



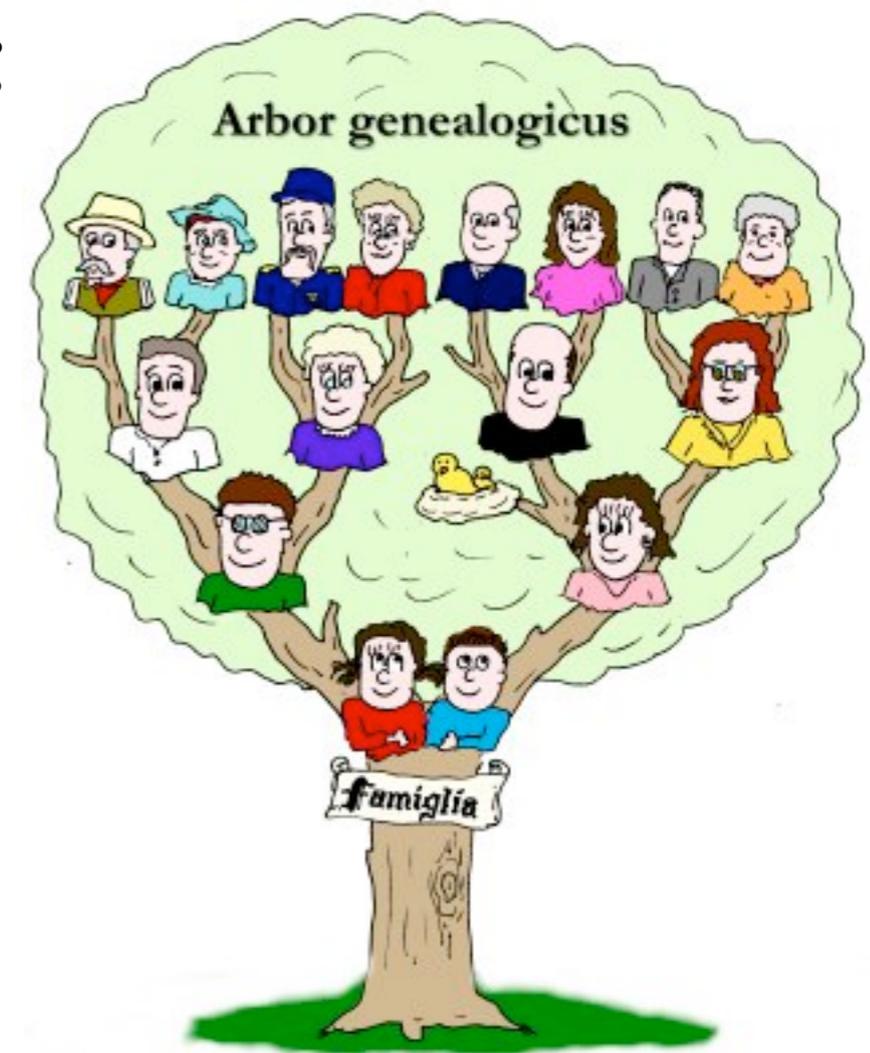
# IEIM

## Esercizio: L'albero Genealogico

Rispiaghiamolo Meglio :)

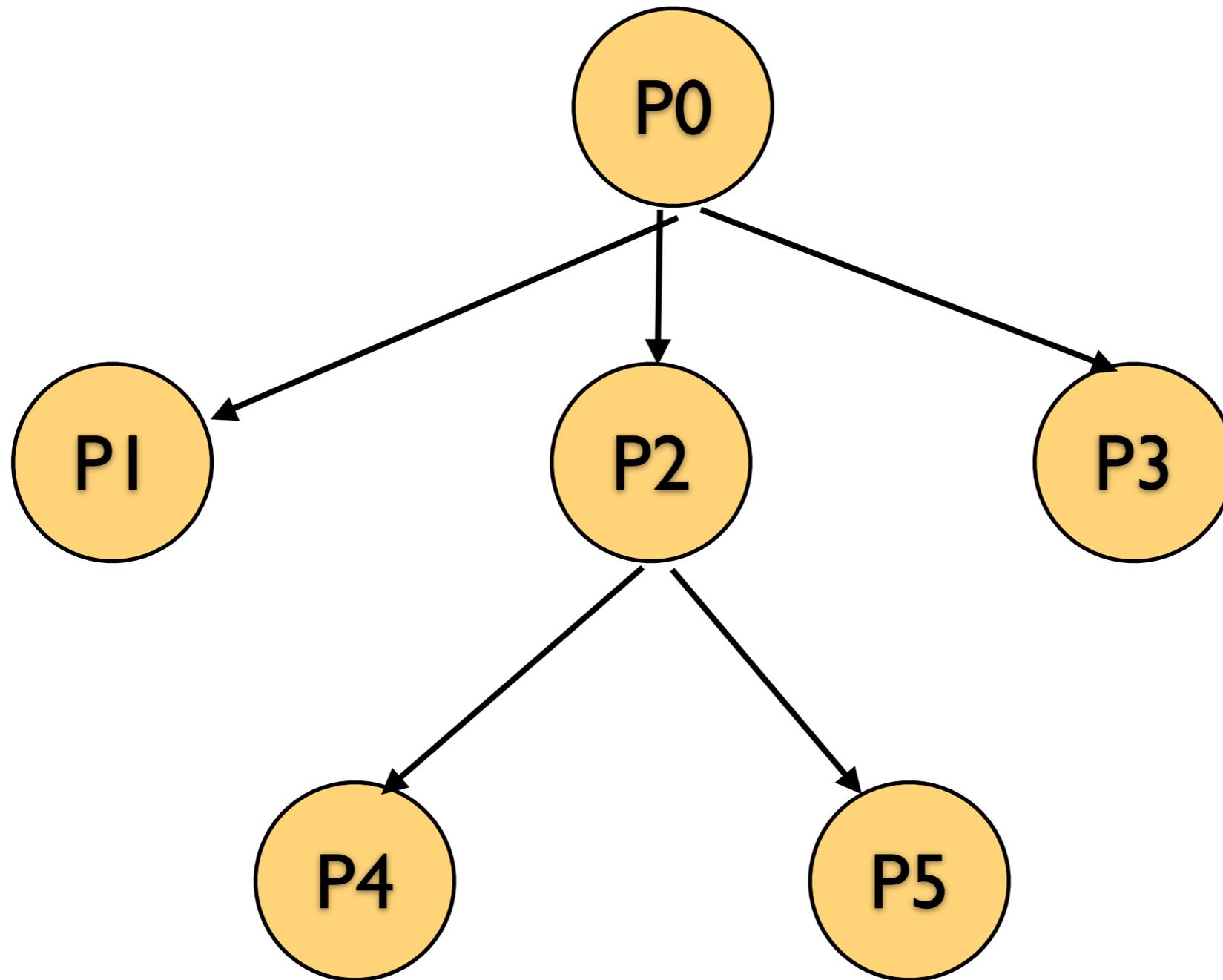
Alessandro A. Nacci  
nacci@elet.polimi.it - [alessandronacci.it](http://alessandronacci.it)

- Scrivere un programma C che sia in grado di rappresentare e gestire un albero genealogico
- In particolare, vogliamo poter fare:
  - Creare una persona
  - Rappresentare di una popolazione
  - Aggiungere figli ad una persona
  - Elencare i figli e i nipoti dato un antenato



