UNIVERSITY OF TRENTO - Italy

Department of Information and Communication Technology

Laboratorio 21
Ripasso Esercizi Teorici

Pierluigi Roberti
Carmelo Ferrante

DISI – aa 2024/2025
Università degli Studi di Trento
pierluigi.roberti@unitn.it

16-01-15 Prova PROG1

Si scriva **esattamente** l’output (istruzioni printf), nel riquadro a fianco, che viene prodotto durante l’esecuzione del programma.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 4
typedef int GGMM[MAX];
typedef int Anno[MAX];
GGMM gm;
Anno a;
int f1 (int anno[ ], int ggmm[ ], int* dim) {
   *dim = anno[(*dim)%MAX];
   printf("dim=%d \n", *dim);
   *dim = (*dim) % MAX;
   printf("dim=%d \n", *dim);
   anno[(*dim)]*=ggmm[(*dim)];
   printf("anno[%d]=%d \n", *dim, anno[(*dim)]);
   return anno[(*dim)]+ggmm[(*dim)];
}
main() {
   int i, k;
   for (i=0;i< MAX;i++)     {  scanf("%d", &a[i]);  }  //anno nascita
   for (i=2;i< MAX;i++)     {  scanf("%d", &gm[i]);  } //mese nascita
   for (i=0;i<2;i++)        {  scanf("%d", &gm[i]);  } //giorno nascita
   for (i=0; i< MAX; i++)  {  printf("%d %d ", a[i], gm[i]);   }
   k=gm[1];
   printf("K=%d \n", k);
   printf("Output: %d\n", f1(a, gm, &k));
   printf("K=%d \n", k);
   for (i=0; i< MAX; i++)  {  printf("%d ", a[i]);   }
}
```

Esercizio simile

Si scriva **esattamente** l'output (istruzioni printf), nel riquadro a fianco, che viene prodotto durante l'esecuzione del programma.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 4
typedef int Data[MAX];
Data a, gm;
void f0(int* k) { *k = MAX % *k;}
int f1(Data ggmm, Data anno) {
  int* dim; int i;
  for (i=0; i<MAX; i++)   {
          anno[i]+=ggmm[i]%2; printf("%d ",ggmm[i]);
  }
  for (i=0;i<MAX;i++)   {
          f0(&ggmm [i]); printf("%d ", anno[i]);
  }
  dim = &anno[ 2 ];
  *dim = *dim % MAX;
  printf("*dim =%d \n", *dim);
  return anno[(*dim)]+ggmm[(*dim)];
}
main()  {
  for (int i=0;i< MAX; i++) { scanf("%d", &a[i]); }
  for (i=0; i<2; i++)        { scanf("%d", &gm[i]);  }
  for (i=2; i< MAX; i++)     { scanf("%d", &gm[i]);  }
  for (i=MAX-1; i>=0; i--)   { printf("%d %d ",a[ i ], gm[ MAX-1-i ]); }
  printf("OUT= %d", f1(a, gm) );
  for (i=MAX-1; i>=0; i--)   { printf("%d %d ",a[ MAX-1-i ], gm[ i ]); }
}
```

01-09-20 Prova PROG1

[Teo04-3punti]

[TEO] Dato un vettore di valori interi di dimensione generica N, per esempio:

```
#define N 5
int vet[N] = { 3, 0, -5, 7, 8 };
```

Scrivere una funzione ricorsiva **SommaRicorsiva** che, passato in modo opportuno il vettore vet, ritorni la somma degli elementi

Mostrare inoltre la parte dell'invocazione della funzione, con la dichiarazione di eventuali variabili ritenute utili.

