**INIZIAMO CON RICORSIONE CHE CERCA SOLO LA SOMMA MAGGIORE**

int sommaCamminoPari(TipoAlbero a) {

if (a -> sinistro == NULL && a -> destro == NULL) {

int val = radice(a);

printf("Radice a: %d\n", radice(a));

printf(" NO\_CHILD val= %d\n", val);

return val;

}

else if (sinistro(a) != NULL && destro(a) == NULL) {

int val = radice(a) + sommaCamminoPari(sinistro(a));

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d\n", val);

return val;

}

else if (sinistro(a) == NULL && destro(a) != NULL) {

int val = radice(a) + sommaCamminoPari(destro(a));

printf("Radice a: %d\n", radice(a));

printf(" DES\_CHILD val= %d\n", val);

return val;

}

else {

int sommaSinistra = radice(a) + sommaCamminoPari(sinistro(a));

int sommaDestra = radice(a) + sommaCamminoPari(destro(a));

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d - DES\_CHILD val= %d\n", sommaSinistra, sommaDestra);

if (sommaSinistra >= sommaDestra) {

printf(" SELECTED val SIN= %d\n", sommaSinistra);

return sommaSinistra;

}

else {

printf(" SELECTED val DES= %d\n", sommaDestra);

return sommaDestra;

}

}

}

Albero Input:

2

\_\_\_\_\_/ \\_\_\_\_\_

/ \

3 2

\_/ \\_ \\_

/ \ \

7 6 2

|

/

8

Soluzione corretta 6

Radice a: 8

NO\_CHILD val= 8

Radice a: 7

SIN\_CHILD val= 15

Radice a: 6

NO\_CHILD val= 6

Radice a: 3

SIN\_CHILD val= 18 - DES\_CHILD val= 9

SELECTED val SIN= 18

Radice a: 2

NO\_CHILD val= 2

Radice a: 2

DES\_CHILD val= 4

Radice a: 2

SIN\_CHILD val= 20 - DES\_CHILD val= 6

SELECTED val SIN= 20

**INTRODUCIAMO LA STRUTTURA MA LAVORIAMO SOLO CON SOMMA COME ESERCIZIO PRECEDENTE. IMPOSTIAMO contaValore ad un valore fittizio**

typedef struct Info {

int somma;

int contaValore;

} TipoInfoPercorso;

typedef TipoInfoPercorso\* InfoPercorso;

int sommaCamminoPari(TipoAlbero a) {

if (a -> sinistro == NULL && a -> destro == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a);

printf("Radice a: %d\n", radice(a));

printf(" NO\_CHILD val= %d\n", rad -> somma);

return rad -> somma;

}

else if (sinistro(a) != NULL && destro(a) == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a) + sommaCamminoPari(sinistro(a));

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d\n", rad -> somma);

return rad -> somma;

}

else if (sinistro(a) == NULL && destro(a) != NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a) + sommaCamminoPari(destro(a));

printf("Radice a: %d\n", radice(a));

printf(" DES\_CHILD val= %d\n", rad -> somma);

return rad -> somma;

}

else {

InfoPercorso sommaSinistra = (InfoPercorso) malloc(1000);

InfoPercorso sommaDestra = (InfoPercorso) malloc(1000);

sommaSinistra -> somma = radice(a) + sommaCamminoPari(sinistro(a));

sommaSinistra -> contaValore = 0;

sommaDestra -> somma = radice(a) + sommaCamminoPari(destro(a));

sommaDestra -> contaValore = 0;

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d - DES\_CHILD val= %d\n", sommaSinistra -> somma, sommaDestra -> somma);

if (sommaSinistra -> somma >= sommaDestra -> somma) {

printf(" SELECTED val SIN= %d\n", sommaSinistra -> somma);

return sommaSinistra -> somma;

}

else {

printf(" SELECTED val DES= %d\n", sommaDestra -> somma);

return sommaDestra -> somma;