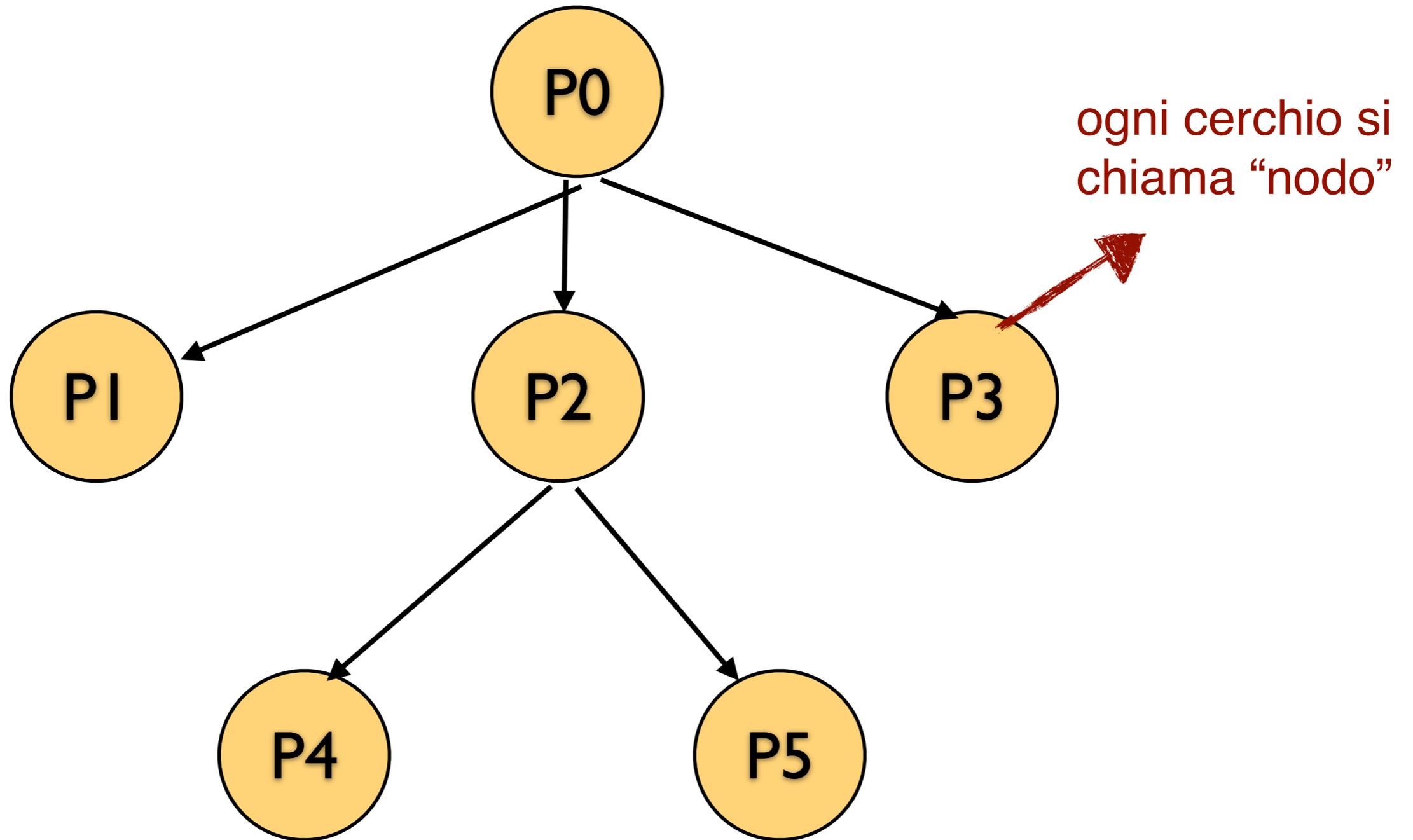


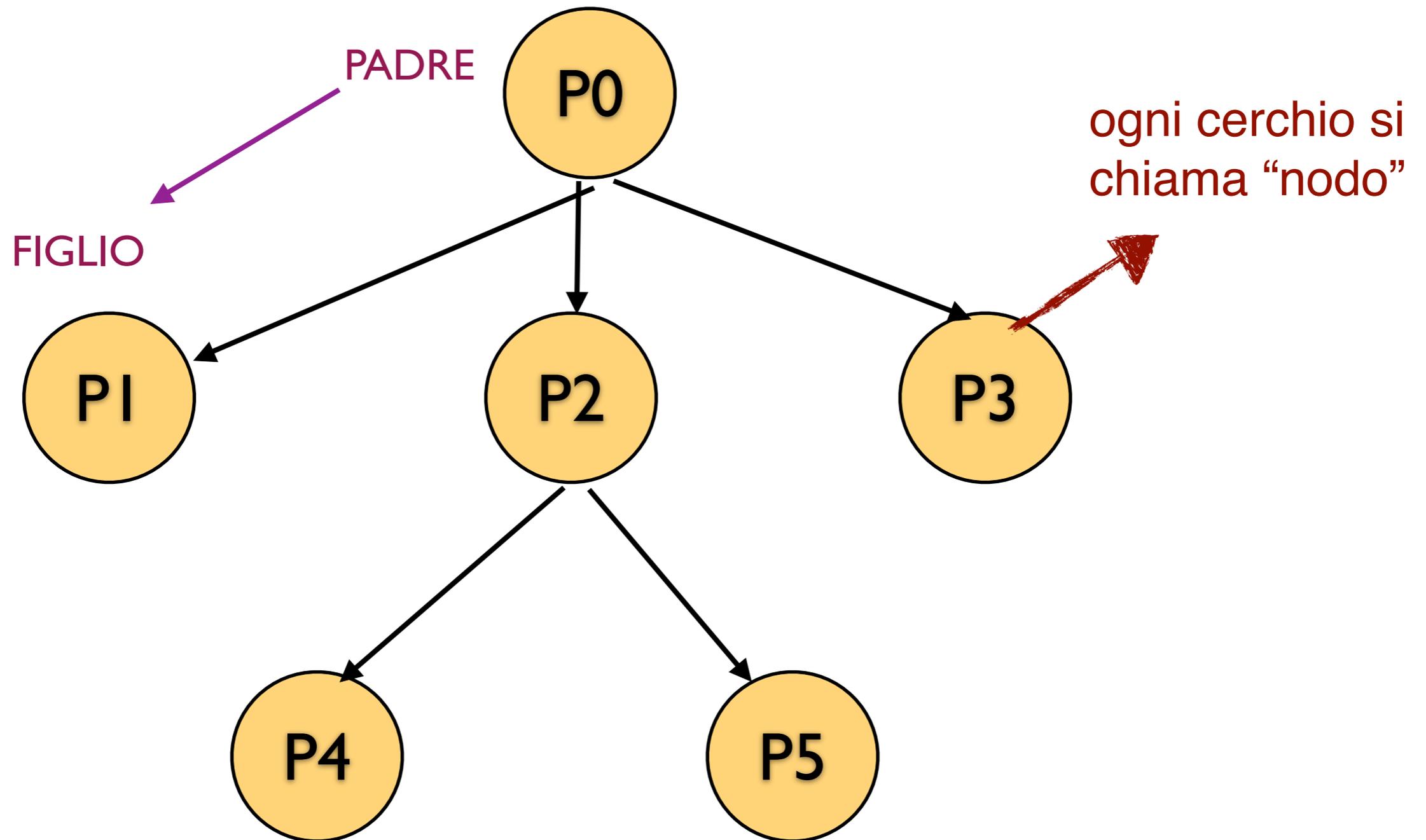
# Una famosa struttura dati: l'albero





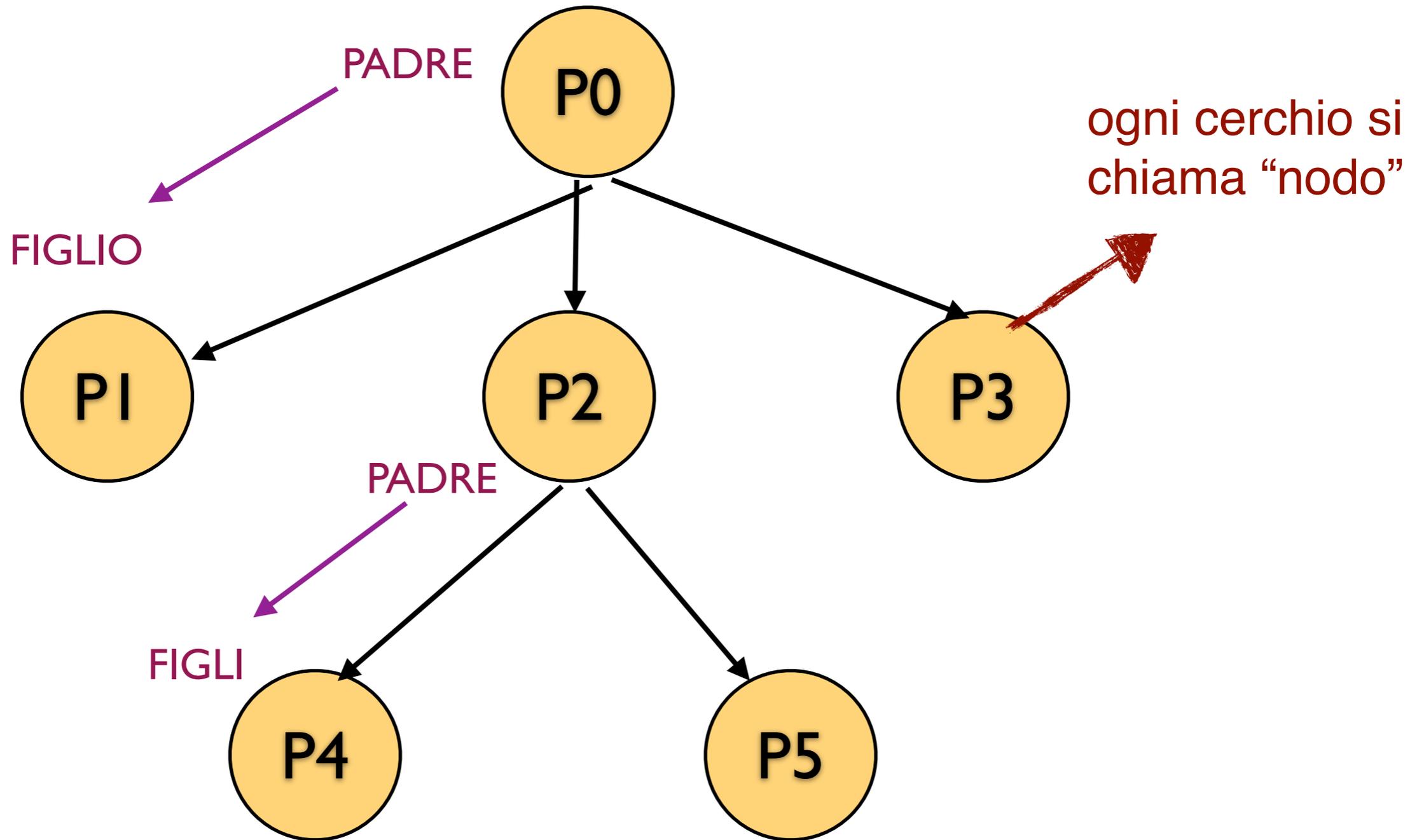
# Una famosa struttura dati: l'albero





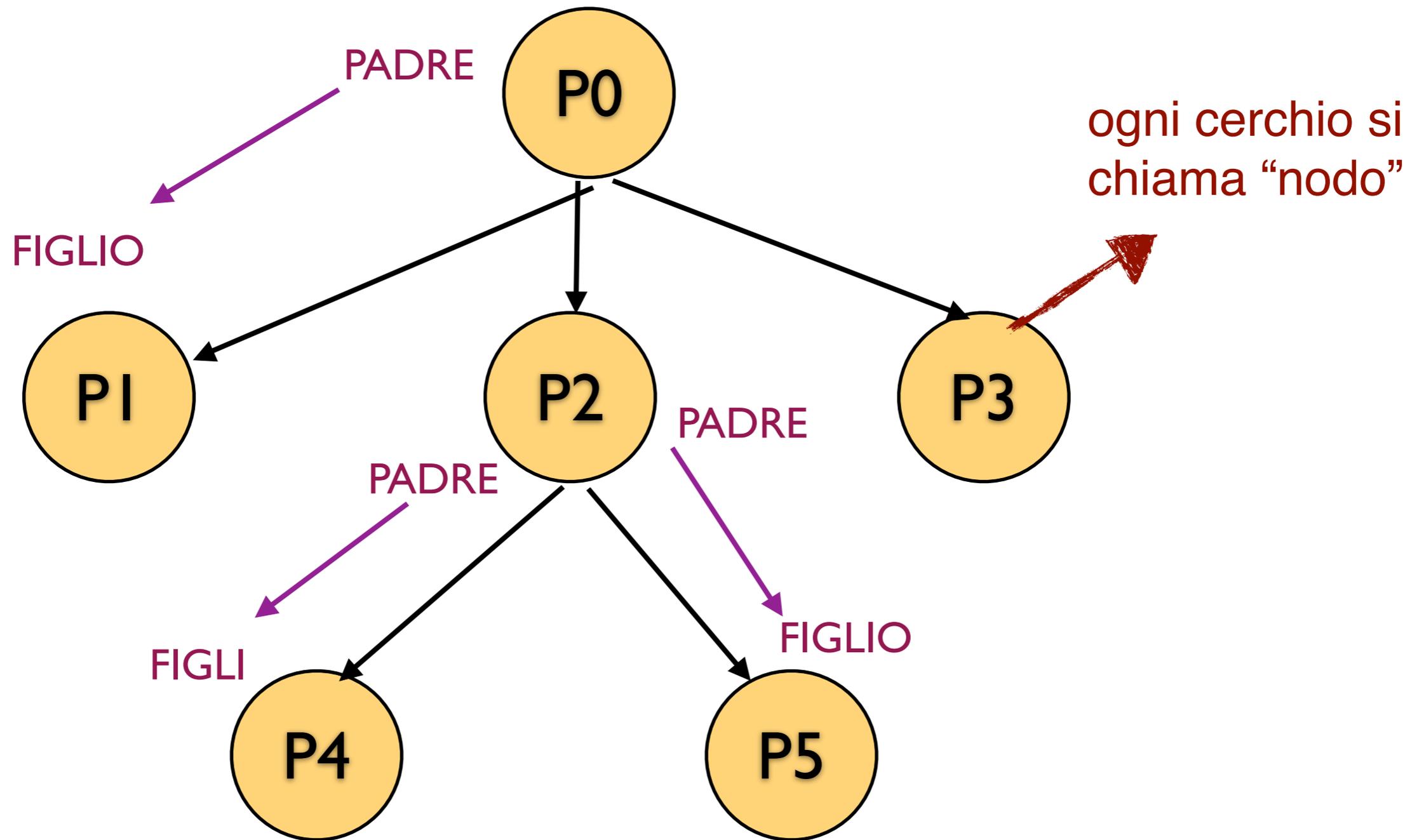


