Corso di Laurea in INFORMATICA

Algoritmi e Strutture Dati

MODULO 8

STRUTTURE NON LINEARI

Alberi n-ari: specifiche sintattiche e semantiche. Realizzazioni. Visita di alberi n-ari.

ALBERO N-ARIO

DEFINIZIONE:

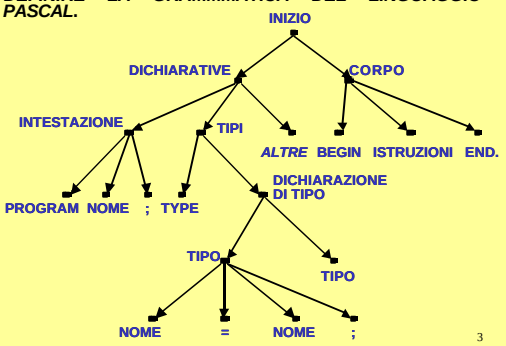
UN ALBERO È UN *GRAFO ORIENTATO* CHE O È VUOTO OPPURE HA LE SEGUENTI CARATTERISTICHE:

- ESISTE UN NODO R, DETTO RADICE, SENZA PREDECESSORI, CON n (n³0) NODI SUCCESSORI $a_1, a_2, \ldots, a_n$;
- TUTTI GLI ALTRI NODI SONO RIPARTITI IN n SOTTOALBERI MUTUAMENTE DISGIUNTI $T_1, T_2, \ldots, T_n$ AVENTI RISPETTIVAMENTE $a_1, a_2, \ldots, a_n$ COME RADICE.

L'ALBERO N-ARIO È UN TIPO ASTRATTO DI DATI UTILIZZATO PER RAPPRESENTARE RELAZIONI GERARCHICHE TRA OGGETTI. NELLA DEFINIZIONE DATA ABBIAMO ASSUNTO CHE SUI FIGLI DI OGNI NODO SIA DEFINITA UNA RELAZIONE D'ORDINE (*ALBERI ORDINATI*).

2

ESEMPIO: APPLICAZIONE DEGLI ALBERI PER DEFINIRE LA GRAMMMATICA DEL LINGUAGGIO *PASCAL*.



3

SPECIFICA SINTATTICA

Tipi: *albero, boolean, nodo*

Operatori:

CREAALBERO:	( ) ® *albero*
ALBEROVUOTO:	(*albero*) ® *boolean*
INSRADICE:	(*nodo, albero*) ® *albero*
RADICE:	(*albero*) ® *nodo*
PADRE:	(*nodo, albero*) ® *nodo*
FOGLIA:	(*nodo, albero*) ® *boolean*
PRIMOFIGLIO:	(*nodo, albero*) ® *nodo*
ULTIMOFRATELLO:	(*nodo, albero*) ® *boolean*
SUCCFRATELLO:	(*nodo, albero*) ® *nodo*
INSPRIMOSOTTOALBERO:	(*nodo,albero,albero*)®*albero*
INSSOTTOALBERO:	(*nodo, albero, albero*) ® *albero*
CANCSOTTOALBERO:	(*nodo, albero*) ® *albero*

4

SPECIFICA SEMANTICA

Tipi:
albero=insieme degli alberi ordinati T=<N,A> in cui ad ogni nodo n in N è associato il livello(n);
boolean=insieme dei valori di verità;
nodo=insieme qualsiasi (non finito).

