. For infinitely many n, there is a program flow graph $G = (V,E,r)$ with $|V| = n$, such that the reverse of G is reducible and any ks-sequence for G contains at least $c_1\ n \log n$ edges*/.

Lemma 2 [20]. For infinitely many n, there is a (non-reducible) program flow graph $G = (V,E,r)$ with $|V| = n$ such that any sequence containing each ls-path ending at a predecessor of r contains at least $c_2 n^2$ edges.

---

*/ Throughout this paper, $c, c_1, c_2, \ldots$ denote suitable positive constants.

Corollary 1. For infinitely many n, there is a (non-reducible) program flow graph $G = (V,E,r)$ with $|V| = n$ such that any ks-sequence for G contains at least $c_2 k n^2$ edges.

Proof. Any ks-sequence for one of the graphs given by Lemma 2 must contain k disjoint subsequences, each containing all Is-paths ending at a predecessor of r. □

For any $k \geq 1$, let $s(k)$ be the length of the shortest sequence containing each permutation of the numbers $1,2,\ldots,k$ as a subsequence. Newey [21] gives the following values of $s(k)$: $s(1) = 1$, $s(2) = 3$, $s(3) = 7$, $s(4) = 12$, $s(5) = 19$, $s(6) = 28$, $s(7) = 39$. Newey [21] and Koutas and Hu [18] have shown that for all k, $s(k) \leq k^2 - 2k + 4$. Kwiatowski and Kleitman [19] have shown that, for all positive ϵ, $s(k) \geq k^2 - c(\epsilon)k^{7/4+\epsilon}$ for all k.

Lemma 3. Let $k \geq 1$. For infinitely many n there is a reducible program flow graph $G = (V,E,r)$ with $|V| = n$ such that any ks-sequence contains at least $c(k)n^{\log_k s(k)}$ edges.

Proof. For any fixed k, we recursively construct a sequence of flow graphs $G(k,i)$. Each $G(k,i)$ will have a unique start vertex $r(k,i)$ and a unique finish vertex $f(k,i)$. Let

$$G(k,0) = (\{r(k,0),f(k,0)\}, \{(r(k,0),f(k,0)), (f(k,0),r(k,0))\}, r(k,0)).$$

We construct $G(k,i+1)$ from k copies of $G(k,i)$ and three new vertices $r(k,i+1)$, $f(k,i+1)$, and $x(k,i+1)$, as follows. Let $G_1(k,i), G_2(k,i), \ldots, G_k(k,i)$ be k copies of $G(k,i+1)$. Let $G(k,i+1) = (V(k,i+1), E(k,i+1), r(k,i+1))$, where

$$V(k,i+1) = \bigcup_{j=1}^{k} V_j(k,i) \cup \{r(k,i+1), x(k,i+1), f(k,i+1)\};$$

$$E(k,i+1) = \bigcup_{j=1}^{k} (E_j(k,i) \cup \{(x(k,i+1),r_j(k,i)), (f_j(k,i),x(k,i+1)), (f_j(k,i),f(k,i+1))\})$$

$$\cup \{(r(k,i+1),x(k,i+1)), (f(k,i+1),r(k,i+1))\}.$$

Figure 1 illustrates $G(k,i+1)$.

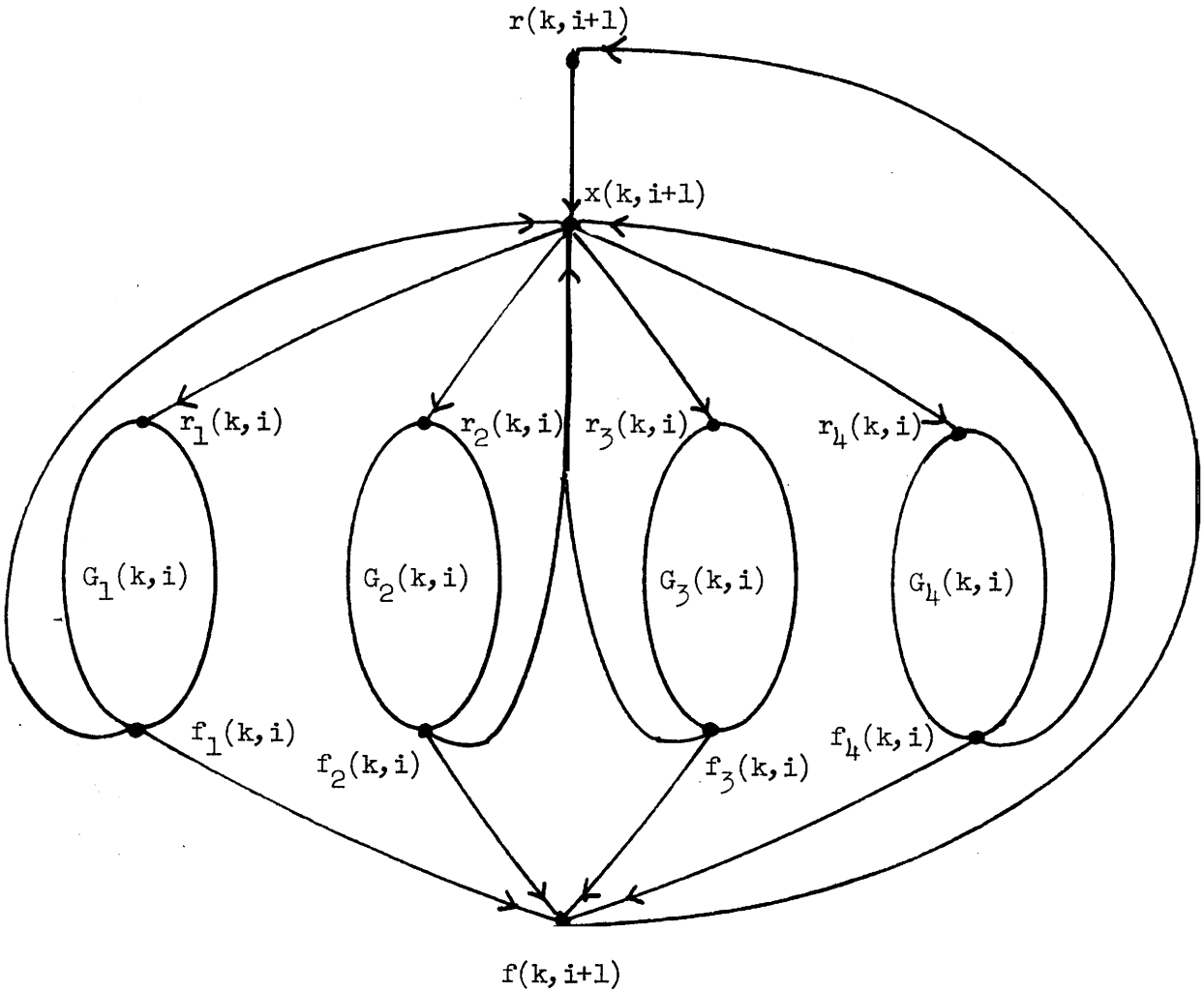

Figure 1. $G(k,i+1)$ for $k = 4$.