# Una famosa struttura dati: l'albero

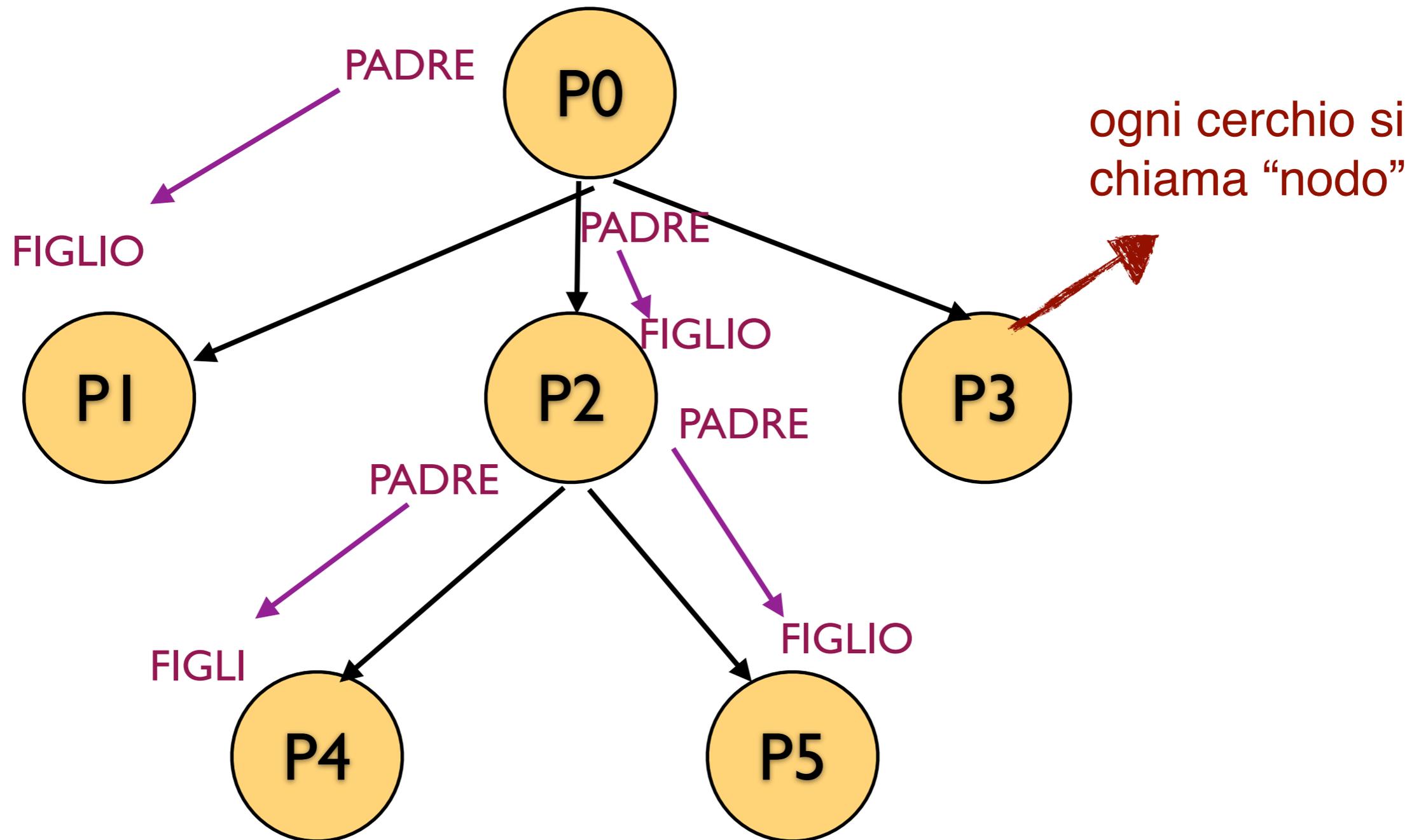




# Una famosa struttura dati: l'albero

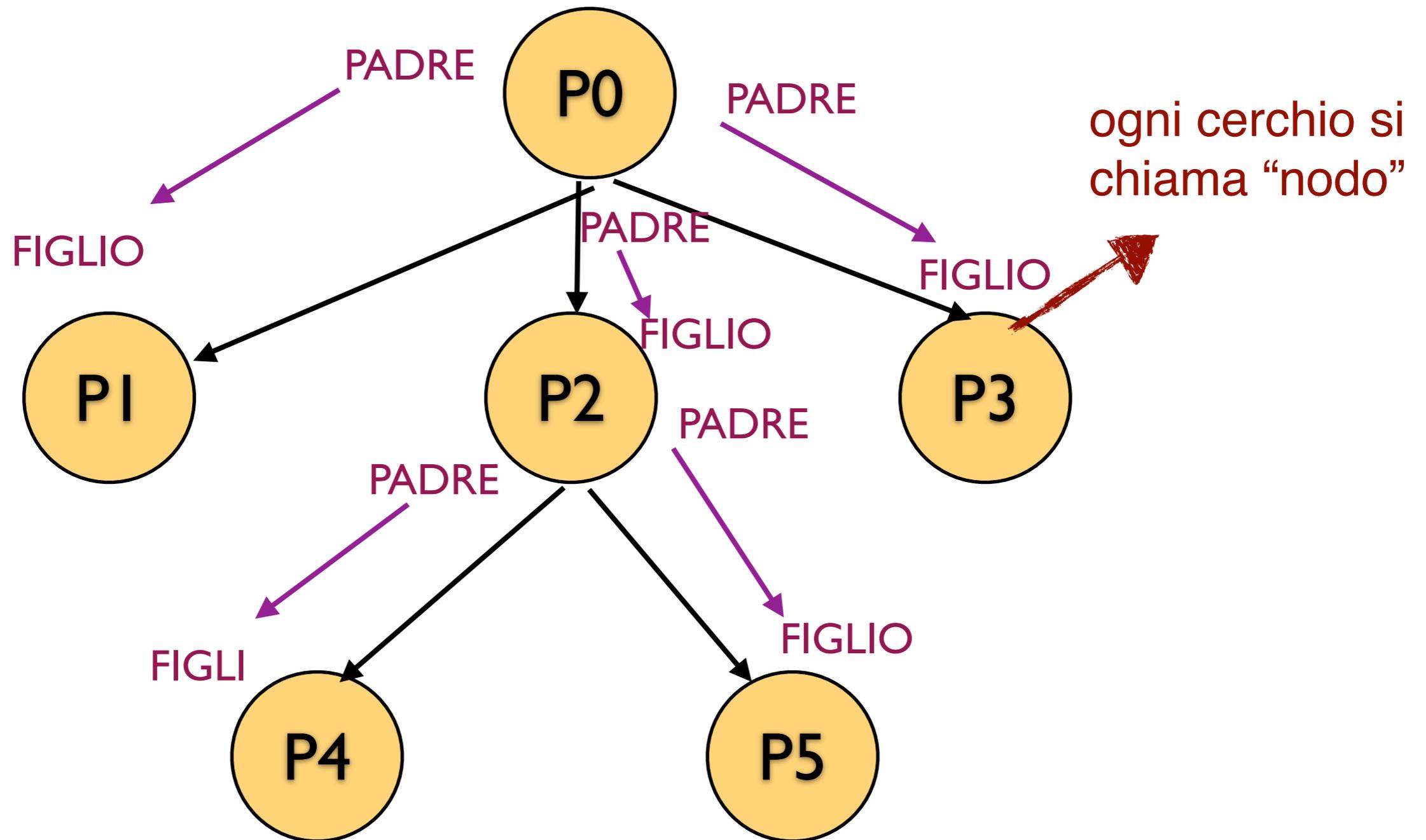


# Una famosa struttura dati: l'albero



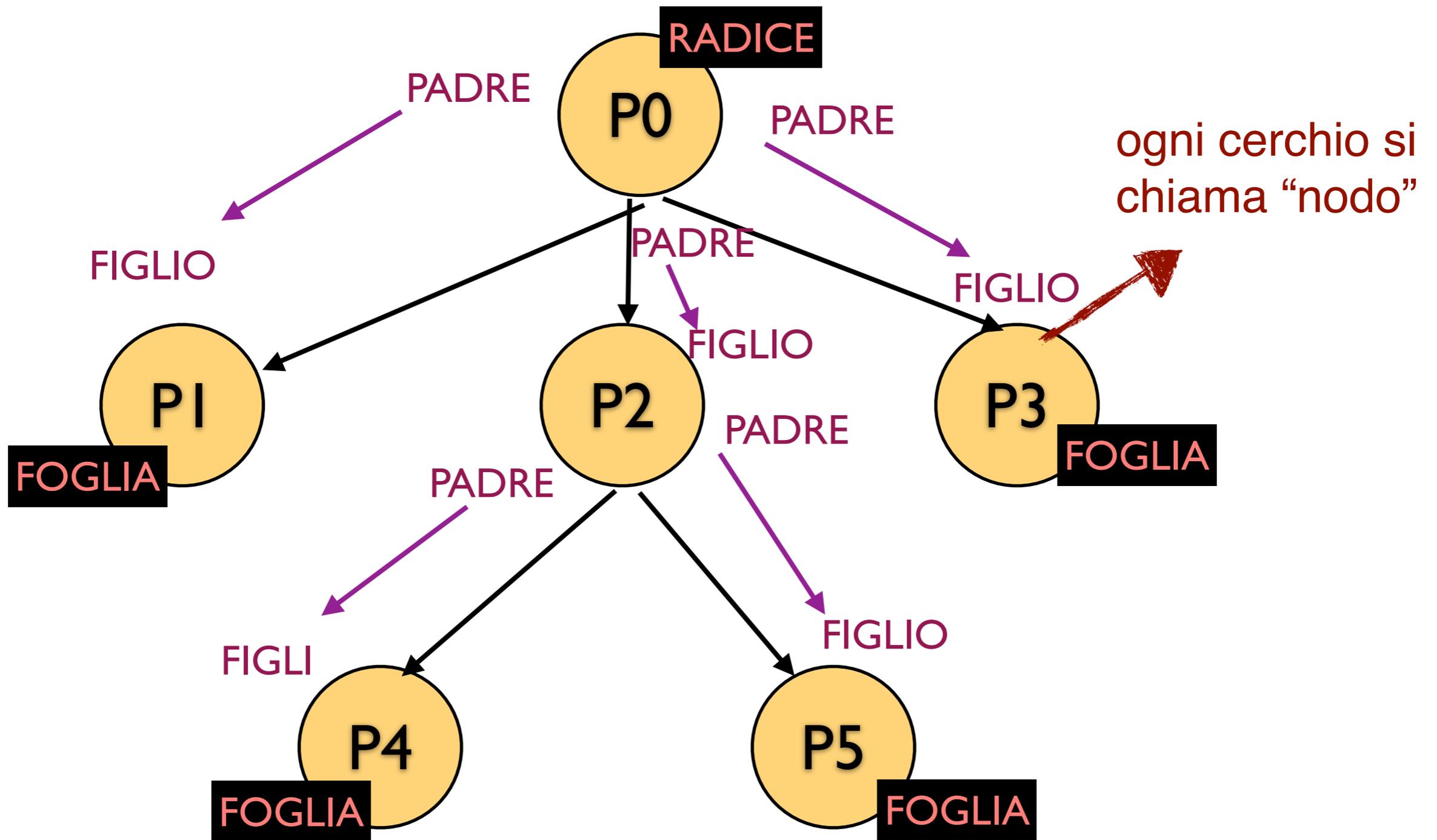


# Una famosa struttura dati: l'albero

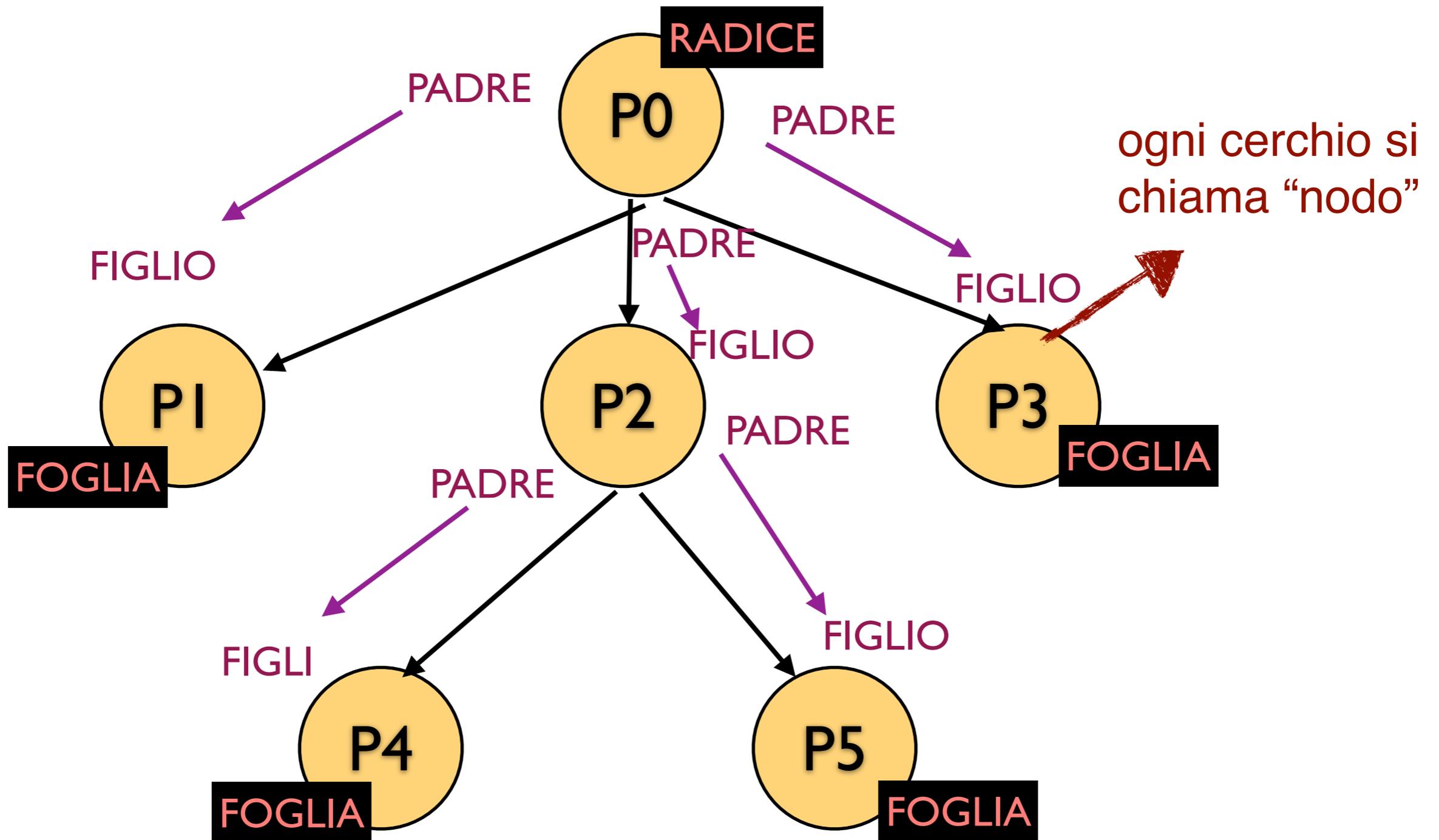