Operatori:

CREAALBERO=T'
POST: T'=Λ (ALBERO VUOTO)

ALBEROVUOTO(T)=b
POST: b=VERO SE T=Λ
b=FALSO, ALTRIMENTI

INSRADICE(u, T) =T'
PRE: T=Λ
POST: T'=(N,A), N={u}, LIVELLO(u)=0, A=Ø

RADICE(T) =u
PRE: T¹Λ
POST: u È RADICE DI T È LIVELLO(u)=0

PADRE(u, T) =v
PRE: T¹Λ, uÎN, LIVELLO(u)>0
POST: v È PADRE DI u, <v,u>Î A
LIVELLO(u)=LIVELLO(v)+1

5

FOGLIA(u, T) =b
PRE: T¹Λ, uÎN
POST: b=VERO SE ØÙ vÎN '' <u,v>Î A Ù
Ù LIVELLO(v)=LIVELLO(u)+1
b=FALSO, ALTRIMENTI

PRIMOFIGLIO(u, T) =v
PRE: T¹Λ, uÎN, FOGLIA(u, T)=FALSO
POST: <u,v>Î A, LIVELLO(v)=LIVELLO(u)+1
v È PRIMO SECONDO LA RELAZIONE D'ORDINE STABILITA TRA I FIGLI DI u

ULTIMOFRATELLO(u, T) =b
PRE: T¹Λ, uÎN
POST: b=VERO SE NON ESISTONO ALTRI FRATELLI DI u CHE LO SEGUONO NELLA RELAZIONE D'ORDINE
b=FALSO, ALTRIMENTI

SUCCFRATELLO(u, T) =v
PRE: T¹Λ, uÎN, ULTIMOFRATELLO(u, T)=FALSO
POST: v È IL FRATELLO DI u CHE LO SEGUE NELLA RELAZIONE D'ORDINE

6

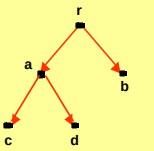
INSPRIMOSOTTOALBERO(u, T, T') =T''
PRE: T ∈ L, T' ∈ L, u ∈ N
POST: T'' È OTTENUTO DA T AGGIUNGENDO L'ALBERO T' LA CUI RADICE r' È IL NUOVO PRIMOFIGLIO DI u

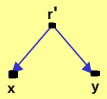
INSSOTTOALBERO(u, T, T') =T''
PRE: T ∈ L, T' ∈ L, u ∈ N, u NON È RADICE DI T
POST: T'' È L'ALBERO OTTENUTO DA T AGGIUNGENDO IL SOTTOALBERO T' DI RADICE r' (CIOÈ r' DIVENTA IL NUOVO FRATELLO CHE SEGUE u NELLA RELAZIONE D'ORDINE)

CANCSOTTOALBERO(u, T) =T'
PRE: T ∈ L, u ∈ N
POST: T' È OTTENUTO DA T TOGLIENDOVI IL SOTTOALBERO DI RADICE u (CIOÈ u E TUTTI I SUOI DISCENDENTI)

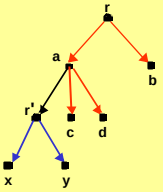
7

ESEMPI DI INSERIMENTI

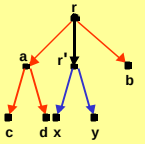




INSPRIMOSOTTOALBERO(a,T,T')



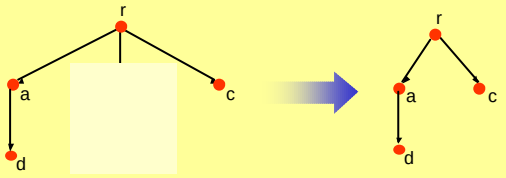
INSSOTTOALBERO(a,T,T')



8

ESEMPIO DI CANCELLAZIONE

CANCSOTTOALBERO(b,T)



9

VISITA DI ALBERI

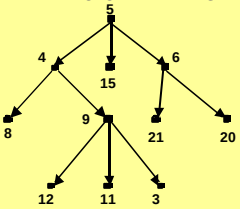
CONSISTE NEL PIANIFICARE E SEGUIRE UNA "ROTTA" CHE CONSENTA DI ESAMINARE OGNI NODO DELL'ALBERO ESATTAMENTE UNA VOLTA.
ESISTONO MODI DIVERSI PER EFFETTUARE UNA VISITA CORRISPONDENTE ALL'*ORDINE* CON CUI SI INTENDE SEGUIRE LA STRUTTURA.