If $p(1),p(2),\ldots,p(k)$ are ks-paths in $G(k,i)$ ending at $f(k,i)$, then

$$(r(k,i+1),x(k,i+1))(x(k,i+1),r_{\sigma(1)}(k,i))p_{\sigma(1)}(1)(f_{\sigma(1)}(k,i),x(k,i+1))$$

$$(x(k,i+1),r_{\sigma(2)}(k,i))p_{\sigma(2)}(2)\cdots p_{\sigma(k)}(k)(f_{\sigma(k)}(k,i),f(k,i+1))$$

is a ks-path in $G(k,i+1)$ ending at $f(k,i+1)$, where σ is any permutation on $1,2,\ldots,k$ and $p_{\sigma(j)}(j)$ is the path in $G_{\sigma(j)}(k,i)$ corresponding to $p(j)$.

Let $S = e_1,e_2,\ldots,e_\ell$ be any sequence for $G(k,i+1)$ containing all ks-paths ending at $f(k,i+1)$. Form a sequence $S' = z_1,z_2,\ldots,z_\ell$ of occurrences of $0,1,2,\ldots,k$ from S as follows. Suppose $z_1,z_2,\ldots,z_b$ have been defined. Let j be such that e_{b+1} is an edge of $G_j(k,i+1)$. (If there is no such j, let $z_{b+1} = 0$.) Let b' be the maximum $b' \leq b$ such that $z_{b'} = j$. (If there is no such b', let $b' = 0$.) If the sequence of edges $e_{b'+1},\ldots,e_{b+1}$ contains every ks-path of $G_j(k,i)$ as a subsequence, let $z_{b+1} = j$. Otherwise let $z_{b+1} = 0$.

We claim S', so defined, contains every permutation of $1,2,\ldots,k$ as a subsequence. For, let σ be any permutation of $1,2,\ldots,k$. Let $z_{b(1)}$ be the first occurrence of $\sigma(1)$ in S', and let $p_{\sigma(1)}(1)$ be a ks-path in $G_{\sigma(1)}(k,i)$ ending at $f_{\sigma(1)}(k,i)$ and contained in $e_1,\ldots,e_{b(1)}$ but not in $e_1,\ldots,e_{b(1)-1}$. In general, let $z_{b(j+1)}$ be the first occurrence of $\sigma(j+1)$ following $z_{b(j)}$ in S', and let $p_{\sigma(j+1)}(j+1)$ be a ks-path in $G_{\sigma(j+1)}(k,i)$ ending at $f_{\sigma(j+1)}(k,i)$ and contained in $e_{b(j)+1},\ldots,e_{b(j+1)}$ but not in $e_{b(j)+1},\ldots,e_{b(j+1)-1}$. It must be possible to define $b(1),b(2),\ldots,b(k)$ since otherwise there is a ks-path

$$(s(k,i+1),x(k,i+1))(x(k,i+1),s_{\sigma(1)}(k,i))p_{\sigma(1)}(1)(f_{\sigma(1)}(k,i),x(k,i+1))$$

$$(x(k,i+1),s_{\sigma(2)}(k,i))p_{\sigma(2)}(2)\cdots p_{\sigma(k)}(k)(f_{\sigma(k)}(k,i),f(k,i+1))$$

in $G(k,i+1)$, ending at $f(k,i+1)$, which is not contained in S . Thus σ is contained in S' .

Let $\ell(k,i)$ be the length of the shortest sequence containing all ks-paths of $G(k,i)$ ending at $f(k,i)$. Clearly $\ell(k,i) > 1$. The above argument implies that $\ell(k,i+1) \geq \ell(k,i)\cdot s(k)$. Thus

(1) $$\ell(k,i) \geq (s(k))^i \quad .$$

Let $|V(k,i)| = n(k,i)$. Then $n(k,0) = 2$ and $n(k,i+1) = kn(k,i)+3$. Thus

(2) $$n(k,i) < 3\sum_{j=0}^{i} k^j \leq 3k^{i+1} \quad .$$

It' follows that

(3) $$\ell(k,i) \geq c(k)\ n(k,i)^{\log_k s(k)} \quad \text{for some constant } c(k) \ .$$

Each $G(k,i)$ is reducible. (To reduce $G(k,i+1)$, reduce each $G_j(k,i)$ to $(\{r_j(k,i)\},\emptyset,r_j(k,i))$, then delete the remaining vertices in the order $r_1(k,i),r_2(k,i),\ldots r_k(k,i),f(k,i+1),x(k,i+1)$.) Furthermore the out-degree of each vertex of $G(k,i)$ is at most $\max\{k,3\}$. From $G(k,i)$ we can form a reducible program flow graph $G'(k,i)$ by replacing each vertex of out-degree greater than two by a binary fan-out tree.

$G'(k,i)$ contains $O(n(k,i))$ vertices. Furthermore every ks-path of $G(k,i)$ ending at $f(k,i)$ is contained in a ks-path of $G'(k,i)$ ending at $f(k,i)$. The lemma follows from (3). □

A slight modification of this proof gives:

Corollary 2. Let $k \geq 1$. For infinitely many n there is a program flow graph $G = (V,E,r)$ with $|V| = n$ such that the reverse of G is reducible and any ks-sequence for G contains at least $c(k)n^{\log_k s(k)}$ edges.

Aho and Ullman's construction [1] of an $O(n \log n)$ length 1-sequence for any reducible program flow graph shows that the bound in Lemma 1 is tight to within a constant factor. The bound in Corollary 1 is obviously tight to within a constant factor, as is the Lemma 3 bound for 1-sequences. For $k \geq 2$, it is an open problem whether the Lemma 3 and Corollary 2 bounds are tight.

