

## PATTERN DESIGN

podrá el ciclo vicioso.  
que

while  $i \leq j$   $\rightarrow PRE - (0 \leq i, j \leq \dim - \dim P + 1)$

## While Match

$\int \nabla K \leq 0$

while ( $k < \dim P$  &&  $T[k + i, 2:i] == f[k]) \{ R_2$

K++ ;

3

// Post- $\tau_2$  [K = dim P $\leq 0$  match Trovato] & (K < dim P $\leq 0$ )  
 $\rightarrow$  T[K = min $\tau_2$ ]  $\neq$  P[K])

$$T[\kappa_{\min}] \neq P[\kappa])$$

$$L_2 = (\exists \leq k \leq \dim P) \& \& (T[\text{in}_k, \dots, \text{in}_{k+m}] = P[\text{o}_{k+1}, \dots, \text{o}_{k+m}])$$

→ int main() {

bool TrySetFalse();

while (inizio < dim - dimP + 1 && !Trovato) { //R

## while match

$$\overline{f(K = \text{dmP})}$$

//dove ho imposto gli uscite soltanto per i libri finiti con successo  
Trovato = True ;

$\{ \cdot \} \in \mathbb{R}^{1 \times n}$

(n) 3/10 ++

$\exists \text{Post} : (\text{Trans} \Rightarrow \exists \text{match } i \text{ Pachtnahme mit } 3$   
 $\quad \quad \quad !\text{Trans} \Rightarrow \nexists \text{match})$

:F{Traverses}{

out between us' as with us

32132

out false;

$\left( \text{do } \{ \dots \text{ in } \dots \} \text{ now ex } \frac{3}{2} \text{ match} \right)$

$R \models (\phi \leq \text{início} \wedge \text{dér } P+1) \wedge \ell_1(\text{Início} \Rightarrow \exists \text{match da linha}) \wedge \ell_2(\text{Início} \Rightarrow \text{match com index in } [0 \dots \text{início} - 1])$

ESERCIZIO 2 GIOVEDÌ 26/1

calcola la sottosequenza crescente max (scrivere in  $A[0..dim-1]$ )

R2 = ( $i < f_c \leq dim-1$ )  $\&$  ( $A[i..f_c] \in \text{cresc}$ )

while ( $f_c < dim-1$   $\&$   $A[f_c] \leq A[f_c+1]$ )  $f_c++$ ;

POST2 = ( $A[i..f_c] \in \text{cresc}$ )  $\&$   $f_c = dim-1 \text{ } || \text{ } A[f_c] > A[f_c+1]$ )

int i, b = 0, f\_b <= 0, i <= 0, f\_c <= 1, b <= 1, i <= 1

while ( $f_c < dim-1$ ) { // R2 = ( $A[i..f_c] \in \text{cresc}$ )  $\&$   $A[i..f_c] \leq$ )

while ( $f_c < dim-1 \& A[f_c] \leq A[f_c+1]$ )  $f_c++$ ;

// POST2 = ( $A[i..f_c] \in \text{cresc}$ )  $\&$  ( $f_c = dim-1 \text{ } || \text{ } A[f_c] > A[f_c+1]$ )

$i_c = f_c - 1 + 1$ ; // lunghezza sequenza crescente

if ( $i_c > 1$ ) {

$b = i_c$ ;

$f_b = f_c$ ;

$b = i_c$ ;

}

if ( $f_c < dim-1$ ) {

$i_c = f_c - f_c + 1$ ;

}

} // POST2 = ( $A[i..f_b] \in \text{cresc}$  in  $A[0..dim-1]$ )

FOR

int i = 0  
for (iniz: dimensione; condizione; incremento) {

    CORPO

} RIFERIMENTI

i riferimenti ci permettono di avere alcune di varibili:

int x, &y = x;

y è un duplo di x, cioè ha le stesse R-e lo stesso L-valore. (x passato di punti al fattore).