Soluzione 1 (dimensione nota)

```
#define N 5
int vet[N] = { 3, 0, -5, 7, 8 };

// passo il vettore e l'indice da cui partire per la somma
int SommaRicorsiva(int vet[], int i) {
   int somma = 0;
   if (i == N-1) {
      // caso base (ovvero la condizione di uscita)
      return vet[i];
   } else {
      /*caso induttivo (ovvero la ricorsione)
      all'elemento attuale, aggiungo la somma della porzione di vettore
      che inizia dall'elemento successivo */
      somma = vet[i] + SommaRicorsiva(vet, i+1);
   }
   return somma;
}

int main() {
   int som = SommaRicorsiva(vet, 0);
   cout << som;
}
```

Soluzione 2 (dimensione ignota)

```
#define N 5
int vet[N] = { 3, 0, -5, 7, 8 };

// passo il vettore e l'indice da cui partire per la somma
int SommaRicorsiva(int vet[], int i, int dim) {
   int somma = 0;
   if (i == dim - 1) {
      // caso base (ovvero la condizione di uscita)
      return vet[i];
   } else {
      /* caso induttivo (ovvero la ricorsione) all'elemento attuale,
      aggiungo la somma della porzione di vettore che inizia
      dall'elemento successivo */
      somma = vet[i] + SommaRicorsiva(vet, i+1, dim);
   }
   return somma;
}

int main() {
   int som = SommaRicorsiva(vet, 0, N);
   cout << som;
}
```

01-09-20 Prova PROG1

[Teo05-4punti]

[TEO] Dato il seguente albero binario di ricerca:

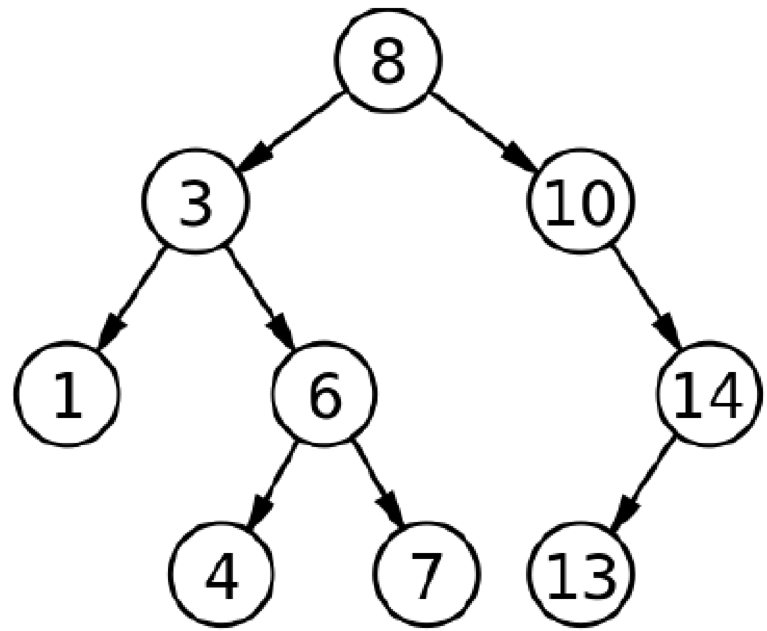
Scrivere cosa produce:

a) Visita Pre-Ordine

b) Visita Post-Ordine

c) Visita In-Ordine

d) Visita Level-Ordine



Calcolare inoltre Cammino e Altezza

Soluzione

Gli alberi possono essere visitati:

- in **pre-ordine**: prima si visita il nodo e poi i suoi sottoalberi sinistro e destro;
- in **in-ordine** (se binario): prima si visita il sottoalbero sinistro, poi il nodo e infine il sottoalbero destro;
- in **post-ordine** : prima si visitano i sottoalberi sinistro e destro, poi il nodo.
- In **level-ordine**: si visitano i nodi secondo l'ordine in cui essi appaiono sulla carta scandendo gli elementi dalla cima al fondo e da sinistra verso destra.

Quindi

- **Pre-ordine:** 8 3 1 6 4 7 10 14 13
- **In-ordine:** 1 3 4 6 7 8 10 13 14
- **Post-ordine:** 1 4 7 6 3 13 14 10 8
- **Level-ordine:** 8 3 10 1 6 14 4 7 13

Soluzione

Definizione: Il **livello di un albero** è definito ricorsivamente sulla struttura dell'albero nel modo seguente:

- la radice ha livello 0;
- ogni altro nodo ha un livello pari al livello del padre più 1.