SIA UN ALBERO NON VUOTO T DI RADICE r. SE r NON È FOGLIA ED HA k (k>0) FIGLI, SIANO T_1, T_2, …, T_k I SOTTOALBERI DI T AVENTI COME RADICI I FIGLI DI r. GLI ORDINI DI VISITA SONO:

- ***PREVISITA*** *(PREORDINE)*: CONSISTE NELL'ESAMINARE r E POI, NELL'ORDINE, EFFETTUARE LA PREVISITA DI T_1, T_2, …, T_k;
- ***POSTVISITA*** *(POSTORDINE)*: CONSISTE NEL FARE, NELL'ORDINE, PRIMA LA POSTVISITA DI T_1, T_2, …, T_k E POI NELL'ESAMINARE LA RADICE r ;
- ***INVISITA*** *(ORD. SIMMETRICO)*: CONSISTE NEL FARE, NELL'ORDINE LA INVISITA DI T_1, T_2, …, T_k, NELL'ESAMINARE r, E POI EFFETTUARE, NELL'ORDINE, LA INVISITA DI T_{i+1}, …, T_k, PER UN PREFISSATO i ≥ 1.

10

ESEMPIO: SIA UN ALBERO CHE HA DEGLI INTERI NEI NODI:



LA VISITA IN PREORDINE HA L'EFFETTO DI VISITARE I NODI SECONDO LA SEQUENZA:
5 4 8 9 12 11 3 15 6 21 20
LA VISITA IN POSTORDINE PRODUCE:
8 12 11 3 9 4 15 21 20 6 5
LA INVISITA (i=1) PRODUCE:
8 4 12 9 11 3 5 15 21 6 20

11

DIAMO UNA REALIZZAZIONE IN **PASCAL**

```
procedure PREVISITA(var T:albero; U:nodo);
var C:nodo;
begin
        {esamina nodo U};              (1)
        if not FOGLIA(U,T) then        (2)
                begin
                C:=PRIMOFIGLIO(U,T);
                while not ULTIMOFRATELLO(C,T) do
                        begin
                        PREVISITA(T,C);
                        C:=SUCCFRATELLO(C,T);
                        end;
                PREVISITA(T,C);
                end;
end;
```

SCAMBIANDO L'ORDINE DELLE ISTRUZIONI (1) E (2) SI OTTIENE LA POSTVISITA.

12

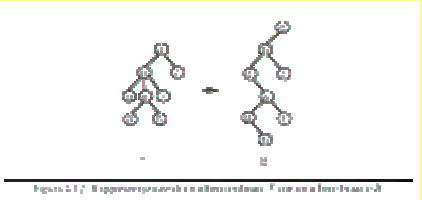
DIAMO ORA LA **INVISITA** (PER i=1):

```
procedure INVISITA(var T:albero; U:nodo);
var C:nodo;
begin
        if FOGLIA(U,T) then
                {esamina nodo U};
                else
                begin
                C:=PRIMOFIGLIO(U,T);
                INVISITA(T,C);
                {esamina nodo U};
                while not ULTIMOFRATELLO(C,T) do
                        begin
                        C:=SUCCFRATELLO(C,T);
                        INVISITA(T,C);
                        end;
                end;
end;
```

13

EQUIVALENZA DI ALBERI N-ARI E BINARI

È EVIDENTE CHE SI TRATTA DI UNA *EQUIVALENZA AI FINI DELLA PRE-VISITA*. E' SEMPRE POSSIBILE RAPPRESENTARE UN ALBERO N-ARIO ORDINATO T CON UN ALBERO BINARIO B AVENTE GLI STESSI NODI E LA STESSA RADICE: IN B OGNI NODO HA COME FIGLIO SINISTRO IL PRIMO FIGLIO IN T E COME FIGLIO DESTRO IL FRATELLO SUCCESSIVO IN T.