3. Global Flow Problems and Iterative Algorithms.

Let L be a set with a binary meet operation $\wedge$ satisfying the following axioms.

A0: L is closed under $\wedge$.

A1; $x \wedge (y \wedge z) = (x \wedge y) \wedge z$.

A2: $x \wedge y = y \wedge x$.

A3: $x \wedge x = x$.

A4: There is an element $0 \in L$ such that $0 \wedge x = 0$.

A5: There is an element $1 \in L$ such that $1 \wedge x = x$.

As a consequence of A0 - A3 we can define a partial order on L by $x \leq y$ if and only if $x \wedge y = x$.

Let F be a set of functions $f: L \to L$ satisfying the following axioms.

A6: F is closed under function composition and $\wedge$, where $f \wedge g$ is the function h defined by $h(x) = f(x) \wedge g(x)$.

A7: There is a function $e \in F$ such that $e(x) = x$.

A8: $f(x \wedge y) = f(x) \wedge f(y)$.

B: For all $f \in F$ there is a function $f^* \in F$ such that $f^* g$ is the maximum solution to $fh \wedge g = h$.

Such a pair (L,F) is a global analysis framework.

Let $G = (V,E,r)$ be a flow graph, let (L,F) be a global analysis framework, let $f: E \to F$, and let $a: V \to L$. (L,F,G,f,a) is a global flow problem. The solution to this problem is the maximum solution to the set of equations

$$Q: \quad x(w) = \bigwedge_{(v,w) \in E} f(v,w)(x(v)) \wedge a(w) , \quad w \in V .$$

We can extend f to paths by defining $f(p) = f(v_k, v_{k+1})\, f(v_{k-1}, v_k) \cdot\cdot e\, f(v_1, v_2)$ if $p = (v_1, v_2)(v_2, v_3) \ldots (v_k, v_{k+1})$, and $f(p) = e$ if p is a path of no edges.

Observation 1. If $x(w)$ is a solution to Q and p is any path from a vertex v to a vertex w, then $x(w) \leq f(p)(a(v))$.

Under the assumed axioms, Q always has a unique maximum solution x such that $x(w)$ is the meet of $f(p)(a(v))$ for all paths p from v to w. The existence of the closure operation * guarantees that this meet of a possibly infinite set of paths exists and can be computed [4,22,23,27]. The asymptotically fastest method known for solving global flow problems uses a form of Gaussian elimination and achieves a time bound of $O(m\, \alpha(m,n))$ on reducible flow graphs [27].

For most practical global flow problems, the closure operation can be defined in terms of function meet and function composition. In such cases, it is possible to compute solutions using only function application and meet on L. We shall consider a hierarchy of global flow problems of this kind. Consider the following axioms.

Bk: $f^k(x) > \bigwedge_{i=0}^{k-1} f^i(x) \wedge f^k(1) \qquad (k \geq 1)$

Bks: $f^k(x) \geq \bigwedge_{i=0}^{k-1} f^i(x) \qquad (k \geq 1)$.

Any global flow problem whose framework satisfies Bk we call a k-bounded global flow problem.. Any global flow problem whose framework satisfies Bks and such that $a(w) = 1$ if $w \neq r$ we call a ks-bounded global flow -Problem.

Observation 2. Bks implies Bk . Bk implies B(k+1)s . Bk implies B .

The k-bounded and ks-bounded global flow problems form a hierarchy which includes some, but not all, of the global flow problems mentioned in the literature. The transitive closure [6,27] and dominators problems [p, 26,27] can be formulated as Is-bounded problems. Problems which use bit vectors, such as available expressions [28] and live variables [9,14] are 1-bounded but not Is-bounded. Problems which use "structured partition" lattices, such as common subexpression detection, [7,13,15] are 2s-bounded but not 1-bounded. Global flow problems involving type checking [30] are not k-bounded unless some bound is artificially imposed.

Kam and Ullman [13] have shown that 1-boundedness is a necessary and sufficient condition for fast convergence of Hecht and Ullman's version of Kildall's algorithm. We shall show that there is a general iterative algorithm, an extension of Kennedy's node listing method, for solving any k-bounded or ks-bounded problem. The algorithm is optimal, among all iterative algorithms, for each k . We give a lower bound on the running time of the algorithm, a bound which shows that the algorithm becomes markedly less efficient, and thus less competitive with the best elimination algorithm, as k increases.

Let (L,F,G,f,a) be a global flow problem, with G = (V,E,r) . Let S be a sequence of edges of G . Consider the following algorithm.

```
procedure ITERATE (set V, set E, vertex r, function f, function a,
                   list s, function x)
    begin
        for w∈V do x(w) := a(w);
        for (v,w)∈S do x(w) := x(w) ∧ f(v,w)(x(v));
    end ITERATE;
```

This algorithm propagates information along paths which are subsequences of s .

Observation 3. Any function x computed by ITERATE satisfies $x(w) = \wedge \{f(p(v,w))(a(v)) \mid p(v,w) \in P\}$, where P is some set of paths leading to w .

Theorem 1. If (L,F) is ks-bounded, S is a ks-sequence for G , and $a(w) = 1$ for $w \neq r$, then the function x computed by ITERATE is a maximum solution to Q .

Proof. Let $y(w) = \wedge \{f(p)(a(r)) \mid p$ a ks-path to $w\}$ for $w \in V$. Let z be any solution to Q . By Observations 1 and 3, $z \leq x$. It is easy to prove by induction on the length of p that $x(w) \leq f(p)(a(r))$ for any ks-path to w . Thus $x \leq y$. It remains to be shown that y is a solution to Q .

$$y(w) = \wedge \{f(p)(a(r)) \mid p \text{ a ks-path to } w\}$$

$$= \bigwedge_{(v,w)\in E} f(v,w)(\wedge \{f(p)(a(r)) \mid p \in P(v)\})$$

where P(v) is a suitable subset of ks-paths to v

$$\geq \bigwedge_{(v,w)\in E} f(v,w)(y(v)) \quad .$$

Let $(v,w) \in E$. Let p be a ks-path to v . If $p(v,w)$ is a ks-path, then $f(p(v,w))(a(r)) \geq y(w)$. Otherwise, $p(v,w) = p_0 p_1 p_2 \bullet p_k$, where p_i starts and ends at w for $1 \leq i \leq k$, and p_0 is possibly empty. Then

$$f(p(v,w))(a(r)) \geq \left(\bigwedge_{i=1}^{k} f(p_i) \right)^k f(p_0)(a(r))$$

$$\geq \bigwedge_{j=0}^{k-1} \left(\bigwedge_{i=1}^{k} f(p_i) \right)^j f(p_0(a(r)) \qquad \text{by Bks}$$

$$= \wedge \{f(p')(a(r)) \mid p' \in P\} \qquad \text{where P is a suitable subset of paths from r to w , each a proper subsequence of } p(v,w) .$$