# Una famosa struttura dati: l'albero



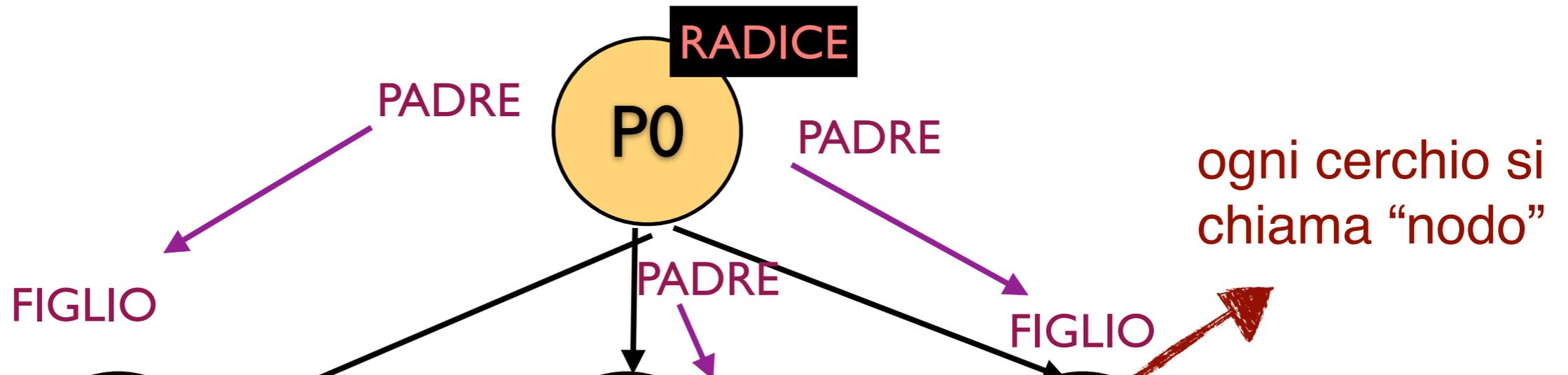
# Una famosa struttura dati: l'albero



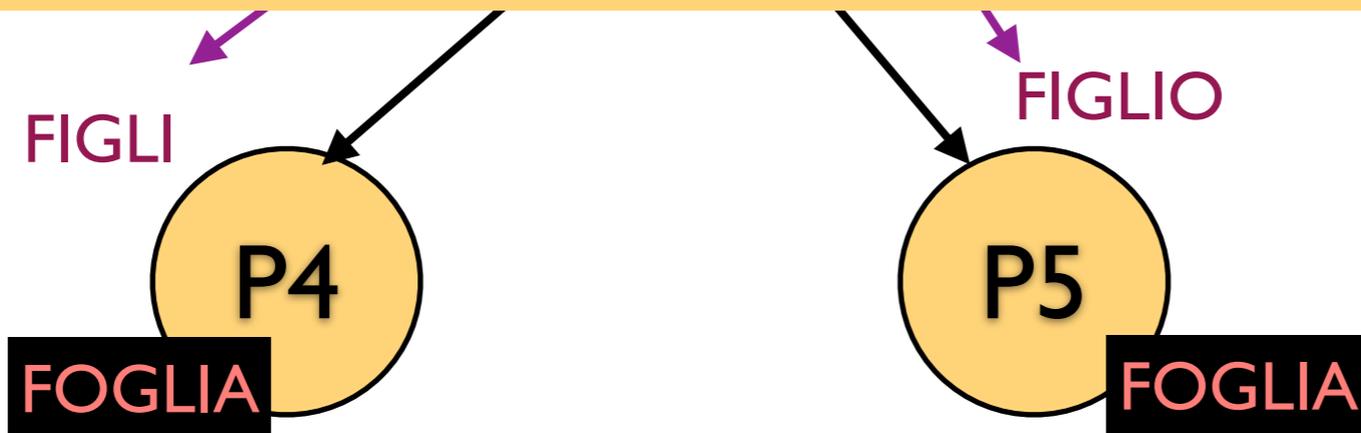
Può essere utile per rappresentare un albero genealogico?



# Una famosa struttura dati: l'albero



**OVVIAMENTE SI**

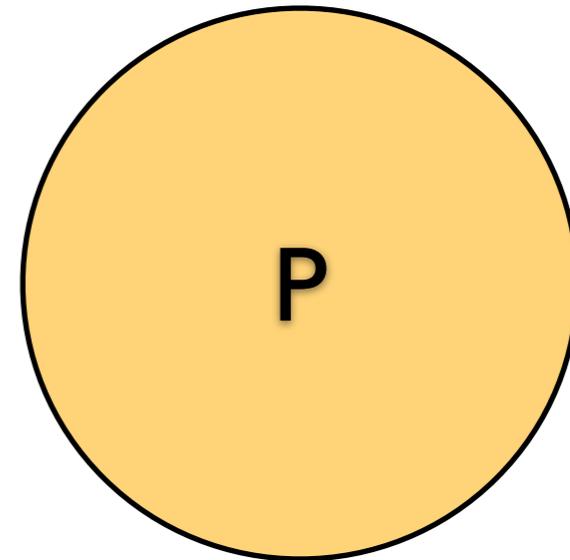


