02-11 → PAO / Algoritmi / Ro

FIRST OF ALL – Homework of PAO / RO in cartella

POI → PAO / Algoritmi (SCHEDULE DI OGGI)

VIENE BONÒ

Ciao mi chiamo Giacomo :)

PAO → Classi container (contenitore)

BOLLETTA

- → È VUOTO?
- → ADD
- → DELETE

} OPERAZIONI CLASSICHE CONTAINER

TEMPLATE → NOME <T>

↑

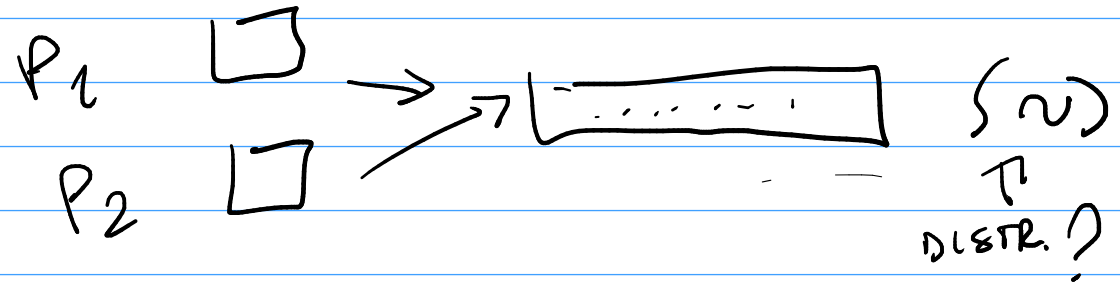
T = TYPENAME (TIPO GENERICO)

LISTA <Int>

o

LISTA <Double>

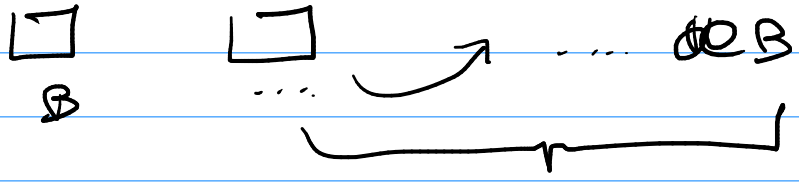
ALIASING → Interferenza in memoria



OPERATOR = (OVERLOADING)

↑ RIDEFINIZIONE

BOOLEAN OPERATOR = (CONST BOOLEAN& B)



RIF = RISPARMIO MEMORIA!

```
BOOLEAN OPERATOR = ( .... ) {
    IF (THIS != &B)        <- //NO STESSA AREA DI MEMORIA
        DELETE (FIRST)
(*) COPIA (B.FIRST)        <- COPIA PROFONDA
    RETURN THIS
}
```

↑ COPIA MEMBRO PER MEMBRO

USUALLY → [VAR = RIF. VAR] (*)

ASSEGNAZIONE

1. THIS != RIF
2. PULIZIA MEMORIA
3. ASSEGN. PROFONDA
4. THIS

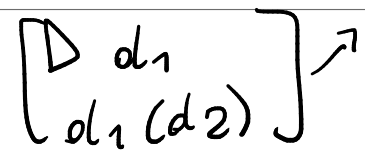
ESERCIZI SU ≠ RIDEF. STANDARD

Si considerino le seguenti definizioni.

```
class Z {
private:
  int x;
};

class B {
private:
  Z x;
};

class D: public B {
private:
  Z y;
public:
  // ridefinizione di operator=
  ...
};
```



COPIA

D& OPERATOR = (CONST D& RIF)
IF (THIS != RIF)
(y = RIF.y)
RETURN THIS

VAR. STATICA = (Z) -> HEAP NO (NEW...)

Ridefinire l'assegnazione `operator=` della classe `D` in modo tale che il suo comportamento coincida con quello dell'assegnazione standard di `D`.

D::OPERATOR = (...) (*)

```
class F: public D {
private:
  list<int*> l;
  int& ref;
  double* p;
public:
  void m() const {}
  // ridefinizione del costruttore di copia di F
};
```

F& OPERATOR = (CONST F&)
IF (THIS != RIF)
(*)
L = F.L
RIF = F.RIF
P = F.P
RETURN THIS

STANDARD

F& OPERATOR = (CONST F&) {
L = F.L
RIF = F.RIF
P = F.P }

COSTRUTTORE

F(CONST F&)
: L(F.L),
RIF(F.RIF),
P(F.P) {}

```
class S {
public:
  string s;
  S(string t): s(t) {}
};
class N {
private:
  S x;
public:
  N* next;
  N(S t, N* p): x(t), next(p) {cout << "N2 ";}
  ~N() {if (next) delete next; cout << x.s + "~N ";}
};
class C {
  N* pointer;
public:
  C(): pointer(0) {}
  ~C() {delete pointer; cout << "~C ";}
  void F(string t1, string t2 = "pippo") {
    pointer = new N(S(t1),pointer); pointer = new N(t2,pointer);
  }
};
main(){
  C* p = new C; cout << "UNO\n";
  p->F("pluto","paperino"); p->F("topolino"); cout <<"DUE\n";
  delete p; cout <<"TRE\n";
}
```