By applying the same decomposition repeatedly, we eventually have

$$f(p(v,w))(a(r)) \geq \wedge \{f(p')(a(r)) \mid p' \in P'\} \qquad \text{where } P' \text{ is a set of ks-paths from r to w}$$

$$\geq y(w) .$$

It follows-that $y(w) \leq \bigwedge_{(v,w) \in E} f(v,w)(y(v))$, and y is a solution to Q . Thus $y = x$ and x is the maximum solution to Q . □

Theorem 2. If (L,F) is k-bounded, and S is a k-sequence for G , then the function x computed by ITERATE is a maxim-urn solution to Q .

Proof. -Let $y(w) = \wedge \{f(p(v,w))(a(v)) \mid p \text{ a k-path to } w\}$. Let z be any solution to Q . As in the proof of Theorem 1, $z \leq x \leq y$, and we must show that y is a solution to Q .

$$y(w) = \wedge \{f(p(v,w))(a(v)) \mid P \text{ a k-Path to } w\}$$

$$= \bigwedge_{(v,w)\in E} f(v,w)(\wedge \{f(p(u,v))(a(u)) \mid p\in P(v)\}) \quad \text{where } P(v) \text{ is a suitable subset of k-paths to v}$$

$$\geq \bigwedge_{(v,w)\in E} f(v,w)y(v) \quad .$$

Let $(v,w)\in E$. Let p be a k-path to v . If $p(v,w)$ is a k-path, then $f(v,w)(a(v)) \geq y(w)$. Otherwise, $p(v,w) = p_0 p_1 p_2 \cdots p_k$ where p_i starts and ends at w for $1 \leq i \leq k$, and p_0 is non-empty. Then

$$f(p(v,w))(a(v)) \geq \left(\bigwedge_{i=1}^{k} f(p_i) \right)^k f(p_0)(a(v))$$

$$\geq \bigwedge_{j=0}^{k-1} \left(\bigwedge_{i=1}^{k} f(p_i) \right)^j f(p_0)(a(v)) \wedge \left(\bigwedge_{i=1}^{k} f(p_i) \right)^k (1)$$

$$\geq \wedge \{f(p'(u,w))(a(u)) \mid p'\in P\} , \quad \text{where P is a suitable subset of paths to w , each a proper subsequence of } p(v,w) .$$

By repeating this decomposition, we eventually have

$$f(p(v,w))(a(v)) \geq \wedge \{f(p'(u,w))(a(u)) \mid p'\in P'\} , \quad \text{where } P' \text{ is some set of k-paths to w .}$$

It follows that $y(w) \leq \bigwedge_{(v,w)\in E} f(v,w)(y(v))$, and y is a solution to Q . Hence $y = x$ and x is the maximum solution to Q . □

ITERATE gives a uniform method for solving k- and ks-bounded global flow problems, with the length of the necessary sequence S dependent upon k . The ks-bounded problems require propagation only from the start vertex; the k-bounded problems require propagation from all vertices. We have left unresolved the problem of finding a k- or ks-sequence to use as input to ITERATE.

Kennedy's algorithm as originally stated is the version of ITERATE which solves 1-bounded global flow problems. Aho and Ullman [1] have given a method for constructing, in O(n log n) time, an O(n log n) length 1-sequence for any reducible program flow graph. Thus ITERATE can be implemented to solve 1-bounded problems on reducible program flow graphs in O(n log n) time.

Hecht and Ullman's [9] depth-first ordering gives a 1-sequence of O(dn) length for any reducible program flow graph, where d is the largest number of "cycle" edges [13,27] on any 1-path. For typical FORTRAN programs, $d \leq 2.75$ [17]. Thus the depth-first ordering gives a linear time implementation for typical programs, although the worst case is $O(n^2)$.

Lemmas 1, 2, and 3 give lower bounds on the lengths of k- and ks-sequences , and thus on the worst case running time of all implementations of ITERATE. We shall see in the next section that these lower bounds apply not only to ITERATE, but to <u>any</u> iterative algorithm for solving k- or ks-bounded problems.

4. Lower Bounds on Iterative Algorithms.

To provide lower bounds on the number of operations required to solve k- or ks-bounded problems by iterative algorithms, we will construct certain "worst-case" global flow frameworks. Let $(L_1,F_1),\ldots,(L_\ell,F_\ell)$ be global flow frameworks. We can define a cross product framework $(L_1 \times L_2 \times \cdots \times L_\ell\ ,\ F_1 \times F_2 \times \cdots \times F_\ell)$, where operations are performed component-wise. That is, $(f_1,\ldots,f_\ell)(x_1,\ldots,x_\ell) = (f(x_1),\ldots,f(x_\ell))$. It is easy to show that $(L_1 \times L_2 \times \ldots \times L_\ell\ ,\ F_1 \times F_2 \times \cdots \times F_\ell)$ is a global flow framework with zero element $(0,\ldots,0)$, one element $(1,\ldots,1)$, and identity function $(e,\ldots,e)$. Furthermore $(x_1,x_2,\ldots,x_\ell) \leq (y_1,y_2,\ldots,y_\ell)$ if and only if $x_i \leq y_i$ for all i. Also, $(L_1 \times \ldots \times L_\ell\ ,\ F_1 \times \ldots \times F_\ell)$ is k-bounded (ks-bounded) if all the (L_i,F_i) are k-bounded (ks-bounded).

Let $G = (V,E,r)$ be a flow graph. Let $k \geq 2$ and let p be any ks-path of G. Let $L_s(p)$ be the semi-lattice defined by

$L_s(p) = \{0,1\} \cup \{P(v) \mid P(v)$ is a non-empty set of subsequences of p, each of which is a path from r to $v\}$

(Λ, the empty path from r to r, is an allowable subsequence if v = r)

$x \wedge 0 = 0 \wedge x = 0$

$x \wedge 1 = 1 \wedge x = x$

$$P_1(v) \text{ A } P_2(w) = \begin{cases} P_1(v) \cup P_2(w) & \text{if } v = w \\ 0 & \text{if } v \neq w \end{cases}.$$

It is clear that $L_s(p)$ satisfies A0 - A5.