Definizione: L'**altezza di un albero** è pari al massimo tra i livelli di tutti i suoi nodi.

Definizione: un **cammino nell'albero** è una sequenza di vertici distinti, in cui i vertici successivi sono connessi da un arco dell'albero.

La lunghezza del cammino di un albero è la somma dei livelli di tutti i nodi dell'albero.

Quindi:

Altezza = 3

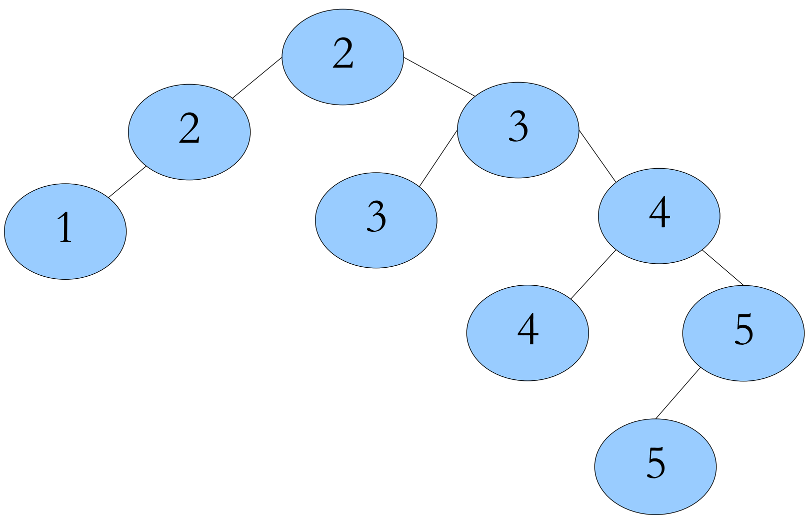
Cammino = 1 + 1 + 2 + 2 + 2 + 3 + 3 + 3 = 17

Alberi BST (btv.melezinek.cz/binary-search-tree.html)

Data la sequenza di valori:

2, 3, 3, 2, 4, 5, 5, 4 ,1

costruire un albero binario di ricerca (BST), inserendo nell'albero nell'esatto ordine i valori riportati nella sequenza, mettendo i valori uguali nel sotto ramo di SX.



Cammino : 1+1+2+2+2+3+3+4 = 18

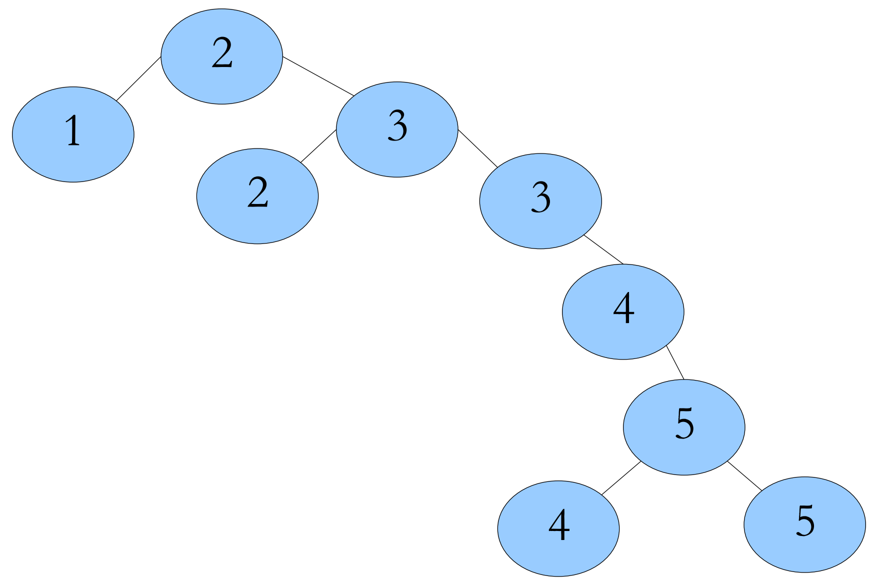
Altezza: 4

Visita pre-ordine :	2 2 1 3 3 4 4 5 5
Visita post-ordine:	1 2 3 4 5 5 4 3 2
Visita in-ordine:	1 2 2 3 3 4 4 5 5
Visita level-ordine:	2 2 3 1 3 4 4 5 5