}

}

}

2

\_\_\_\_\_/ \\_\_\_\_\_

/ \

3 2

\_/ \\_ \\_

/ \ \

7 6 2

|

/

8

Soluzione corretta 6

Radice a: 8

NO\_CHILD val= 8

Radice a: 7

SIN\_CHILD val= 15

Radice a: 6

NO\_CHILD val= 6

Radice a: 3

SIN\_CHILD val= 18 - DES\_CHILD val= 9

SELECTED val SIN= 18

Radice a: 2

NO\_CHILD val= 2

Radice a: 2

DES\_CHILD val= 4

Radice a: 2

SIN\_CHILD val= 20 - DES\_CHILD val= 6

SELECTED val SIN= 20

Soluzione studente 20

**INTRODUCIAMO LA STRUTTURA COME VALORE DI RITORNO. LASCIAMO ALGORITMO SU RICERCA SOMMA E VALORE FITTIZIO contaValore**

typedef struct Info {

int somma;

int contaValore;

} TipoInfoPercorso;

typedef TipoInfoPercorso\* InfoPercorso;

InfoPercorso sommaCamminoPari(TipoAlbero a) {

if (a -> sinistro == NULL && a -> destro == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a);

printf("Radice a: %d\n", radice(a));

printf(" NO\_CHILD val= %d\n", rad -> somma);

return rad;

}

else if (sinistro(a) != NULL && destro(a) == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a) + sommaCamminoPari(sinistro(a)) -> somma;

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d\n", rad -> somma);

return rad;

}

else if (sinistro(a) == NULL && destro(a) != NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad -> contaValore = 0;

rad -> somma = radice(a) + sommaCamminoPari(destro(a)) -> somma;

printf("Radice a: %d\n", radice(a));

printf(" DES\_CHILD val= %d\n", rad -> somma);

return rad;

}

else {

InfoPercorso sommaSinistra = (InfoPercorso) malloc(1000);

InfoPercorso sommaDestra = (InfoPercorso) malloc(1000);

sommaSinistra -> somma = radice(a) + sommaCamminoPari(sinistro(a)) -> somma;

sommaSinistra -> contaValore = 0;

sommaDestra -> somma = radice(a) + sommaCamminoPari(destro(a)) -> somma;

sommaDestra -> contaValore = 0;

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD val= %d - DES\_CHILD val= %d\n", sommaSinistra -> somma, sommaDestra -> somma);

if (sommaSinistra -> somma >= sommaDestra -> somma) {

printf(" SELECTED val SIN= %d\n", sommaSinistra -> somma);

return sommaSinistra;

}

else {

printf(" SELECTED val DES= %d\n", sommaDestra -> somma);

return sommaDestra;

}

}

}

**ORA POSSIAMO MODIFICARE L’ALGORITMO COME RICHIESTO DALL’ESERCIZIO, OVVERO CONTANTO I PARI E VALUTANDO LA SCELTA DEL PERCORSO CON Più PARI ALL’INTERNO**

typedef struct Info {

int somma;

int contaValore;

} TipoInfoPercorso;

typedef TipoInfoPercorso\* InfoPercorso;

InfoPercorso sommaCamminoPari(TipoAlbero a) {

if (a -> sinistro == NULL && a -> destro == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

if(radice(a) % 2 == 0) {

rad -> contaValore = 1;

} else {

rad -> contaValore = 0;

}

rad -> somma = radice(a);

printf("Radice a: %d\n", radice(a));

printf(" NO\_CHILD\_val= %d contaValore=%d\n", rad -> somma, rad -> contaValore);

return rad;

}

else if (sinistro(a) != NULL && destro(a) == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad = sommaCamminoPari(sinistro(a));

if(radice(a) % 2 == 0) {

rad -> contaValore += 1;

}

rad -> somma += radice(a);

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD\_val= %d contaValore=%d\n", rad -> somma, rad -> contaValore);

return rad;

}

else if (sinistro(a) == NULL && destro(a) != NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad = sommaCamminoPari(destro(a));

if(radice(a) % 2 == 0) {

rad -> contaValore += 1;