S("PLUTO")

N("

PLUTO, PIPPO

PLUTO, PAPERINO → (SOVRA SCRITTO)

P → F(PLUTO)

→ ① UNO → NO STRINGHE & NON TRACCIA NULLA

BRUTTO P → (PLUTO, PAPERINO)

P → (TOPOLINO, PIPPO)

DISTR. PROFONDA → (P) → P* ($S_1, S_2 \dots S_n$)

S = STRINGA

$S_1, S_2, S_3 \dots S_n$

CLASS D

COSTRUZIONE (VERSO = AVANTI) $D_1, D_2 \dots$

DISTRUZIONE (VERSO = INDIETRO) $D_n, D_2 \dots$

(STAMPE - ORDINE)

- COSTRUZIONE → PLUTO ① PAPERINO ② PIPPO ③
- DISTRUZIONE ← PIPPO ① PAPERINO ② PLUTO ③

Esercizio 1 Dato un array di interi $A[1..n]$, chiamiamo *gap* un indice $i \in [1, n)$ tale che $A[i + 1] - A[i] > 1$.

i. Mostrare per induzione su n che un array $A[1..n]$ tale che $A[n] - A[1] \geq n$ (quindi $n \geq 2$) contiene almeno un gap.

ii. Fornire lo pseudocodice di una procedura ricorsiva divide et impera *gap* che dato un array $A[1..n]$ tale che $A[n] - A[1] \geq n$ restituisce un gap in A.

iii. Valutare la complessità della funzione, utilizzando il master theorem.

```
if A[q] - A[p] > q-p+1
   gap(A,p,q)
else
   // note che allora A[r] - A[q] > r-q+1
   gap(A,q,r)
```

→ INDUZIONE

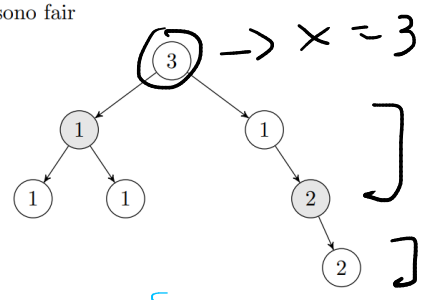
$i = 1$ $\quad A[i] \to A[i+1] < A[i+2]$

$i = 2$

$\ldots$ $\quad i = n$ $\quad A[i+2] \; A[i+1] >$
$A[i+1] \; A[i] > 1$

$T(n) = T(\frac{n}{2}) + C \to O(n) \to$ LINEARE
$O(\log(n))$
↑ DIVIDE ET IMPERA

Esercizio 10 Un nodo x di un albero binario T si dice *fair* se la somma delle chiavi nel cammino che conduce dalla radice dell'albero al nodo x (escluso) coincide con la somma delle chiavi nel sottoalbero di radice x (con x incluso). Realizzare un algoritmo ricorsivo *printFair(T)* che dato un albero T stampa tutti i suoi nodi fair. Supporre che ogni nodo abbia i campi *x.left*, *x.right*, *x.p*, *x.key*. Valutare la complessità dell'algoritmo.
Un esempio: i nodi grigi sono fair



→ $x = 3$

$S = 3$ { $\quad S = 3 = 3$

$3 = 3$

! NOT FAIR
$4 > 3$

↳ FAIRNESS

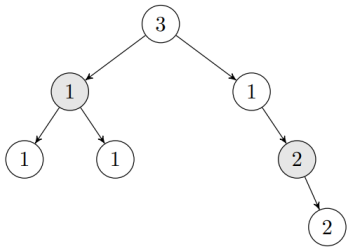
$x.p$

x $\quad x.KEY$ (VALORE)

L $\quad$ R $\quad$ LEFT / RIGHT

Esercizio 10 Un nodo x di un albero binario T si dice *fair* se la somma delle chiavi nel cammino che conduce dalla radice dell'albero al nodo x (escluso) coincide con la somma delle chiavi nel sottoalbero di radice x (con x incluso). Realizzare un algoritmo ricorsivo *printFair(T)* che dato un albero T stampa tutti i suoi nodi fair. Supporre che ogni nodo abbia i campi *x.left*, *x.right*, *x.p*, *x.key*. Valutare la complessità dell'algoritmo.