Data la sequenza di valori:

2, 3, 3, 2, 4, 5, 5, 4 ,1

costruire un albero binario di ricerca (BST), inserendo nell'albero nell'esatto ordine i valori riportati nella sequenza, mettendo i valori uguali nel sotto ramo di DX.



Cammino : 1+1+2+2+3+4+5+5 = 23

Altezza: 5

Visita pre-ordine :	2 1 3 2 3 4 5 4 5
Visita post-ordine:	1 2 4 5 5 4 3 3 2
Visita in-ordine:	1 2 2 3 3 4 4 5 5
Visita level-ordine:	2 1 3 2 3 4 5 4 5

07-06-18 Prova PROG1

[Domanda 1][3.1 - punti 1]

Data la seguente sequenza di valori 4, 4, 2, 6, 3, 5, 5 costruire un albero binario di ricerca (BST), mettendo i valori uguali nel sottoramo di SX.

[3.2 - punti 3] Per l'albero così realizzato inoltre calcolare

Cammino

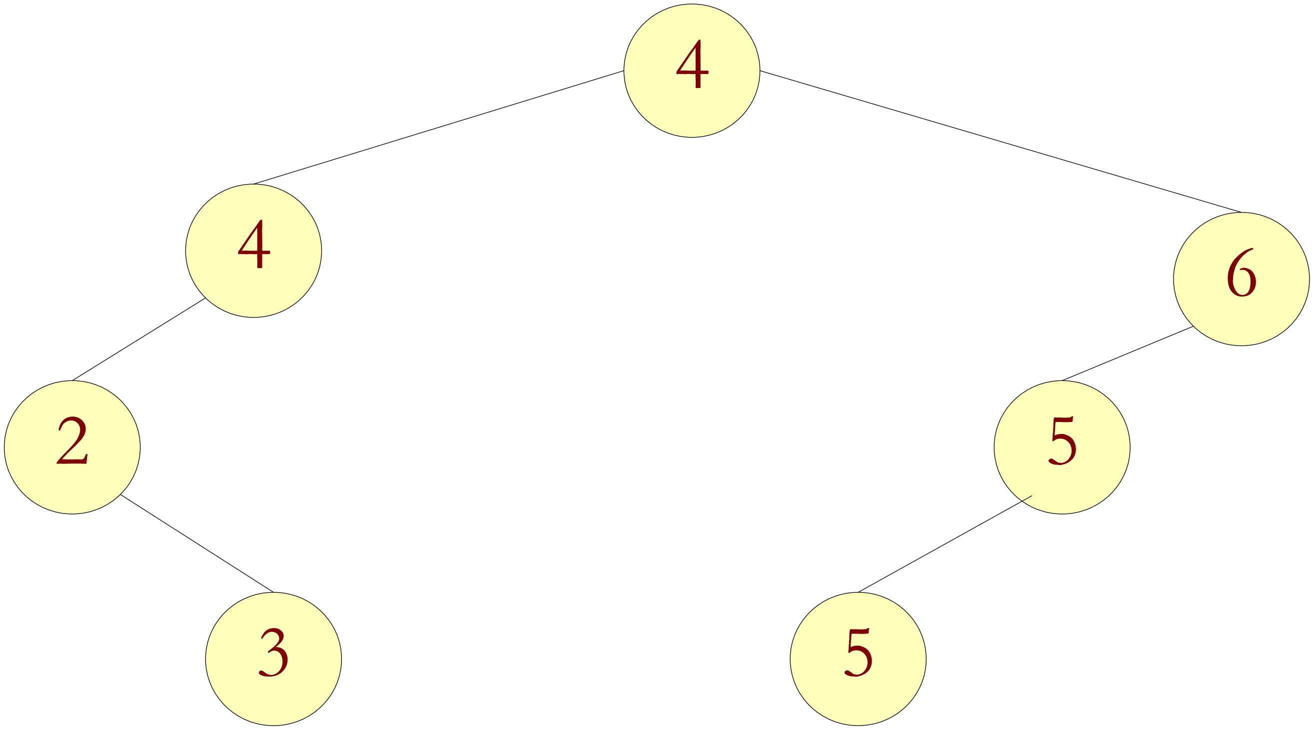
Altezza

Visita pre-ordine

Visita post-ordine

Visita in-ordine

Soluzione



Cammino : 1 + 1 + 2 + 2 + 3 + 3 = 12

Altezza: 3

Visita pre-ordine : 4 4 2 3 6 5 5

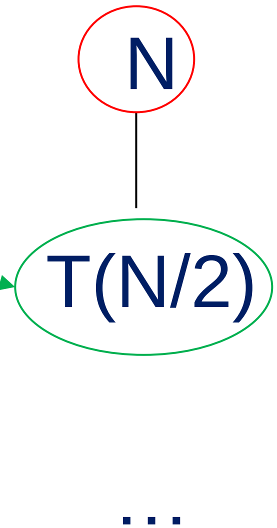
Visita post-ordine: 3 2 4 5 5 6 4

Visita in-ordine: 2 3 4 4 5 5 6

Calcolo O(n)

$T(1) = 1$

$T(N) = T(N/2) + N$