Può essere utile per rappresentare un albero genealogico?

- **OGNI NODO DELL'ALBERO SARA' PER NOI UNA PERSONA**



==



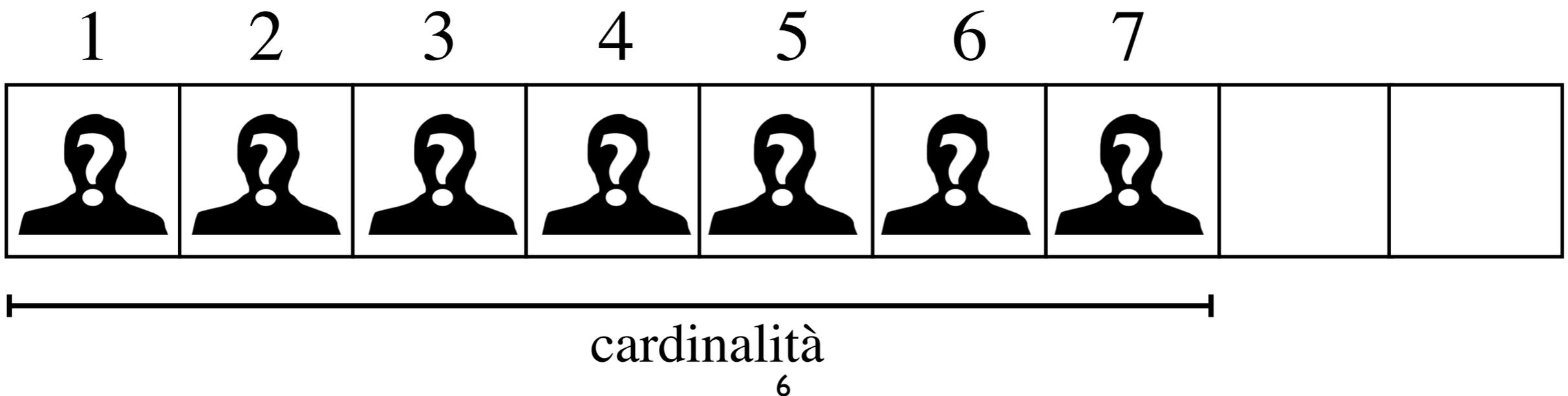


# Una Persona

- SESSO
- NOME
- ETA?
- CHI SONO I GENITORI?
- CHI SONO I FIGLI?
- QUANTI FIGLI?