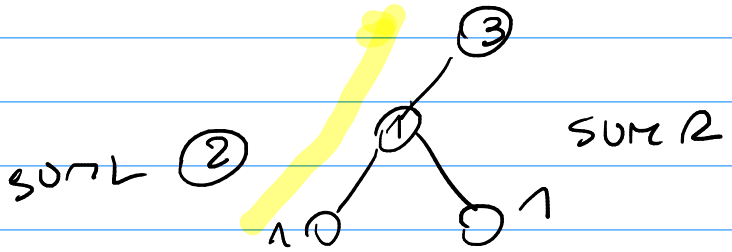
Un esempio: i nodi grigi sono fair



PRINT FAIR (T)

IF (!T) PRINT (T.ROOT) // ROOT è FAIR

[T.LEFT + T.RIGHT = T.ROOT]



WHILE (T.LEFT OOO T.RIGHT OOO SUM L + SUM R < T.ROOT.KEY)

T = T.LEFT

SUM L = SUM L + T

← PRINT (T)

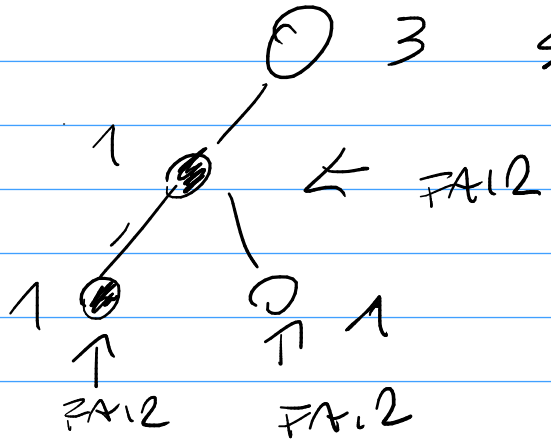
↑ NODO FAIR

T.KEY <

T = T.RIGHT

SUM R = SUM R + T

PRINT (T.ROOT)

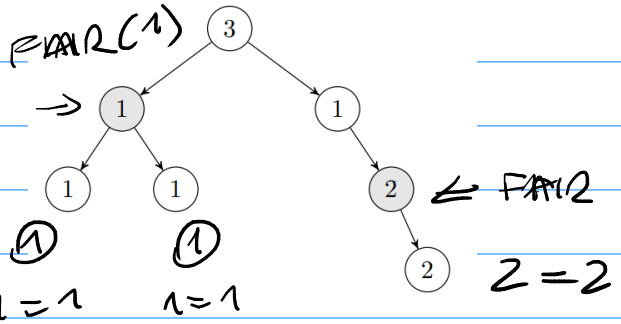


SUML, LEFT = PRINTFAIR (T.L)
IF (LEFT ≤ SUML)
PRINT (x)

```
printFair(x,path)        // x = node of the tree
                         // path=sum of the keys in the path from the root to x

                         // Action: print the fair nodes and
                        //         returns the sum of the keys in the subtree
   if (x == nil)
      return 0

   left    = printFair(x.l, path + x.key)
   right   = printFair(x.r, path + x.key)
   sumTree = left + right + x.key
   if (path == sumTree)
      print x
   return sumTree
```



SOTTOALBERO = STESSA SOMMA DI X

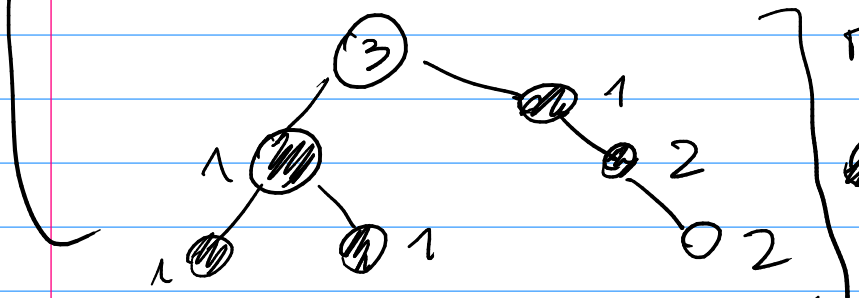
SUML + SUMR < x
=
STOP

T.root

TUTTI NODI PERCORSI

= NON FAIR

CON x = RADICE



h → O(h) = O(log |N| + 1)

Domanda 16 Data la ricorrenza $T(n) = 5T(n/3) + (n-2)^2$, trovare la soluzione asintotica.

a b $f(n)$

Limite	Risultato	Soluzione
$\lim_{n\to\infty} \frac{f(n)}{n^{\log_b a}} = k$	$k = l$	$T(n) = \Theta\left(n^{\log_b a} \log(n)\right)$
	$k = 0$	$T(n) = \Theta\left(n^{\log_b a}\right)$ se $\exists k > 0 \mid \lim_{n\to\infty} \frac{f(n)}{n^{\log_b a - \varepsilon}} = k$
	$k = \infty$	$T(n) = \Theta\left(f(n)\right)$ se $\exists k > 0 \mid \lim_{n\to\infty} \frac{f(n)}{n^{\log_b a - \varepsilon}} = k$ se $\exists k, 0 < k < 1 \mid \left[a f\left(\frac{n}{b}\right) \le k f(n)\right]$

$$\lim_{n\to\infty} \frac{(n-2)^2}{n^{\log_3 5}} = \frac{1}{0} = \infty$$

> 0

$$5\left(\frac{(n-2)^2}{3}\right) \le K(n-2)^2$$

$K = 3/2 \quad \ldots \quad \forall n \in \mathbb{N}$

CASO 3 → COND. DI REGOLARITÀ

(TROVI K PER RISOLVERE)