# ES 1 Giovedì 31/1 CORREZIONE

`int A[100]; // A = int*(const)`

`char A[5][10]; // 2 dimensioni`

$A: \text{char}^*[10]$  puntatore alla prima riga  
 $\rightarrow \text{char}^*[10]$   $A+3 \equiv$  3 righe.  
 $\rightarrow A: \text{char}^* \rightarrow A+5 \equiv$  5 elements

3 dimensioni:

`char A[5][10][20]`

~~char~~  $\rightarrow$   $\begin{matrix} 5 \\ 10 \\ 20 \end{matrix}$

$A: \text{char}^*[10][20]$   
 $\rightarrow \text{char}^*[10][20]$

$\begin{matrix} 5 \\ 10 \\ 20 \end{matrix}$

$\rightarrow A: A[0] \rightarrow \text{char}^* 20$

$\rightarrow A \equiv A[0][0] \rightarrow \text{char} 20$

$\rightarrow A \equiv A[0][0][0] \rightarrow \text{char}$

`char A[5][10], *P = A`

`NP >> dim,`

`for(int i=0; i<dim; i++)`

`NP >> p[i];`

`int np = dim/10, nf = dim % 10`

`bool Trovata = false;`

`for(int i=0; i<np; i&& !Trovata); // R = (non corrisponde NPA in [0..i-1])`

`int c = 0;`

`for(int j=i; j<np; j++) {`

unmatch di 'c'

$\&R_{j-1} \neq A[i-1] \& R_j \neq A[i]$   $\&R_j \neq A[i-1] \& R_j \neq A[i]$   
 $\&R_j \neq A[i-1] \& R_j \neq A[i]$   $\&R_j \neq A[i-1] \& R_j \neq A[i]$

`if(A[i][j]) = = 'c'`

`c++;`

`} if(c < 2)`

`} if(c < 2)`

`Trovata = true;`

`} out << Trovata << i;`

`}`

$f(!\text{Found})$

: $f(\text{not } \pi)$

// Combinons la nrg nrp con not elment

: $\exists i \text{ nr dans } [0, \text{nr}[\text{ et } \text{elmt } \% i]$

bool Trouvé = false;

for (int i = 0; i < nr && !Trouvé) :+2) {} // R1

: $i \leq 0$

for (int j = 0; j < i; j++) // R2

: $f(\text{if } (\pi_j) = \text{?} \text{?})$

c<sub>42</sub>j

// Post 2

: $f(!(\text{c}\gamma_2)) \{$

OUT[Two] = c : second;

Trouvé = Two;

} //

UP<sub>2</sub> = min(UP<sub>1</sub>, c)

} // Post 1

R1 = ( $0 \leq i \leq \text{nr}$ ) && ( $\forall 0 \dots i-1 \text{ dans } \text{NP}_2$ ) && ( $\text{trouvé} \Rightarrow \forall i \dots \text{ dans } \text{NP}_2$ )

&& ( $\text{trouvé} \Rightarrow \text{OUT}[ \text{c} : \text{second} ] = \text{true}$ )

Post 1 = ( $\text{trouvé} \Rightarrow \text{min}(UP_1, UP_2)$ ) && ( $\text{trouvé} \Rightarrow \text{c} = \text{second}$ )  
(c = max(UP<sub>1</sub>, UP<sub>2</sub>))

R2 = ( $0 \leq j \leq i$ ) && ( $c = \text{max}(UP_1, UP_2) \text{ et } c \in A[i][j \dots i-1]$ )

Post 2 = ( $c = \text{max}(UP_1, UP_2) \text{ et } c \in A[i]$ )

Post 3 =  $\text{UP}_3$ .

1) construction de R1: c est dans le post de R2 = ( $0 \leq i \leq \text{nr}$ ) && ( $\forall 0 \dots i-1 \text{ dans } \text{NP}_2$ )

et aussi ( $0 \dots i-1$ ), visto d'ns. 2. Si le wmp<sub>2</sub> ès regardé le R2 le post de

2) construction: R2 && ( $j \leq i$ ) et aussi :  $f(A[i][j]) = c$ );

R2 && ( $j \leq i$ ) A[i][j] è la boîte de la nrg 1, dopo averlo confrontato con i, se doppia  
navigazione. Visto che R2 ha una var per il post, se essa fissa per A[i][Q(j-1)] è questo

~~caso~~, c'è posto per  $A[i][0 \dots j]$ , quindi deve essere  
giusto  $A[i][0 \dots j-1]$ , cosa in R2 e molto semplice da provare.