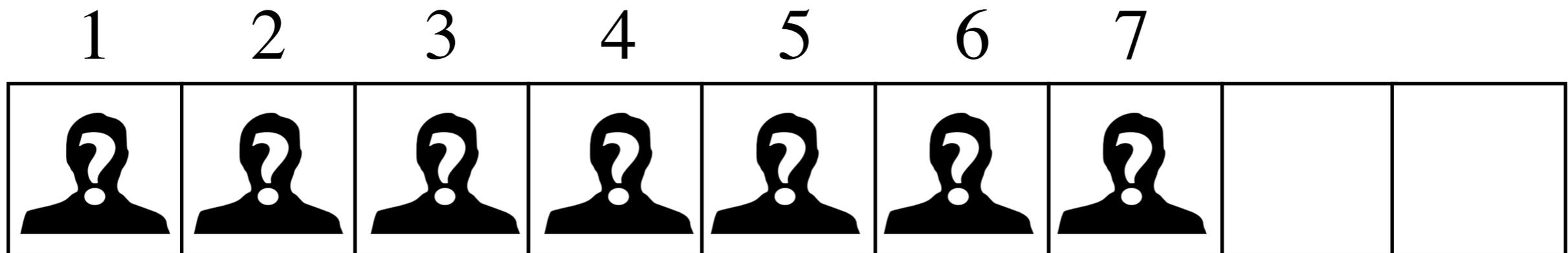
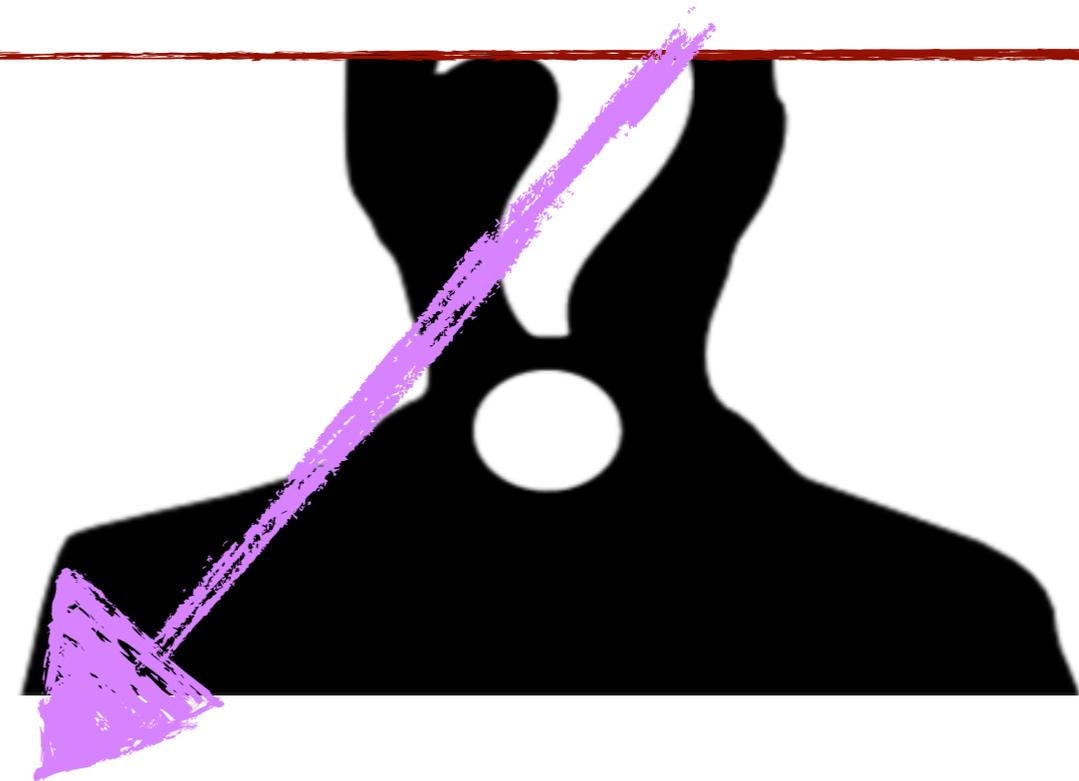
- Una popolazione è rappresentata da un insieme di persone
- Ogni persona ha un suo indice (numero univoco di identificazione)
- Esiste un numero di persone della



# Una Persona nella popolazione

- SESSO
- NOME
- ETA?
- CHI SONO I GENITORI?
- CHI SONO I FIGLI?
- QUANTI FIGLI?

Li rappresentiamo con l'indice della persona nella popolazione



cardinalità



# Persona e Popolazione (codice C)

```
typedef struct {
```

```
} persona;
```

```
int main () |  
{
```

```
    persona* popolazione[MAX_PERSONE];  
    int cardinalita_popolazione = -1;
```



# Persona e Popolazione (codice C)

```
typedef struct {
    genere sesso;
    char nome[STR_LEN];
    int i_persona;
    int i_genitore1;
    int i_genitore2;
    int i_figli[MAX_FIGLI];
    int numero_figli;
    int eta;
} persona;

int main () |
{

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;
```



# Persona e Popolazione (codice C)

```
typedef enum {MASCHIO, FEMMINA} genere;
```

```
typedef struct {  
    genere sesso;  
    char nome[STR_LEN];  
    int i_persona;  
    int i_genitore1;  
    int i_genitore2;  
    int i_figli[MAX_FIGLI];  
    int numero_figli;  
    int eta;  
} persona;
```

```
int main () |  
{
```

```
    persona* popolazione[MAX_PERSONE];  
    int cardinalita_popolazione = -1;
```



# Creazione di una persona

```
persona crea_persona(genere sesso, char nome[STR_LEN], int eta)  
{
```

```
}
```



# Creazione di una persona

```
persona crea_persona(genere sesso, char nome[STR_LEN], int eta)
{
    persona p;
    p.sesso = sesso;
    strcpy(p.nome, nome);
    p.i_genitore1 = -1;
    p.i_genitore2 = -1;
    p.numero_figli = 0;
    p.eta = eta;
    return p;
}
```





# Aggiunta persona alla popolazione

```
int aggiungi_a_popolazione(persona* popolazione[MAX_PERSONE],
                             persona* p, int* cardinalita_popolazione)
{
    (*cardinalita_popolazione)++;
    p->i_persona = *cardinalita_popolazione;
    popolazione[*cardinalita_popolazione] = p;
}
```



# Aggiunta di un figlio

```
void aggiungi_figlio(persona* genitore, persona* figlio)
{
    .....
}
}
```



# Aggiunta di un figlio

```
void aggiungi_figlio(persona* genitore, persona* figlio)
{
    genitore->i_figli[genitore->numero_figli] = figlio->i_persona;
    genitore->numero_figli++;

    if (figlio->i_genitore1 == -1)
        figlio->i_genitore1 = genitore->i_persona;
    else if (figlio->i_genitore2 == -1)
        figlio->i_genitore2 = genitore->i_persona;
    else
        printf("errore\n");
}
```



# Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)  
{
```

```
}
```

```
void stampa_persona(persona* p)  
{
```

```
}
```



# Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)
{
    if (sesso == MASCHIO) return "MASCHIO";
    if (sesso == FEMMINA) return "FEMMINA";
    return "?";
}
```

```
void stampa_persona(persona* p)
{
    // ...
}
}
```



# Funzioni di stampa a schermo

```
char* genere_to_str(genere sesso)
```

```
{  
    if (sesso == MASCHIO) return "MASCHIO";  
    if (sesso == FEMMINA) return "FEMMINA";  
    return "?";  
}
```

```
void stampa_persona(persona* p)
```

```
{  
    printf("Nome: %s - Genere: %s - Eta':%d - Id:%d - #Figli: %d\n",  
p->nome, genere_to_str(p->sesso), p->eta, p->i_persona, p->numero_figli);  
}
```



# Elenco dei figli e dei nipoti

```
void elenca_figli_nipoti(persona* popolazione[MAX_PERSONE],
    persona* p, genere sesso, int eta_minima)
{

}
}
```



# Elenco dei figli e dei nipoti

```
void elenca_figli_nipoti(persona* popolazione[MAX_PERSONE],
    persona* p, genere sesso, int eta_minima)
{

    int i;

    if (p->sesso == sesso && p->eta >= eta_minima)
        stampa_persona(p);

    for (i = 0; i < p->numero_figli; i++)
    {
        persona* f = popolazione[p->i_figli[i]];
        elenca_figli_nipoti(popolazione, f, sesso, eta_minima);
    }
}
```