Ma sappiamo che
$T(N) = T(N/2) + N$,
quindi sostituiamo la formula nei rami, partendo però da N/2

Calcolo O(n) – sviluppo rami

$$T(1) = 1$$

$$T(N) = T(N/2) + N$$

$$
\begin{array}{c}
N \\
| \\
N/2 \\
| \\
T(N/4)=N/4 \\
\dots
\end{array}
$$

Ma ancora una volta sappiamo che

$$T(N/2) = T(N/4) + N/2,$$

quindi sostituiamo la formula nei rami, partendo però da N/4 e così via per ogni ramo

Calcolo O(n) – calcolo pesi livelli

$T(1) = 1$

$T(N) = T(N/2) + N$

i=0	N	**Sommo**	= N
		i costi	+
i=1	N/2	**per ogni**	= N/2
		ramo	+
i=2	N/4		= N/4
			+
i=3	N/8		= N/8
			…
			+1

Quando ci si deve fermare? => quando si raggiunge $N/2^i = 1$ e quindi in **$i = \log_2 N$** passaggi (livelli)

Quindi la complessità è funzione di
($N/2^i$ in sommatoria da 0 a $\log_2 N$)
$= 1 + \Sigma\, N/2^i$

Calcolo O(n) – calcolo complessità

T(1) = 1

T(N) = T(N/2) + N

$$\sum_{k=m}^{n} x^k = \frac{x^{n+1} - x^m}{x-1} \quad \text{con } x \neq 1.$$

$$\boxed{1 + \sum_{i=0}^{log_2 N-1} \frac{N}{2^i}} = 1 + N * \sum_{i=0}^{log_2 N-1} \frac{1}{2^i} =$$