È FACILE NOTARE CHE LE SEQUENZE DI NODI ESAMINATI SU T E SU B COINCIDONO SE T E B SONO VISITATI IN PREVISITA.

14

OLTRE ALLE VISITE, UN'ALTRA FUNZIONE UTILE E' LA DI **PROFONDITA' DI UN ALBERO** INTESA COME IL MASSIMO LIVELLO DELLE FOGLIE.

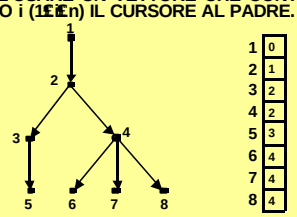
```
function MAXPROFONDITA(U:nodo; var T:albero):integer;
var     V:nodo;
        MAX, CORR:integer;
begin
        if FOGLIA(U,T) then
                MAXPROFONDITA:=0
                else
                begin
                V:=PRIMOFIGLIO(U,T);  MAX:=0;
                while not ULTIMOFRATELLO(V,T) do
                        begin
                        CORR:=MAXPROFONDITA(U,T);
                        if MAX<CORR then MAX:=CORR;
                        V:=SUCCFRATELLO(V,T);
                        end;
                 MAXPROFONDITA:=MAX+1;
                end;
end;
```

VA CHIAMATA COME MAXPROFONDITA(RADICE(T),T).

15

RAPPRESENTAZIONE CON VETTORE DI PADRI

IMMAGINANDO DI NUMERARE I NODI DI T DA 1 A n, LA PIU SEMPLICE REALIZZAZIONE (SEQUENZIALE) CONSISTE NELL'USARE UN VETTORE CHE CONTIENE, PER OGNI NODO i (1≤i≤n) IL CURSORE AL PADRE.



È FACILE, COSÌ, VISITARE I NODI LUNGO PERCORSI CHE VANNO DA FOGLIE A RADICE. È, INVECE, PIÙ COMPLESSO INSERIRE E CANCELLARE SOTTOALBERI.

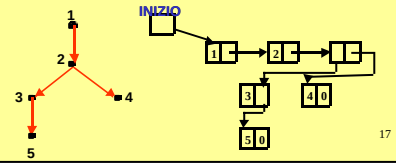
16

LA RAPPRESENTAZIONE MEDIANTE LISTA

UN ALBERO PUÒ ESSERE RAPPRESENTATO MEDIANTE LISTA SECONDO LE SEGUENTI REGOLE:

- SE L'ALBERO È VUOTO LA LISTA CHE LO RAPPRESENTA È VUOTA;
- ALTRIMENTI, L'ALBERO È COMPOSTO DA UNA RADICE E DA k SOTTOALBERI T_1, T_2, …, T_k E LA LISTA È FATTA DA k+1 ELEMENTI: IL PRIMO RAPPRESENTA LA RADICE, MENTRE GLI ALTRI SONO GLI ALBERI T_1, T_2, …, T_k (CON k≥0);

USANDO UNA RAPPRESENTAZIONE COLLEGATA L'ALBERO CORRISPONDE A



17

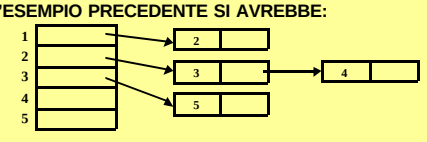
UNA VARIANTE MOLTO USATA, POICHÉ NON SI PUÒ ASSOCIARE AD OGNI ELEMENTO UN NUMERO DI PUNTATORI UGUALE AL MASSIMO DEI FIGLI, E'

LA RAPPRESENTAZIONE ATTRAVERSO LISTE DI FIGLI

COMPRENDE:

- IL VETTORE DEI NODI, IN CUI, OLTRE ALLE EVENTUALI ETICHETTE DEI NODI, SI MEMORIZZA IL RIFERIMENTO INIZIALE DI UNA LISTA ASSOCIATA AD OGNI NODO;
- UNA LISTA PER OGNI NODO, DETTA *LISTA DEI FIGLI*. LA LISTA ASSOCIATA AL GENERICO NODO i CONTIENE TANTI ELEMENTI QUANTI SONO I SUCCESSORI DI i; CIASCUN ELEMENTO È IL RIFERIMENTO AD UNO DEI SUCCESSORI;

PER L'ESEMPIO PRECEDENTE SI AVREBBE:



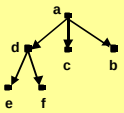
18

RAPPRESENTAZIONE MEDIANTE LISTA PRIMOFIGLIO/FRATELLO

PREVEDE LA GESTIONE DI UNA LISTA E QUESTO PUÒ ESSERE FATTO IMPONENDO CHE TUTTI GLI ALBERI *CONDIVIDANO UN'AREA COMUNE* (AD ESEMPIO UN VETTORE NELLA REALIZZAZIONE CON CURSORI) E CHE *OGNI CELLA CONTENGA ESATTAMENTE DUE CURSORI: UNO AL PRIMOFIGLIO ED UNO AL FRATELLO SUCCESSIVO*. LA REALIZZAZIONE È SIMILE A QUELLA PROPOSTA PER GLI ALBERI BINARI CON L'UNICA DIFFERENZA CHE IL CURSORE NEL TERZO CAMPO PUNTA AL FRATELLO.

INIZIO: 4

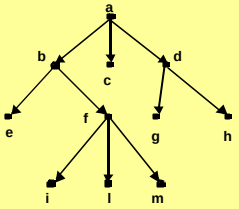
	FIGLIO	NODO	FRATELLO
1	0	e	2
2	0	f	0
3	0	c	5
4	7	a	0
5	0	b	0
6			
7	1	d	3
8			



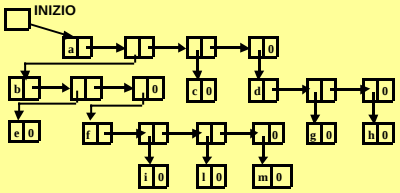
19

LA RAPPRESENTAZIONE CON LISTA DINAMICA

LA RAPPRESENTAZIONE CON LISTA DINAMICA È PIÙ COMPLESSA: LA RADICE DELL'ALBERO VIENE MEMORIZZATA NEL PRIMO ELEMENTO DELLA LISTA CHE CONTIENE IL RIFERIMENTO AD UNA LISTA DI ELEMENTI, UNO PER OGNI SOTTOALBERO. CIASCUNO DI QUESTI ELEMENTI CONTIENE, A SUA VOLTA, IL RIFERIMENTO INIZIALE ALLA LISTA CHE RAPPRESENTA IL CORRISPONDENTE SOTTOALBERO.



20



UNA POSSIBILE REALIZZAZIONE PREVEDE RECORD E PUNTATORI: PER ESEMPIO UN RECORD CON TRE CAMPI, UNO PER LA PARTE INFORMAZIONE E DUE PER I PUNTATORI. PER OGNI RECORD SARÀ SEMPRE SIGNIFICATIVO UNO DEI CAMPI PUNTATORI, MENTRE O IL CAMPO INFORMAZIONE O L'ALTRO CAMPO NON VERRÀ UTILIZZATO.

21

REALIZZAZIONE CON VETTORE DI PADRI/1

DIAMO UNA POSSIBILE DICHIARATIVA *PASCAL*

```
const
        MAXLUNG=50;
type
        nodo=integer;
        albero=array[1..MAXLUNG] of nodo;
```

E LA REALIZZAZIONE DI ALCUNI OPERATORI

```
function ALBEROVUOTO(var T:albero):boolean;
var     I:nodo;
        VUOTO:boolean;
begin
        I:=1;
        VUOTO:=true;
        while ((VUOTO) and (i £ MAXLUNG)) do
                VUOTO:=(T(I)=999);
        ALBEROVUOTO:=VUOTO;
end;
```