Let $F_s(p)$ be the smallest set of functions closed under meet and composition which contains the identity function e, the function $\underline{1}$ such that $\underline{1}(x) = 1$, and a function $f_p(v,w)$ for each edge (v,w) in p, defined by

$$f_p(v,w)(0) = 0$$

$$f_p(v,w)(1) = 1$$

$$f_p(v,w)(P(v)) = \begin{cases} \{p_1(v,w) \mid p_1 \in P(v) \text{ and } p_1(v,w) \text{ is a subsequence of } p\} & \text{if this set is non-empty} \\ 1 & \text{otherwise} \end{cases}$$

$$f_p(v,w)(P(u)) = 0 \text{ if } u \neq v .$$

Lemma 4. $(L_s(p), F_s(p))$ is a ks-bounded global flow framework.

proof. A6 and A7 hold by definition. Consider A8. Suppose $f(x \wedge y) = f(x) \wedge f(y)$ and $g(x \wedge y) = g(x) \wedge g(y)$. Then $fg(x \wedge y) = f(g(x) \wedge g(y)) = fg(x) \wedge fg(y)$ and $(f \wedge g)(x \wedge y) = f(x) \wedge g(x) \wedge g(y) \wedge f(y) = (f \wedge g)(x) \wedge (f \wedge g)(y)$. Thus we need verify A8 only for e, $\underline{1}$ and $f_p(v,w)$. A8 clearly holds for e and 1. Consider $f_p(v,w)$.

$$f_p(v,w)(x \wedge 0) = f_p(v,w)(0) = 0 = f_p(v,w)(x) \wedge f_p(v,w)(0)$$

$$f_p(v,w)(x \wedge 1) = f_p(v,w)(x) = f_p(v,w)(x) \wedge 1 = f_p(v,w)(x) \wedge f_p(v,w)(1)$$

$$f_p(v,w)(P_1(v) \wedge P_2(v)) = f_p(v,w)(P_1(v) \cup P_2(v))$$

$$= f_p(v,w)(P_1(v)) \wedge f_p(v,w)(P_2(v))$$

$$f_p(v,w)(P_1(u) \wedge P_2(x)) = 0 = f_p(v,w)(P_1(u)) \wedge f_p(v,w)(P_2(x))$$

$$\text{if } u \neq v \text{ or } x \neq v .$$

In all cases A8 holds.

To prove Bks , consider any function $g \in F_s(p)$. We can write $g = \bigwedge_{i=1}^{j} g_i$, where each g_i is either e or a composition of functions $f_p(v,w)$, possibly followed by 1 .

We wish to prove Bks : $g''(x) \geq A \sum_{i=1}^{k-1} g^i(x)$. We need only prove this inequality for x = 0 , x = 1 , or x containing a single ks-path. The result is obvious for x = 0 or x = 1 . Let x be a ks-path.

Consider any term $g_{i(1)}\, g_{i(2)} \cdots g_{i(k)}(\{x\})$ of $g^k(\{x\}) = \left(\bigwedge_{i=1}^{j} g_i \right)^k (\{x\})$. If $g_{i(k)}(\{x\})$ does not denote a ks-path, then either the left side of Bks is 1 or the right hand side of Bks is 0, and Bks holds. If $g_{i(k)}(\{x\})$ is a ks-path but $g_{i(k)}(\{x\})$ does not end at the end of x , then $g_{i(k)}(\{x\}) A \{x\} = 0$ and the right side of Bks is zero. Extending this argument, we can show that Bks holds unless $g_{i(1)} \cdots g_{i(k)}(\{x\})$ is a ks-path which contains the last vertex of x k+1 times. But this is impossible. Thus Bks holds. □

Lemma 5. Let p be a ks-path in G . Consider the global flow problem

$$(L_s(p), F_s(p), G, g, a) \text{ , where } g(v,w) = \begin{cases} f_p(v,w) & \text{if } (v,w) \text{ is on } p \\ \underline{1} & \text{otherwise} \end{cases}$$

$$a(r) = A$$

$$a(w) = 1 \qquad \text{if } w \neq r \text{ .}$$

The solution to this problem is

$$x(w) = \begin{cases} \{p' \mid p' \text{ is a ks-path to } w \text{ which is a subsequence of } p \\ \qquad \text{if this set is non-empty}\} \\ 1 \qquad \text{otherwise.} \end{cases}$$

Proof. Similar to the proof of Theorem 1. □

To deal with the case of Is-bounded problems, we shall use a special construction. Let $G = (V,E,r)$ be any flow graph and let p be any ls-path of G . Let $L_s(p)$ be the semi-lattice defined by

$$L_s(p) = \{0,1\} \cup \{p' \mid p' \text{ is an initial segment of p } (p' = \Lambda \text{ is included})\}$$

$$0 \wedge x = x \wedge 0 = 0 \quad , \quad 1 \wedge x = x \wedge 1 = x$$

$$p_2 \wedge p_1 = p_1 \wedge p_2 = p_1 \qquad \text{if } p_1 \text{ is an initial segment of } p_2 .$$

Let $F_s(p)$ be the smallest set of functions closed under meet, composition, and containing e , $\underline{1}$, and a function $f_p(v,w)$ for each edge (v,w) on p , defined by

$$f_p(v,w)(0) = 0$$

$$f_p(v,w)(1) = 1$$

$$f_p(v,w)(p') = \begin{cases} p'(v,w) & \text{if this is an initial segment of p} \\ p' & \text{otherwise.} \end{cases}$$

Lemma 6. $(L_s(p) , F_s(p))$ is a Is-bounded global flow framework.

Proof. A0-A7 hold obviously. A8 clearly holds for e and $\underline{1}$. Consider $f_p(v,w)$.

$$f_p(v,w)(x \wedge 0) = f_p(v,w)(0) = 0 = f_p(v,w)(x) \wedge f_p(v,w)(0)$$

$$f_p(v,w)(x \wedge 1) = f_p(v,w)(x) = f_p(v,w)(x) \wedge f_p(v,w)(1)$$

$$f_p(v,w)(p_1 \wedge p_2) = f_p(v,w)(p_1) = p_1(v,w)$$

$$= f_p(v,w)(p_1) \wedge p_2 = f_p(v,w)_{(P,)} \wedge f_p(v,w)(p_2)$$

if $p_1 < p_2$ and p_1 ends at v

$$f_p(v,w)(p_1 \wedge p_2) = f_p(v,w)(p_1) = p_1 = f_p(v,w)(p_1) \text{ A } f_p(v,w)(p_2)$$

if $p_1 < p_2$ and p_1 ends other than at v .