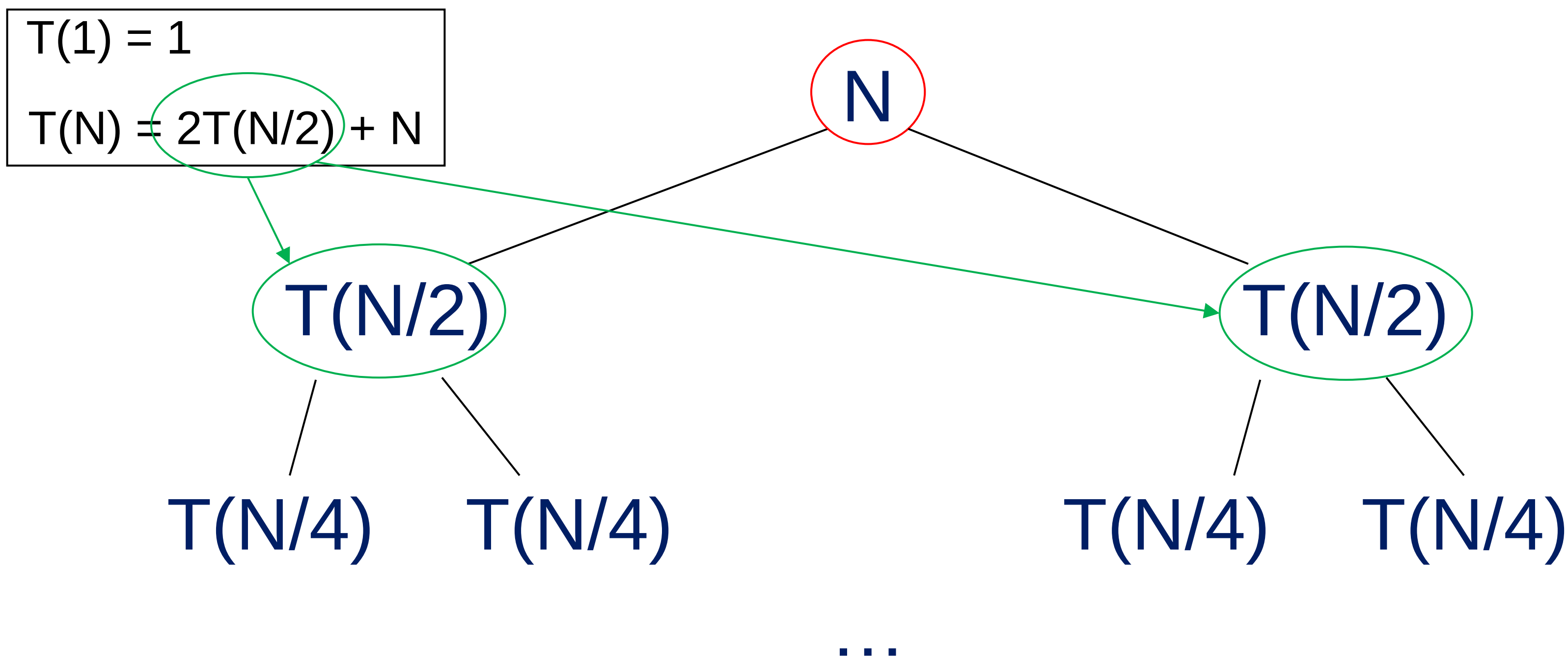
$$= 1 + N * \frac{\left(\frac{1}{2}\right)^{log2N} - 1}{\frac{1}{2} - 1} = 1 + N * \frac{\frac{1}{N} - 1}{-\frac{1}{2}} =$$

= 1 + (-2) + 2*N*

=> Complessità = **O(2N-1)**

$$\boxed{\sum_{i=0}^{\log_2 N} \frac{N}{2^i}}$$

Calcolo O(n)



Ma sappiamo che:
$T(N) = 2T(N/2) + N$
quindi creiamo due rami, partendo da N. Ora
$T(N/2) = 2T(N/4) + N/2$
quindi sostituiamo la formula nei due rami, partendo però da N/2
.. fino al limite T(1)