22

REALIZZAZIONE CON VETTORE DI PADRI/2

```
function PRIMOFIGLIO(U:nodo; var T:albero):nodo
var     I:nodo;
        TROVATO: boolean;
begin
        TROVATO:=false;
        I:=1;
        while (not TROVATO) and (i £ MAXLUNG) do
                if T[I]=U then TROVATO:=true
                          else I:=I+1;
        PRIMOFIGLIO:=I
end;
-----------------------------------------------------------------------------
function SUCCFRATELLO(U:nodo; var T:albero):nodo;
var     I,J:nodo;
        TROVATO:boolean;
begin
        J:=T[U];
        I:=U+1;
        TROVATO:=false;
        while (not TROVATO) and (i £ MAXLUNG) do
                if T[I]=J then
                        begin
                        SUCCFRATELLO:=I;
                        TROVATO:=true;
                        end
                        else
                        I:=I+1;
end;
```

23

LA REALIZZAZIONE MEDIANTE LISTE DI FIGLI/1

TRASCURANDO, PER IL MOMENTO, IL TIPO LISTA, UNA POSSIBILE DICHIARATIVA IN PASCAL PER L'ALBERO È':

```
type    nodo=integer;
        albero=record
                testa:array[1..MAXLUNG] of lista;
                radice:nodo;
                end;
```

DIAMO LA REALIZZAZIONE DI ALCUNI OPERATORI

```
function PRIMOFIGLIO(U:nodo; var T:albero):nodo;
var L:lista;
begin
        L:=T.TESTA[U];
        PRIMOFIGLIO:=LEGGILISTA(PRIMOLISTA(L),L);
end;
```

24

LA REALIZZAZIONE MEDIANTE LISTE DI FIGLI/2

```
function PADRE(U:nodo; var T:albero):nodo;
var I:nodo;
begin
        I:=1;
        while (not APPARTIENE(U, T.TESTA[U]) and (I<>N) do
                I:=I+1;
        PADRE:=I;
end;
```

QUESTA IMPONE L'USO DI UN OPERATORE DISPONIBILE PER LE LISTE

```
function APPARTIENE(U:nodo; var L:lista):boolean;
var     P:posizione;
        TROVATO:boolean;
begin
        P:=PRIMOLISTA(L);
        TROVATO:=false;
        while (not TROVATO) and (not FINELISTA(L)) do
                if LEGGILISTA(P,L)=U then
                        TROVATO:=true;
        APPARTIENE:=TROVATO;
end;
```


LA REALIZZAZIONE MEDIANTE LISTE DI FIGLI/3

È POSSIBILE IMPLEMENTARE LE LISTE UTILIZZANDO "*LISTA*" E "*POSIZIONE*" COME *CURSORI* IN UN ARRAY DI RECORD E T.TESTA[U] COME PUNTATORE ALLA CELLA CHE CONTIENE IL PRIMO ELEMENTO DELLA LISTA.

```
var CELLE: array[1..MAXLUNG] of record
                elemento: integer;
                successivo:integer
                end;
```

LA REALIZZAZIONE DI ALCUNI OPERATORI

```
function PRIMOFIGLIO(U:nodo; var T:albero):nodo;
var L:lista;
begin
        L:=T.TESTA[U];
        PRIMOFIGLIO:=CELLE[L].ELEMENTO;
end;
```


LA REALIZZAZIONE MEDIANTE LISTE DI FIGLI/4

```
function PADRE(U:nodo; var T:albero):nodo;
var     P:nodo;
        I:posizione;
        TROVATO:boolean;
begin
        P:=1;
        TROVATO:=false;
        while (not TROVATO) AND (P<=MAXLUNG) do
                begin
                I:=T.TESTA[P];
                while (not TROVATO) and (I<>0) do
                        if CELLE[I].ELEMENTO=U THEN
                                begin
                                PADRE:=P;
                                TROVATO:=true;
                                end
                                else
                                I:=CELLE[I].SUCCESSIVO;
                P:=P+1;
                end;
end;
```