# La nostra popolazione



MARCO

P0



STEFANIA

P1



LUCA

P2



PIPPO

P3



LUCIA

P4



ARIANNA

P5



RINALDO

P6



STEFANO

P7



# La nostra popolazione



MARCO

P0



STEFANIA

P1



LUCA

P2



PIPPO

P3



LUCIA

P4



ARIANNA

P5



RINALDO

P6



STEFANO

P7

Marco e' padre di LUCA e di PIPPO

Stefania e' madre di LUCA e di PIPPO

Arianna e' figlia di Marco e Lucia

Stefano e' figlio di Arianna e Rinaldo



# La nostra popolazione



MARCO

P0



STEFANIA

P1



LUCA

P2



PIPPO

P3



LUCIA

P4



ARIANNA

P5



RINALDO

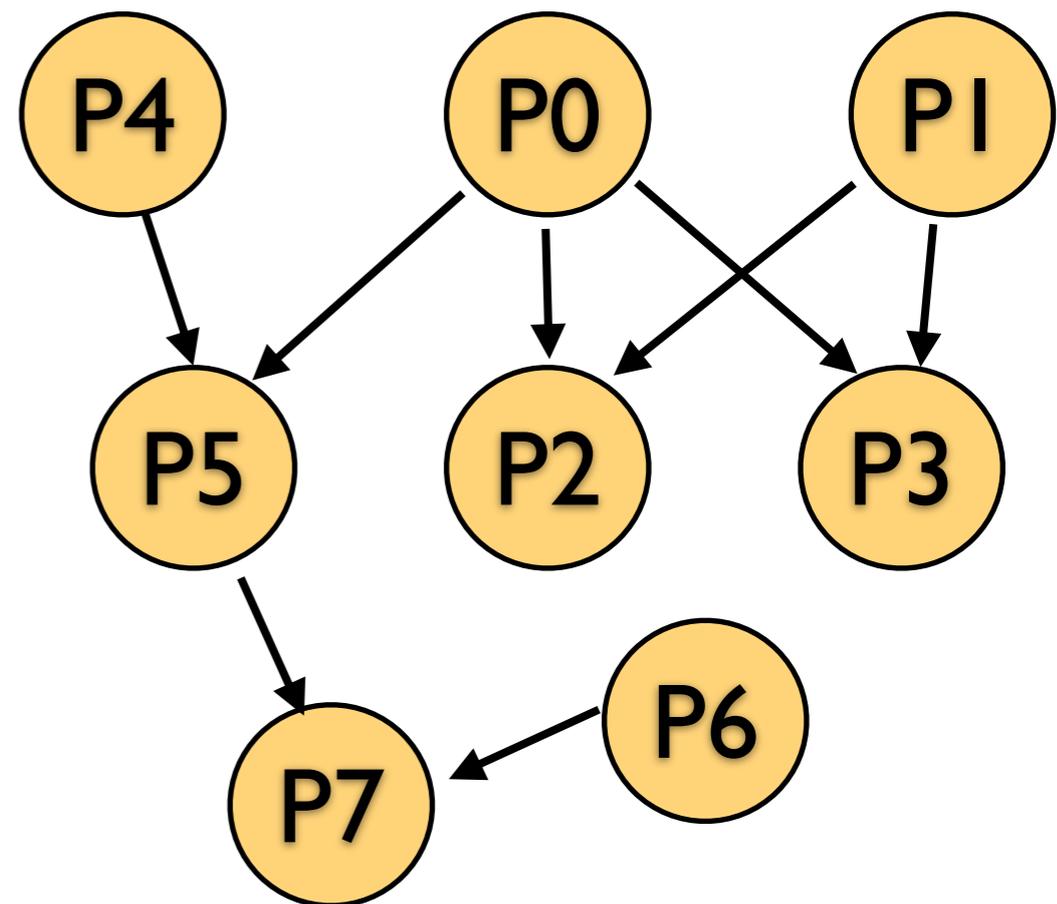
P6



STEFANO

P7

Marco e' padre di LUCA e di PIPPO  
Stefania e' madre di LUCA e di PIPPO  
Arianna e' figlia di Marco e Lucia  
Stefano e' figlio di Arianna e Rinaldo





# La nostra popolazione (codice C)

```
int main ()
{

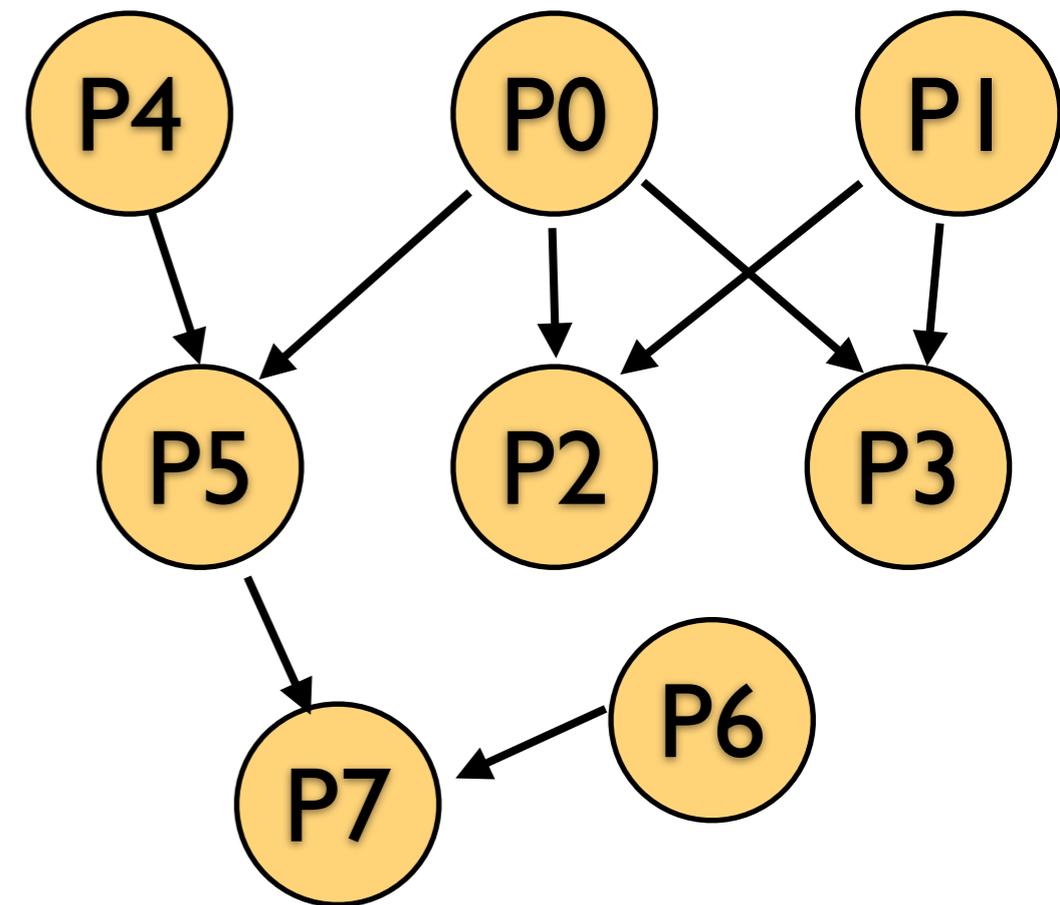
    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;

    persona p0 = crea_persona(MASCHIO, "MARCO", 50);
    persona p1 = crea_persona(FEMMINA, "STEFANIA", 49);
    persona p2 = crea_persona(MASCHIO, "LUCA", 30);
    persona p3 = crea_persona(MASCHIO, "PIPP0", 26);
    persona p4 = crea_persona(FEMMINA, "LUCIA", 53);
    persona p5 = crea_persona(FEMMINA, "ARIANNA", 30);
    persona p6 = crea_persona(MASCHIO, "RINALDO", 32);
    persona p7 = crea_persona(MASCHIO, "STEFANO", 10);

    aggiungi_a_popolazione(popolazione, &p0, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p1, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p2, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p3, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p4, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p5, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p6, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p7, &cardinalita_popolazione);
}
```



# Aggiungiamo le parentele





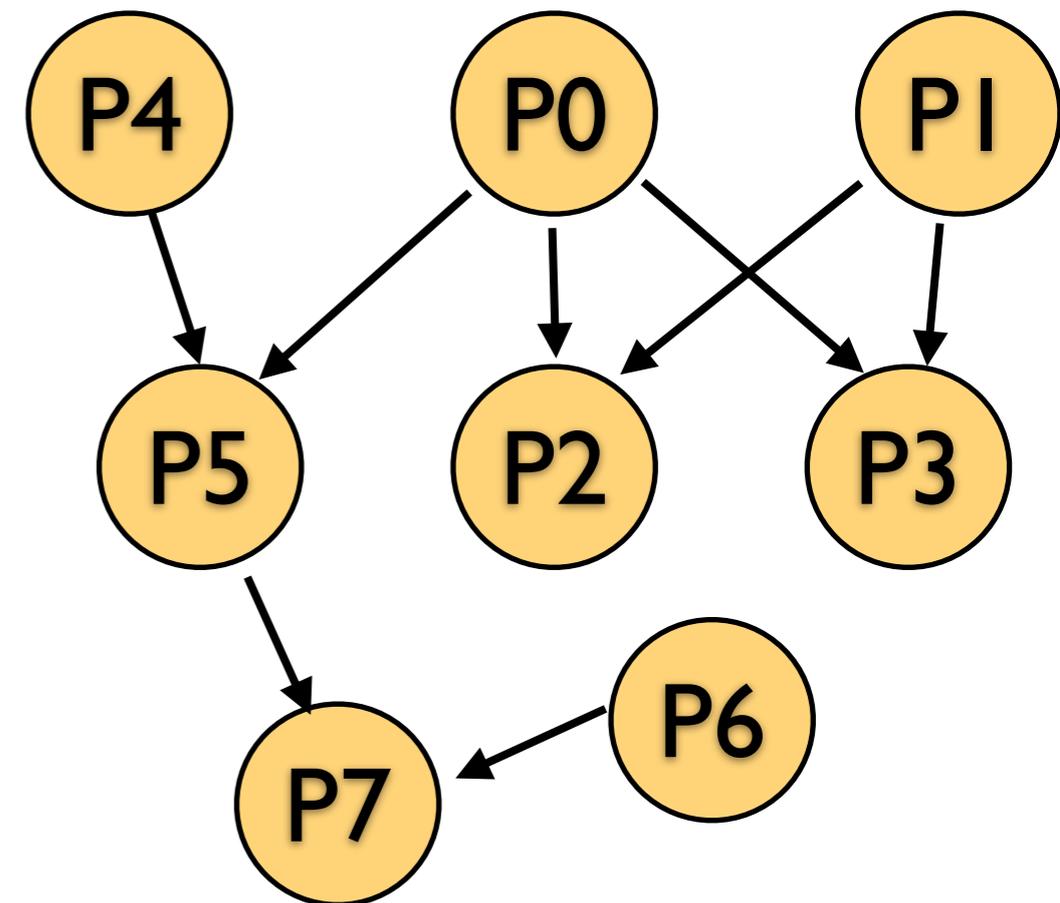
# Aggiungiamo le parentele

```
// Marco e' padre di LUCA e di PIPPO
aggiungi_figlio(popolazione[0], popolazione[2]);
aggiungi_figlio(popolazione[0], popolazione[3]);

// Stefania e' madre di LUCA e di PIPPO
aggiungi_figlio(popolazione[1], popolazione[2]);
aggiungi_figlio(popolazione[1], popolazione[3]);

// Arianna e' figlia di Marco e Lucia
aggiungi_figlio(popolazione[0], popolazione[5]);
aggiungi_figlio(popolazione[4], popolazione[5]);

// Stefano e' figlio di Arianna e Rinaldo
aggiungi_figlio(popolazione[5], popolazione[7]);
aggiungi_figlio(popolazione[6], popolazione[7]);
```