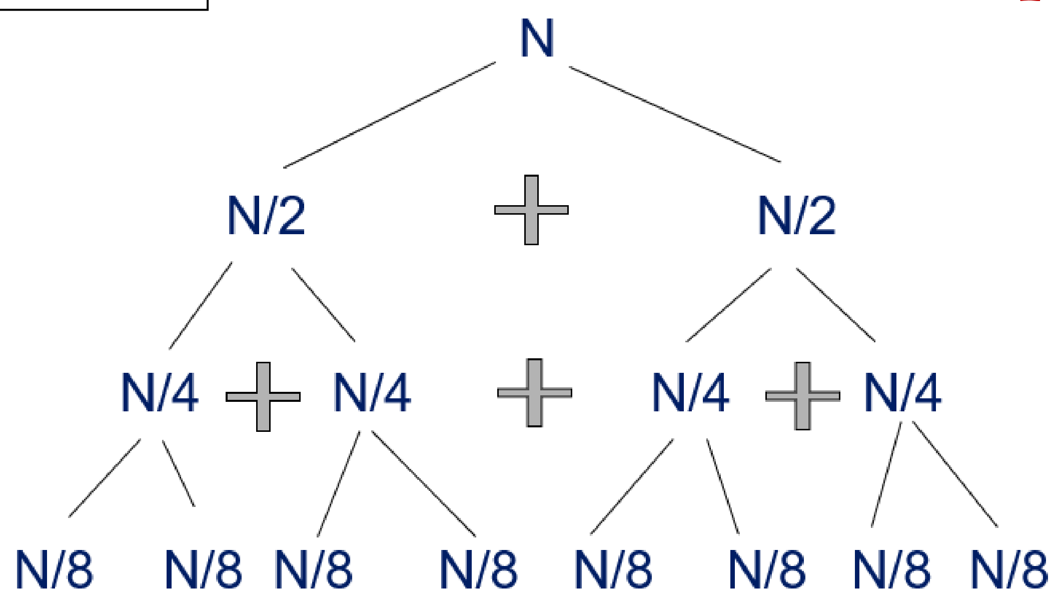
Calcolo O(n) – calcolo pesi livelli

$T(1) = 1$

$T(N) = 2T(N/2) + N$

Sommo i costi per ogni ramo

i=0	= N
	+
i=1	= N/2+N/2=N
	+
i=2	= N
	+
i=3	= N
	…
	+N



Quando ci si deve fermare? => quando si raggiunge $N/2^i = 1$ e quindi in **$i = \log_2 N$** passaggi

Quindi la complessità è funzione di

N sommato per ($\log_2 N + 1$) volte) = N log2 N + N

O(N log2 N + N) = N log2 N

03-07-18 Prova PROG1

[Domanda 4 - punti 4]

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 6
void mergesort (int a[], int left, int right){
    if (left < right) {
        int center = (left + right) / 2;
        printf("left=%d center=%d right=%d \n", left, center,
          right);
        mergesort(a, left, center);
        mergesort(a, center+1, right);
        //merge(a, left, center, right); /*si suppone
          implementata*/
    }
}
void main(){
    //Inserisci le cifre della tua matricola, una alla volta
    int m[MAX]; int i;
    for (i= 0; i<MAX; i++) { scanf("%d", &m[i]); }
    mergesort(m, 0, MAX-1);
}
```

Considerando l'esecuzione del programma su calcolatore, si scriva esattamente l'output mostrato a video.

Soluzione

- Prima chiamata con 0 5 (genera 0, 2, 5)
  - Chiamata con 0 2 (0, 1, 2)
    - Chiamata con 0 1 (0, 0, 1)
      - ~~Chiamata con 0 0 (condizione non verificata)~~
      - ~~Chiamata con 1 1 (condizione non verificata)~~
    - ~~Chiamata con 2 2 (condizione non verificata)~~
  - Chiamata con 3 5 (3, 4, 5)
    - chiamata con 3 4 (3 3 4)
      - ~~Chiamata con 3 3 (condizione non verificata)~~
      - ~~Chiamata con 4 4 (condizione non verificata)~~
    - ~~Chiamata con 5 5 (condizione non verificata)~~

Output finale:
left=0 center=2 right=5
left=0 center=1 right=2
left=0 center=0 right=1
left=3 center=4 right=5
left=3 center=3 right=4