2. Algorithms on Graph

2.1 Basic Algorithms

Let R be a relation on A. Find $R^*(B)$ for $B \subseteq A$.

directed graph(A, R)

Depth First Search

```
procedure Traverse(node a);
begin
    mark a;
    for each edge (a, b) ∈ R do
        if b is unmaked then
            Traverse(b)
    od  fi
end
begin
    unmark all node in A;
    for each node a ∈ B do Traverse(a) od
end.
```

Theorem 2.2 *Let R be a relation on a finite set A. Give a subset B of A, $R^*(B)$ can be computed in time $O(max\{|R|, |A|\})$.*

Theorem 2.3 *Let R be a relation on a finite set A. R^* can be computed in time $O(|A|\cdot|R|)$.*

2.2 Finding Strongly Connected Components

Strongly connected components

Scc(a): equivalence class of A under the equivalence relation $R^* \cap (R^{-1})^*$ *on A.*

Scc(a) = Scc(b), iff a R^* *b and b* R^* *a.*

A substring of a depth-first traversal that begins with ***enter****(a) and ends with* ***exit****(a) is called a* ***depth first traversal of*** *a.*

Let (A, R) be a graph and π *be a depth-first traversal.*
depth-first order *of (A, R)* ***with respect to*** π

$\tau o \tau \alpha \lambda$ $o \rho \delta \varepsilon \rho$ $\leq$ *on A,*

a < b, iff ***enter****(a) proceeds* ***enter****(b) in* π*; or a is* <u>*marked*</u> *before b*

Fact 2.4 *A depth-first order is* ***properly nested****.*

*(1) …****enter****(a)…****exit****(a)…****enter****(b)…****exit****(b)…*
(2) *…****enter****(b)…****exit****(b)…****enter****(a)…****exit****(a)…*
(3) *…****enter****(a)…****enter****(b)…****exit****(b)…****exit****(a)…*
*(4) …****enter****(b)…****enter****(a)…****exit****(a)…****exit****(b)…*

(1) a is traversed ***before*** *b.*
(2) a is traversed ***after*** *b.*
(3) b is traversed ***during*** *the traversal of a.*
(4) a is traversed ***during*** *the traversal of b.*

Fact 2.5 *Let π is the of form*

*… **enter**(a) … **enter**(b) … **exit**(b) … **exit**(a) …,*

*(b is traversed **during** traversal of a)*

if and only if $a\ (R \cap <)^{+}\ b$.

Lemma 2.6

$\boldsymbol{a} \le b \le c$ *and* $a\ (R \cap <)^{*}\ c \Rightarrow a\ (R \cap <)^{*}\ b$

Proof.

If a = b or b = c is trivial.

So assume that a < b < c.

*… **enter**(a) … **enter**(b) … **enter**(c) …*

$a\ (R \cap <)^{*}\ c$ *and* $a \neq c$ *implies*

*… **enter**(a) … **enter**(c) … **exit**(c) … **exit**(a) …*

$\therefore$ ***enter***(b) *must be after* ***enter***(a).

$\therefore$ *… **enter**(a) … **enter**(b) … **exit**(b) … **exit**(a) …*

$\therefore\ a\ (R \cap <)^{*}\ b$.

Lemma 2.7 *Let* $(a_0, \dots, a_n)$ *be a path of length n.*

(1) $a_0 (R \cap <)^* a_n$; ***or***

(2) $^{\exists}i, 0 \le i < n, a_0 (R \cap <)^* a_i$ *and* $a_{i+1} < a_0$.

Proof *Let k be the number of back edges in* $(a_0, \dots, a_n)$,

i.e., (a_i, a_{i+1}) *such that* $a_{i+1} \le a_i$.

If k=0, statement (1) is tautology.

Assume that k-back edges with k>0, and let

$j = min\{i \mid i \le 0 < n, a_{i+1} \le a_i\}$.

Then $0 \le^{\forall} j < n$: $a_0 (R \cap <)^* a_j$, *and* $a_{j+1} \le a_j$.

If $a_{j+1} < a_0$, *statement (2) holds.*

If $a_0 \le a_{j+1}$, *then* $a_0 (R \cap <)^* a_{j+1}$. *Hence*

$^{\exists}$*path* $(b_0, \dots, b_m)$ *in* $(A, R \cap <)$, $b_0 = a_0$, $b_m = a_{j+1}$.