LA REALIZZAZIONE MEDIANTE LISTA PRIMOFIGLIO/FRATELLO

UNA POSSIBILE DICHIARATIVA DI TIPO IN PASCAL DELLA IMPLEMENTAZIONE CON LISTA FIGLIO/FRATELLO O *AREA LIBERA*. PER ESEGUIRE EFFICIENTEMENTE ANCHE L'OPERATORE PADRE SI PUO IMPORRE CHE OGNI CELLA CONTENGA UN QUARTO CAMPO RISERVATO PER IL CURSORE AL PADRE.

```
type    tipoelem=char;
        indice=1..MAXLUNG;
        albero=indice;
        arealibera=array[indice] of elemento;
        elemento=record
                figlio:indice;
                nodo:tipoelem;
                fratello:indice;
                genitore:indice
                end;

var     SPAZIO:arealibera;
        T:albero;
```


LA REALIZZAZIONE CON LISTA DINAMICA

UNA POSSIBILE DICHIARATIVA IN PASCAL

```
type    punt=^nodo_alb;
        nodo_alb=record
                info:tipoelem;
                figlio, succ:punt
                end;
```

FIGLIO È IL CAMPO RISERVATO AL PUNTATORE AL SOTTOALBERO, SUCC È IL CAMPO PER IL PUNTATORE AL SUCCESSIVO ELEMENTO LISTA.

REALIZZAZIONE DI MFSET

COME E' NOTO UN MFSET È UNA PARTIZIONE DI UN INSIEME FINITO IN SOTTOINSIEMI DISGIUNTI DETTI COMPONENTI.

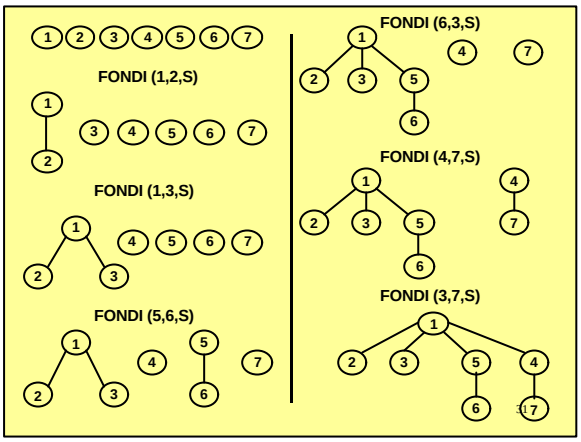
È POSSIBILE RAPPRESENTARLO MEDIANTE:

UNA FORESTA DI ALBERI RADICATI

IN CUI CIASCUN ALBERO RAPPRESENTA UNA COMPONENTE.
LE COMPONENTI INIZIALI DI MFSET SONO I NODI. ATTRAVERSO OPERAZIONI SUCCESSIVE DI FONDI E TROVA SI CREA LA STRUTTURA.

L'OPERATORE FONDI COMBINA DUE ALBERI NELLO STESSO ALBERO. SI REALIZZA IMPONENDO CHE UNA DELLE DUE RADICI DIVENTI NUOVO FIGLIO DELL'ALTRA.

L'OPERATORE TROVA VERIFICA SE DUE ELEMENTI SONO NEL MEDESIMO ALBERO. SI REALIZZA ACCEDENDO AI NODI CONTENENTI GLI ELEMENTI E RISALENDO DA TALI NODI, ATTRAVERSO I PADRI, FINO AD ARRIVARE ALLE RADICI.


1 2 3 4 5 6 7
FONDI (1,2,S)
FONDI (1,3,S)
FONDI (5,6,S)
FONDI (6,3,S)
FONDI (4,7,S)
FONDI (3,7,S)