In all cases A8 holds.

Note that the functions $f \in (e, \underline{1}, f_p(v,w)\}$ satisfy $f \geq e$. It follows that all functions f in $F_s(p)$ satisf'y $f \geq e$. Thus $(L_s(p), F_s(p))$ is a Is-bounded global flow framework. □

Lemma 7. Let p be a Is-path in G . Consider the global flow problem

$$(L_s(p), F_s(p), G, g, a) \quad \text{where} \quad g(v,w) = \begin{cases} \tilde{f_p}(v,w) & \text{if } (v,w) \text{ is on } p \\ 1 & \text{otherwise} \end{cases}$$

$$a(r) = A$$

$$a(w) = 1 \qquad \text{if } w \# r \ .$$

The solution to this problem is

$$x(w) = \begin{cases} p' & \text{if } p' \text{ starts at r , ends at w , and is an initial segment of p} \\ 1 & \text{if there is no path from r to w which is an initial segment of p .} \end{cases}$$

proof. Similar to the proof of Theorem 1. □

Let $G = (V,E,r)$ be a flow graph. Let $k \geq 1$ and let p be any k-path of G starting at some vertex s. Let $L(p)$ be the semi-lattice defined by

$L(P) = \{0,1\} \cup \{P(v) \mid F(v)$ is a non-empty set of subsequences of p, each a path to v, such that $P(v)$ contains $\Lambda(v)$, the empty path from v to $v]$

$x \wedge 0 = 0 \wedge x = 0$

$x \wedge 1 = 1 \wedge x = 0$

$$P_1(v) \wedge P_2(w) = \begin{cases} P_1(v) \cup P_2(w) & \text{if } v = w \\ 0 & \text{if } v \neq w \ . \end{cases}$$

It is clear that $L(p)$ satisfies A0 - A5.

Let $F(p)$ be the smallest set of functions closed under meet and composition which contains the identity function e, the function 1 such that $\underline{1}(x) = \underline{1}$, and a function $f_p(v,w)$ for each edge (v,w) on p, defined by

$f_p(v,w)(0) = 0$

$f_p(v,w)(1) = \{(v,w),\Lambda(w)\}$

$f_p(v,w)(P(v)) = \{p_1(v,w) \mid p_1 \in P(v)$ and $p_1(v,w)$ is a subsequence of $p]$

$\cup \{\Lambda(w)\}$

$f_p(v,w)(P(u)) = 0$ if $u \neq v$.

Lemma 8. $(L(p), F(p))$ is a k-bounded global flow framework.

Proof. A6 and A7 hold by definition. Consider A8. A8 clearly holds for e and $\underline{1}$. Consider $f_p(v,w)$.

$$f_p(v,w)(x \wedge 0) = f_p(v,w)(0) = 0 = f_p(v,w)(x) \wedge f_p(v,w)(0)$$

$$f_p(v,w)(P(v) \wedge 1) = f_p(v,w)P(v) = f_p(v,w)(P(v)) \wedge \{(v,w), \Lambda(w)\}$$

$$= f_p(v,w)(P(v)) \wedge f_p(v,w)(1)$$

$$f_p(v,w)(P(u) \wedge 1) = f_p(v,w)(P(u)) = 0 = f_p(v,w)(P(u)) \wedge f_p(v,w)(1)$$

if $u \neq v$

$$f_p(v,w)(P_1(v) \wedge P_2(v)) = f_p(v,w)(P_1(v) \cup P_2(v))$$

$$= f_p(v,w)(P_1(v)) \wedge f_p(v,w)(P_2(v))$$

$$f_p(v,w)(P_1(u) \wedge P_2(x)) = 0 = f_p(v,w)(P_1(u)) \wedge f_p(v,w)(P_2(x))$$

if $u \neq v$ or $x \neq v$.

In all cases A8 holds.

To prove Bk, suppose $k \geq 2$ and consider any function $g \in F(p)$. We can write $g = \bigwedge_{i=1}^{j} g_i$, where each g_i is either e or a composition of functions $f_p(v,w)$, possibly followed by $\underline{1}$. We wish to prove Bk: $g^k(x) > \bigwedge_{i=1}^{k-1} g^i(x) \wedge g^k(1)$. We need only prove this inequality for $x = o$, $x = 1$, or x of the form $x = \{p', \Lambda(w)\}$ where p' is a path from v to w. The result is obvious for x = 0 and for x = 1. Suppose $x = \{p', \Lambda(w)\}$, where p' is a path from v to w.

Consider any term $g_{i(1)}\, g_{i(2)} \cdots g_{i(k)}(x)$ of $g^k(x) = \left(\bigwedge_{i=1}^{j} g_i \right)^k (x)$. If any $g_{i(j)}$ contains $\underline{1}$, Bk holds. If $g_{i(k)}(x)$ is not a set of k-paths, the right side of Bk is 0 and Bk holds. If $g_{i(k)}(x)$ does

not denote a set of k-paths ending at w , $g_{i(k)}(x)$ $Ax = 0$ and Bk holds. Extending this argument, we can show that for all $1 \leq j \leq k$, $g_{i(j)} \cdots g_{i(k)}(x)$ is a set of k-paths ending at w . The only possible kind of path occuring in $g_{i(1)}\, g_{i(2)} \cdot * \cdot\, g_{i(k)}(x)$ which does not occur in a set on the right side of Bk is a path of the form $p_{i(1)} p_{i(2)} \cdots p_{i(k)}\, p'$, where $p_{i(j)}$ is the sequence of edges corresponding to the functions composed to form $g_{i(j)}$ and p' is non-empty. But such a path is not a k-path and thus does not occur in $g_{i(1)} \cdot \cdot \cdot g_{i(k)}(x)$. Hence Bk holds. A similar argument shows that Bk holds if $k = 1$. □