# Il main()

```
int main ()
{

    persona* popolazione[MAX_PERSONE];
    int cardinalita_popolazione = -1;

    persona p0 = crea_persona(MASCHIO, "MARCO", 50);
    persona p1 = crea_persona(FEMMINA, "STEFANIA", 49);
    persona p2 = crea_persona(MASCHIO, "LUCA", 30);
    persona p3 = crea_persona(MASCHIO, "PIPP0", 26);
    persona p4 = crea_persona(FEMMINA, "LUCIA", 53);
    persona p5 = crea_persona(FEMMINA, "ARIANNA", 30);
    persona p6 = crea_persona(MASCHIO, "RINALDO", 32);
    persona p7 = crea_persona(MASCHIO, "STEFANO", 10);

    aggiungi_a_popolazione(popolazione, &p0, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p1, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p2, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p3, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p4, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p5, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p6, &cardinalita_popolazione);
    aggiungi_a_popolazione(popolazione, &p7, &cardinalita_popolazione);

    // Marco e' padre di LUCA e di PIPPO
    aggiungi_figlio(popolazione[0], popolazione[2]);
    aggiungi_figlio(popolazione[0], popolazione[3]);

    // Stefania e' madre di LUCA e di PIPPO
    aggiungi_figlio(popolazione[1], popolazione[2]);
    aggiungi_figlio(popolazione[1], popolazione[3]);

    // Arianna e' figlia di Marco e Lucia
    aggiungi_figlio(popolazione[0], popolazione[5]);
    aggiungi_figlio(popolazione[4], popolazione[5]);

    // Stefano e' figlio di Arianna e Rinaldo
    aggiungi_figlio(popolazione[5], popolazione[7]);
    aggiungi_figlio(popolazione[6], popolazione[7]);

    elenca_figli_nipoti(popolazione, popolazione[0], MASCHIO, 3);

    return 0;
}
```

**See You Next Time!**

**Tutte il materiale sar  disponibile sul mio  
sito internet:**

**[alessandronacci.it](http://alessandronacci.it)**





# IEIM

## Esercizio: Il Gioco dell'Impiccato

Alessandro A. Nacci

nacci@elet.polimi.it - [alessandronacci.it](http://alessandronacci.it)



# Il gioco dell'impiccato

- Scrivere un programma che permetta di giocare al gioco dell'impiccato



-----  
A B C D E F G H I J K L M  
N O P Q R S T U V W X Y Z

```
bash
c o _ _ _ _ _
Inserisci una lettera
m
Hai a disposizione ancora 8 tentativi
c o m _ _ _ _
Inserisci una lettera
r
Hai a disposizione ancora 8 tentativi
c o m _ _ _ r
Inserisci una lettera
t
Hai a disposizione ancora 8 tentativi
c o m _ _ t _ r
Inserisci una lettera
p
Hai a disposizione ancora 8 tentativi
c o m p _ t _ r
Inserisci una lettera
u
Hai a disposizione ancora 8 tentativi
c o m p u t _ r
Inserisci una lettera
e
Hai a disposizione ancora 8 tentativi
c o m p u t e r
Complimenti!
MacBook-Pro-di-Alessandro-Nacci:es5 alessandronacci$
```



# Vediamo cosa ci serve...

- L'idea è quella di avere
  - un dizionario di parole da indovinare
  - un numero massimo di tentativi
  - lo stato delle parola (quali lettere sono state indovinate?)



# Dichiarazione ed inizializzazione variabili

```
//Dizionario di parole
char dizionario[5][20]={"computer", "proiettore", "aula", "ciao", "telecomando"};

int i;
int tentativi=0;
int found=0;

//Puntatore alla parola da indovinare, scelta casualmente
char* parola_da_indovinare;
parola_da_indovinare = dizionario[scegliParola()];

//Campo di gioco, stato della parola
char stato_parola[20];

//Lunghezza della parola da indovinare
int len = strlen(parola_da_indovinare);

char lettera;

for(i=0;i<len;i++)
    stato_parola[i]='_'; //Inizializzo il campo da gioco con '_'
```



# Dichiarazione ed inizializzazione variabili

```
//Dizionario di parole
char dizionario[5][20]={"computer", "proiettore", "aula", "ciao", "telecomando"};

int i;
int tentativi=0;
int found=0;

//Puntatore alla parola da indovinare, scelta casualmente
char* parola_da_indovinare;
parola_da_indovinare = dizionario[scegliParola()];

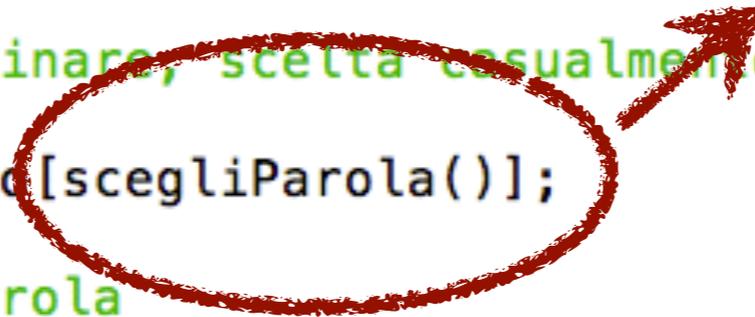
//Campo di gioco, stato della parola
char stato_parola[20];

//Lunghezza della parola da indovinare
int len = strlen(parola_da_indovinare);

char lettera;

for(i=0;i<len;i++)
    stato_parola[i]='_'; //Inizializzo il campo da gioco con '_'
```

come la implemento?





# Scegli parola

```
int scegliParola(){  
    int num;  
    srand(time(0));  
    //Estraggo un numero tra 0 e "numero di parole del dizionario"  
    num=rand()%5;  
    return num;  
}
```



## Ragioniamo sulle funzioni necessarie...

- Ora ci servono ancora alcune funzionalità:
  - Controllare se una data lettera è corretta o meno
  - Sostituire le lettere indovinate al posto dei trattini ‘\_’
  - Stampare a schermo lo stato attuale della parola
  - Controllare se il giocatore ha vinto



# Controllo e sostituzione lettera

- Ora ci servono ancora alcune funzionalità:
- Controllare se una data lettera è corretta o meno
- Sostituire le lettere indovinate al posto dei trattini ‘\_’

```
int selezionaLettere(char * parola_da_indovinare,  
                   char* stato_parola, char lettera)
```

- Controllare se il giocatore ha vinto



# Controllo e sostituzione lettera

```
int selezionaLettere(char * parola_da_indovinare,
                    char* stato_parola, char lettera)
{
    int i;
    int count=0;

    //Segno nel campo da gioco, tutte le posizioni
    //corrispondenti alla lettera scelta dall'utente
    for(i=0;i<strlen(parola_da_indovinare);i++)
        if(parola_da_indovinare[i]==lettera)
        {
            stato_parola[i]=lettera;
            count++;
        }

    if(count==0)
        return 0;
    else
        return 1;
}
```



# Controllo vincita & stampa a schermo



# Controllo vincita & stampa a schermo

```
//Controllo se la parola è stata completata
int check(char* stato_parola, int len){
    int i;
    for(i=0;i<len;i++)
        if(stato_parola[i]=='_')
            return 0;

    return 1;
}
```



# Controllo vincita & stampa a schermo

```
//Controllo se la parola è stata completata
int check(char* stato_parola, int len){
    int i;
    for(i=0;i<len;i++)
        if(stato_parola[i]=='_')
            return 0;

    return 1;
}
```

```
void stampa(char* parola, int len, int tentativi){

    int i;
    printf("Hai a disposizione ancora %d tentativi\n",
           |TENTATIVI - tentativi);
    for(i=0;i<len;i++)
        printf("%c ", parola[i]);
    printf("\n");
}
```



# Gestione del gioco



# Gestione del gioco

```
while(!found && tentativi<TENTATIVI){
//Chiedo lettere, finchè non indovina la parola o esaurisce i tentativi

    stampa(stato_parola, len, tentativi);
    printf("Inserisci una lettera\n");
    scanf("%c", &lettera);
    getchar();

    int result = selezionaLettere(parola_da_indovinare, stato_parola, lettera);
    if(result==0)
        tentativi++;
    else if(check(stato_parola, len))
        found=1;
}

stampa(stato_parola, len, tentativi);
if(found==1)
    printf("Complimenti!\n");
else
    printf("Hai superato i tentativi a disposizione\n");

return 0;
```

**See You Next Time!**

**Tutte il materiale sarà disponibile sul mio  
sito internet:**

[alessandronacci.it](http://alessandronacci.it)