}

rad -> somma += radice(a);

printf("Radice a: %d\n", radice(a));

printf(" DES\_CHILD\_val= %d contaValore=%d\n", rad -> somma, rad -> contaValore);

return rad;

}

else {

InfoPercorso sommaSinistra = (InfoPercorso) malloc(1000);

InfoPercorso sommaDestra = (InfoPercorso) malloc(1000);

sommaSinistra = sommaCamminoPari(sinistro(a));

if(radice(a) % 2 == 0) {

sommaSinistra -> contaValore += 1;

}

sommaSinistra -> somma += radice(a);

sommaDestra = sommaCamminoPari(destro(a));

if(radice(a) % 2 == 0) {

sommaDestra -> contaValore += 1;

}

sommaDestra -> somma += radice(a);

printf("Radice a: %d\n", radice(a));

printf(" SIN\_CHILD\_val=%d contaValore=%d - DES\_CHILD\_val=%d contaValore=%d\n", sommaSinistra -> somma, sommaSinistra -> contaValore, sommaDestra -> somma, sommaDestra -> contaValore);

if (sommaSinistra -> contaValore > sommaDestra -> contaValore) {

printf(" SELECTED val\_SIN= %d\n", sommaSinistra -> somma);

return sommaSinistra;

}

else if (sommaSinistra -> contaValore < sommaDestra -> contaValore) {

printf(" SELECTED val\_DES= %d\n", sommaDestra -> somma);

return sommaDestra;

}

else {

if (sommaSinistra -> somma >= sommaDestra -> somma) {

printf(" SELECTED val\_SIN= %d\n", sommaSinistra -> somma);

return sommaSinistra;

} else {

printf(" SELECTED val\_DES= %d\n", sommaDestra -> somma);

return sommaDestra;

}

}

}

}

**CODICE FINALE CHE NON DA ERRORI**

typedef struct Info {

int somma;

int contaValore;

} TipoInfoPercorso;

typedef TipoInfoPercorso\* InfoPercorso;

InfoPercorso sommaCamminoPari(TipoAlbero a) {

if (a -> sinistro == NULL && a -> destro == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

if(radice(a) % 2 == 0) {

rad -> contaValore = 1;

} else {

rad -> contaValore = 0;

}

rad -> somma = radice(a);

return rad;

}

else if (sinistro(a) != NULL && destro(a) == NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad = sommaCamminoPari(sinistro(a));

if(radice(a) % 2 == 0) {

rad -> contaValore += 1;

}

rad -> somma += radice(a);

return rad;

}

else if (sinistro(a) == NULL && destro(a) != NULL) {

InfoPercorso rad = (InfoPercorso) malloc(1000);

rad = sommaCamminoPari(destro(a));

if(radice(a) % 2 == 0) {

rad -> contaValore += 1;

}

rad -> somma += radice(a);

return rad;

}

else {

InfoPercorso sommaSinistra = (InfoPercorso) malloc(1000);

InfoPercorso sommaDestra = (InfoPercorso) malloc(1000);

sommaSinistra = sommaCamminoPari(sinistro(a));

if(radice(a) % 2 == 0) {

sommaSinistra -> contaValore += 1;

}

sommaSinistra -> somma += radice(a);

sommaDestra = sommaCamminoPari(destro(a));

if(radice(a) % 2 == 0) {

sommaDestra -> contaValore += 1;

}

sommaDestra -> somma += radice(a);

if (sommaSinistra -> contaValore > sommaDestra -> contaValore) {

return sommaSinistra;

}

else if (sommaSinistra -> contaValore < sommaDestra -> contaValore) {

return sommaDestra;

}

else {

if (sommaSinistra -> somma >= sommaDestra -> somma) {

return sommaSinistra;

} else {

return sommaDestra;

}

}

}

}

int cammino\_pari(TipoAlbero a) {

if (a != NULL) {

return sommaCamminoPari(a) -> somma;

}

else return 0;

}