Lemma 9. Let p be a k-path in G starting at s . Consider the global flow problem

$$(L(p),F(p),G,g,a) \text{ , where } g(v,w) = \begin{cases} f_p(v,w) & \text{if } (v,w) \text{ is on } p \\ 1 & \text{otherwise} \end{cases}$$

$$a(s) = \{\Lambda(s)\}$$

$$a(w) = 1 \qquad \text{for } w \neq s \text{ .}$$

The solution to this problem is

$$. \; x(w) = \begin{cases} \{p' \mid p' \text{ is a k-path to w which is a subsequence of p}\} & \text{if w lies on p} \\ 1 & \text{otherwise.} \end{cases}$$

Proof. Similar to the proof of Theorem 2. □

Consider any algorithm, which, starting from the values a(v) , 0, 1, computes a solution to the global flow problem (L,F,G,f,a) by computing

meets of elements in L and applying functions in $\{f(v,w)\}$. We call such an algorithm an iterative global flow algorithm. The derived sequence of such an algorithm is the list of edges (v,w) such that $f(v,w)$ is applied by the algorithm, with the edges occuring in the order the functions are applied. We provide lower bounds on the number of operations required by iterative algorithms by showing properties of their derived sequences.

Theorem 3. Let $G = (V,E,r)$ be any flow graph. Consider the global flow problem

$$(L,F) = \bigwedge\{(L_s(p), F_s(p)) \mid p \text{ a ks-path}\}.$$

Let

$$f(v,w) = (g_p(v,w)), \text{ where } g_p(v,w) = \begin{cases} f_p(v,w) & \text{if } (v,w) \text{ lies on } p \\ \underline{1} & \text{otherwise.} \end{cases}$$

Let $a(r) = (\Lambda,\Lambda,\ldots,\Lambda)$, $a(w) = (1,1,\ldots,1)$ for $w \neq r$. Then the derived sequence for any iterative algorithm which solves (L,F,G,f,a) must contain a ks-sequence.

proof. Let $p=(r=v_1,v_2)(v_2,v_3)\ldots(v_k,v_{k+1}=w)$ be a ks-path. Consider the p-component of the solution x to Q. The only way to build up the correct value in the p component of x(w) is to apply $f(v_1,v_2),\ldots,f(v_k,v_{k+1})$ in sequence. □

Theorem 4. Let $G = (V,E,r)$ be any flow graph. Consider the global flow problem

$$(L,F) = \bigwedge\{(L(p), F(p)) \mid p \text{ a k-path}\}.$$

Let

$$f(v,w) = (g_p(v,w)) \text{ , where } g_p(v,w) = \begin{cases} f_p(v,w) & \text{if } (v,w) \text{ lies on } p \\ 1 & \text{otherwise.} \end{cases}$$

Let

$$a(w) = (a_p(v)) \text{ , where } a_p(v) = \begin{cases} \Lambda(v) & \text{if } p \text{ starts at } v \\ 1 & \text{otherwise.} \end{cases}$$

Then the derived sequence for any iterative algorithm which solves (L,F,G,f,a) must contain a k-sequence.

Proof. Similar to the proof of Theorem 3. □

Thus Theorems 1, 2, 3, and 4 characterize the exact number of function applications needed to solve a k- or ks-bounded global flow problem in the worst case. This number is equal to the length of the shortest k- or ks-sequence for the graph. Lemmas 1, 2, and 3 give lower bounds on this length, and hence we have the following corollaries:

Theorem 5. In the worst case, the solution of a k- or ks-bounded global flow problem on an n-vertex (non-reducible) program flow graph requires at least $c\,kn^2$ function applications, if no function compositions or function meets are used.

Proof. Immediate from Corollary 1 and Theorem 3. □

Theorem 6. In the worst case, the solution of a k- or ks-bounded global flow problem on an n-vertex reducible program flow graph requires at least $c(k)n^{\log_k s(k)}$ function applications, if no function compositions or function meets are allowed.

proof. Immediate from Lemma, 3 and Theorem □

Theorem 7. In the worst case, the solution of a 1-bounded global flow problem on an n-vertex reducible program flow graph requires $cn \log n$ function applications if function compositions and function meets are not used.

Proof. Immediate from Lemma 1 and Theorem 4. □

Some global flow problems, notably the live variables problem [9,10] require propagating information backward through the graph. By Corollary 2, the lower bound of Theorem 6 applies to program flow graphs whose reverses are reducible. We also have the following lower bound.

Theorem 8. In the worst case, the solution of a Is-bounded global flow problem on an n-vertex program graph whose reverse is reducible requires cn log n function applications if function compositions and function meets are not used.

proof. From Lemma 1 and Theorem 4. 0

Our worst case global flow problems are somewhat contrived. However, it is possible, for instance, to construct a worst-case bit-vector type problem and use it in place of (L_P, F_P) in the 1-bounded case.

Let $G = (V,E,r)$ be any flow graph and let $P = (v_1,v_2)(v_2,v_3)\ldots(v_\ell,v_{\ell+1})$ be any-1-path of G. Let L_P be the semi-lattice defined by

$$L_P = (s \mid s \subseteq \{x_{ij} \mid 0 \leq i \leq j \leq \ell\}$$

$$S_1 \wedge S_2 = S_1 \cap S_2 .$$

Let

$$o = \emptyset \quad \text{and} \quad 1 = \{x_{ij} \mid 0 \leq i \leq j \leq \ell\} \ .$$

Clearly L_p satisfies A0 - A5.

Let F_p be the set of functions closed under composition and intersection and containing the identity function e, the function $\underline{1}(x) = 1$, and a function $f_p(v_i, v_{i+1})$ for each edge (v_i, v_{i+1}), defined by

$$f_p(v_i, v_{i+1})(S) = s - (x_{ij} \mid i \leq j \leq \ell\} \cup \{x_{ji} \mid 0 \leq j < i\} \ .$$

The pair (L_p, F_p) is an example of a typical global flow framework for available expressions or any similar bit vector type problem and is 1-bounded [12].

Let (L_p, F_p, G, g, a) be the global framework with

$$g(v,w) = \begin{cases} f_p(v, w) & \text{if } (v,w) \text{ is on } p \\ 1 & \text{otherwise} \end{cases}$$

$$a(v_1) = \{x_{ij} \mid 1 \leq i \leq j \leq \ell\} \cup \{x_{0j} \mid 1 \leq j \leq \ell\}$$

$$a(w) = 1 \quad \text{for } w \neq v_1 \ .$$

If x is the solution to this problem, a computation shows that $x(v_{\ell+1}) = \{x_{ij} \mid 1 < i < j < \ell\} \cup \{x_{0j} \mid 1 < j < \ell\}$. In order to compute $x(v_{\ell+1})$ from 0 , 1 , or $a(v_1)$, $f_p(v_i, v_{i+1})$ for $i = 1, 2, \ldots, \ell$ must be applied in sequence.