Then $(b_0, \dots, b_m, a_{j+2}, \dots, a_n)$ *is a path in (A, R)*

with k-1 back edges.

$\therefore$ $a_0 (R \cap <)^* a_n$ *or*

$j+1 \le^{\forall} l < n$: $a_0 (R \cap <)^* a_l$ *and* $a_{l+1} < a_0$.

Lemma 2.8 *Let* $(a_0, \ldots, a_n)$ *be a path of length n.*

(1) $a_0 \, (R \cap <)^* \, a_n$; ***or***

(2) $^{\exists}j$, $0<j\leq n$, $a_j < a_0$ *and* $a_j \, (R \cap <)^* \, a_n$.

Lemma 2.9

$a \leq b \leq c$ *and* $a \, R^* \, c \Rightarrow a \, R^* \, b$.

Proof.

If $a = b$ *or* $b = c$ *is trivial.*

So assume that $a < b < c$.

… ***enter***(*a*) … ***enter***(*b*) … ***enter***(*c*) …

$a \, R^* \, c$ *and implies (1)* $a \, (R \cap <)^* \, c$ *or*

(2) $e < a$, $a \, R^* \, e$ *and* $e \, (R \cap <)^* \, c$.

If (1) $a \, (R \cap <)^* \, c$ *then* $a \, R^* \, b$ *(L 2.6)*

If (2)

… ***enter***(*e*) … ***enter***(*c*) … ***exit***(*c*) … ***exit***(*e*) …

$\therefore$ ***enter***(*b*) *must be after* ***enter***(*a*).

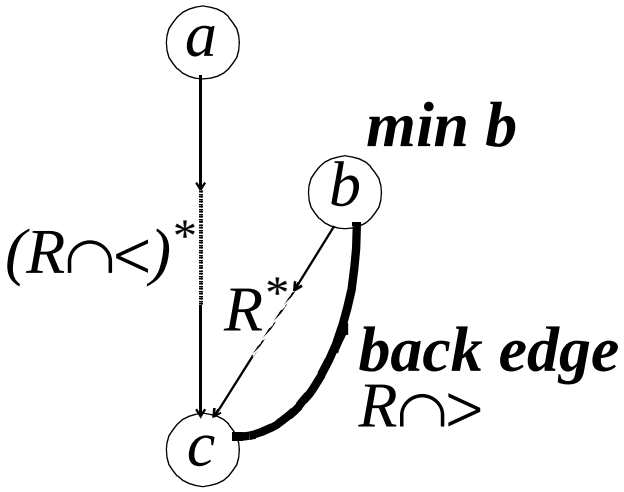
$\therefore$ … ***enter***(*e*) … ***enter***(*b*) … ***exit***(*b*) … ***exit***(*e*) …

$\therefore \; e \, (R \cap <)^* \, b$.

$\therefore \; e \, R^* \, b$, *and* $a \, R^* \, b$.

***minimum elements** of Scc's*
with respect to a depth-first order ≤.

$f(a) = min(\{a\}, \{b \mid a\ (R \cap <)^*\ c, b\ R^* c, c\ R\ b, b<c\})$
f(a) either a or
minimum b,
*a **back edge** entering b and leaving c, b<c*



f(a) < a, iff
a $(R \cap <)^*$ *c R b* R^**c, b<c, and b<a.*
≡ $a \in Scc(b)$, *b<a.*

***Lemma 2.12** f(a) = a, iff a = min Scc(a).*

Lemma 2.13

$$g(a) = min(\{a\} \cup \{b \mid a\ R\ b, b\ R^* a, b < a\} \cup \{g(b) \mid a\ R\ b, a < b\}).$$

The g = f.

Lemma 2.14 *If a = min Scc(a),*
$a\ (R \cap <)^* b, \forall b \in Scc(a).$

```
procedure FinScc(node a);
begin
    mark a;
    f̂(a) := a; push a onto Stack;
    for each edge (a, b) ∈ R do
        if b is unmaked then
            FinScc(b);
            f̂(a) := min{f̂(a), f̂(b)}
        else if b is on Stack then
            f̂(a) := min{b, f̂(a)}
            fi
    od  if;
    if f̂(a) = a then
        repeat
            pop b off Stack
        until a = b
    fi
end.
```