Thus, by using the appropriate product framework, we can show that Theorems 4, 5, 7 hold for all iterative algorithms which solve bit-vector type problems. We have not tried constructing "natural" worst-case examples for other values of k .

Remarks.

We have exhibited a hierarchy of global flow problems, an optimal iterative algorithm for solving any problem in the hierarchy, and lower bounds on the time necessary for solving worst-case problems in each level of the hierarchy. For k- and ks-bounded problems on non-reducible program graphs, the lower bound is $c\,kn^2$, and this bound is tight to within a constant factor independent of the boundedness. For 1-bounded problems on reducible program graphs, the lower bound is $O(n \log n)$, which is tight to within a constant factor by a result of Aho and Ullman [1]. For k- and ks-bounded problems on reducible program graphs, the lower bound is $c(k)n^{\log_k s(k)}$. Both the tightness of this bound and the exact value of $s(k)$ are unknown.

The lower bounds indicate that, at least theoretically, iterative methods become markedly less competitive with elimination methods (which have an $O(n\,\alpha(n,n) \log k)$ running time for k-bounded problems on reducible program flow graphs) as K increases. For instance, for $k = 2$, the lower bound on iterative algorithms is $cn^{\log_2 3} > cn^{1.59}$. Of course, real-world problems may exhibit a different behavior, especially since iterative algorithms are so easy to program.

A natural next step in this research would be to prove a non-trivial (i.e., $c\,n\alpha(n,n)$) lower bound on the time required by any elimination algorithm for solving global flow problems on reducible program flow graphs. The lower bounds in [24,26] are probably relevant to this question.

Acknowledgments. My thanks to Prof. Jeffrey Ullman, for helpful discussions on the ideas used in the lower bound results, and to Prof. Richard Karp, for suggesting an improvement in the result of Lemma 3.

References

[1] A. V. Aho and J. D. Ullman, "Node listings for reducible flow graphs," Proc. Seventh Annual ACM Symposium on Theory of Computing (1975), 177-185.

[2] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. II: Compiling, Prentice-Hall, Englewood Cliffs, N.J. (1972).

[3] F. E. Allen, "Control flow analysis," SIGPLAN Notices, Vol. 5 (1970), 1-19.

[4] R. C. Backhouse and B. A. Carré, "Regular algebra applied to path-finding problems," J. Inst. Maths. Applics., Vol. 15 (1975), 161-186.

[5] J. Cocke, "Global common subexpression elimination," SIGPLAN Notices, Vol. 5 (1970), 20-24.

[6] J. Eve, "On computing the transitive closure of a relation," STAN-CS-75-508, Computer Science Dept., Stanford University (1975).

[7] A. Fong, J. Kam, and J. Ullman, "Application of lattice algebra to loop optimization," Conf. Record of the Second ACM Symposium on Principles of Prog. Lang. (1975), 1-p.

[8] S. Graham and M. Wegman, "A fast and usually linear algorithm for global flow analysis," J. ACM, vol. 29 (1975), 172-202.

[9] M. S. Hecht and J. D. Ullman, "A simple algorithm for global flow analysis problems," SIAM J. Comp., to appear.

[10] M. S. Hecht and J. D. Ullman, "Flow graph reducibility," SIAM J. Comp., Vol. 1 (1972), 188-202.

[11] M. S. Hecht and J. D. Ullman, "Characterizations of reducible flow graphs," J. ACM., Vol. 21 (1974), 367-375.

[12] J. Kam and J. D. Ullman, "Global optimization problems and iterative algorithms," J. ACM., Vol.23 (1976), 158-171.

[13] J. Kam and J. D. Ullman, "Monotone data flow analysis frameworks," unpublished report, Princeton University (1975).

[14] K. W. Kennedy, "Node listings applied to data flow analysis," Conf. Record of the Second ACM Symposium on Principles of Prog. Lang. (1973, 10-21.

[15] G. A. Kildall, "A unified approach to global program optimization," Conf. Record of the ACM Symposium on Principles of Prog. Lang. (1973), 194-206.

[16] S. C. Kleene, "Representation of events in nerve nets and finite automata," Automata Studies, Shannon and McCarthy, eds., Princeton University Press, Princeton, N.J. (1956), 3-40.

[17] D. E. Knuth, "An empirical study of FORTRAN programs," Software Practice and Experience (1971), 105-134.

[18] P. J. Koutas and T. C. Hu, 'Shortest string containing all permutations," Discrete Mathematics, Vol. 11 (1975), 125-132.

[19] D.J. Kwiatowski and D. J. Kleitman, "A lower bound on the length of a sequence containing all permutations as subsequences," submitted to J. Comb. Theory, Series A.

[20] G. Markowsky and R. Tarjan, "Lower bounds on the lengths of node sequences in directed graphs," Discrete Mathematics, to appear.

[21] M. Newey, "Note on a problem involving permutations as subsequences," STAN-CS-73-340, Computer Science Dept., Stanford University (1973).

[22] A. Salomaa, "Two complete axiom systems for the algebra of regular events," J. ACM., Vol. 13 (1966), 158-169.

[23] A. Salomaa, Theory of Automata, Pergamon Press, Oxford, England (1969), 120-127.

[24] R. Tarjan, "Efficiency of a good but not linear set union algorithm," J. ACM., Vol. 22 (1975), 215-225.

[25] R. Tarjan, "Testing flow graph reducibility," J. Comp. Sys. Sciences, Vol. 9 (1974), 355-365.

[26] R. Tarjan, "Applications of path compression on balanced trees," STAN-CS-75-512, Computer Science Dept., Stanford University (1975).

[27] R. Tarjan, "Solving path problems on directed graphs," STAN-M-75-528, Computer Science Dept., Stanford University (1975).

[28] J. D. Ullman, "A fast algorithm for the elimination of common subexpressions," Acta Informatica, Vol. 2 (1973), 191-213.

[29] V. A. Vyssotsky, private communication to M. S. Hecht, 1973.

[30] B. Wegbreit, "Property extraction in well-founded property sets," Computer Science Division, Bolt Beranek and Newman, Inc., Cambridge, Mass. (1973).