## R2. Caso base

3) ~~caso~~ di ~~base~~:  $R2 \ L2 \ | \ (j \leq i) = (\cos \alpha)^j \cdot A[i:j-1] \ L2 \ (j \leq i)$   
 $\Rightarrow (c-a \leq \alpha^j \leq 0 \ \forall j \in [0..i])$

PROVA DEL CASO ESTERNO

1) CORRISPONDENZA: ~~caso~~ è chiaro - applicabile in tutti:

$0 \leq j < i \leq m-1$ . In  $[0..m-1]$  non ci sono elementi  $j \geq i$  e non  
può contenere 1' e questo dimostra che tutto che l'ultimo  
caso non si può verificare.

2) INVARIANZA: dimostrare che  $L2 \ L2 \ | \ (j \leq i)$

Essendo tratta di  $R2$ , la riga  $i$  è non ordinata. La riga  $i$  quindi deve essere  
una concatenazione.

3) CORRISPONDENZA:  $R2 \ L2 \ | \ (c \leq \alpha^j \leq 0 \ \forall j \in [0..i]) \ L2 \ (c \leq \alpha^j \leq 0 \ \forall j \in [0..i])$   
 $\Rightarrow i \geq m-1$  (non ci sono elementi  $j \geq m-1$  con  $\alpha^j \leq 0$ )

2) Trovato un  $i \geq m-1$  tale che  $c \leq \alpha^i \leq 0$  (la riga  $i$  è  $c$ )  
non contiene  $\alpha^j$  (~~non contiene~~ contiene  $\alpha^j$  solo  $\alpha^{m-1}$ )  
 $\alpha^j$  (non contiene  $\alpha^j$  se  $i < j$ ) quindi solo post 1.

1) Trovato  $i \geq m-1$  tale che  $c \leq \alpha^i \leq 0$  e  $\alpha^j$  non contiene  $\alpha^i$  per tutti  $j < i$ :  
(la riga  $i$  è  $m-1$ ) non contiene  $\alpha^j$  ( $\alpha^j$  non contiene  $\alpha^i$ )  
che  $\alpha^j$  non contiene  $\alpha^i$  ( $\alpha^j$  non contiene  $\alpha^i$ ) e non contiene  $\alpha^j$  per tutti  $j < i$   
quindi solo post 1.

Sx(1) have expression  $\pi/2/13$

ORIGINIALE

$\sqrt{A(s)_{10}[20]}$  equivalenti tipi, divisione e valore approssimazione stessa  
 $\cdot (+A)[2] \xrightarrow{\text{+} \text{int}^*}$   $\xrightarrow{\text{+} \text{int}^*(+A)(2)}$   $\xrightarrow{\text{+} ((+A)+2)} = A+16$

$\cdot (A[-10]) \xrightarrow{\text{+} \text{int}^*(A[-10])}$   $\xrightarrow{\text{+} \text{int}^*(A[-10])}$   $\xrightarrow{\text{+} (A-0)} = (A-10) = A-30000 \xrightarrow{\text{+} \text{int}^*} A-30000+30$

$\xrightarrow{\text{+} \text{int}^* \text{ punto} \text{ 4bit}} A \approx 80$

$A: \text{int}^*(10)[20]$

A punto equivalente di dimensione  $4+10+20 = 30$  bit

EDONDA PRATICA

void R ( $\text{int}^* A$ , int dim, int lims,  $\text{int}^* \text{lim2}$ ,  $\text{int}^* r$ , ofstream &out){

int nnp = dim / lims, npf = dim % lims;

f(r < nnp)

for (int i = 0; i < lims; i++)

out << A[i \* lims + i] << endl;

else

if (r == nnp) { if (npf > 0)

for (int i = 0; i < npf; i++)

out << A[i \* lims + i] << endl;

else

out << r << endl;

void L ( $\text{int}^* A$ , int dim, int lims,  $\text{int}^* \text{lim2}$ ,  $\text{int}^* c$ , ofstream &out){

int nnp = dim / lims, npf = dim % lims;

if (dim <= c){

out << c << endl;

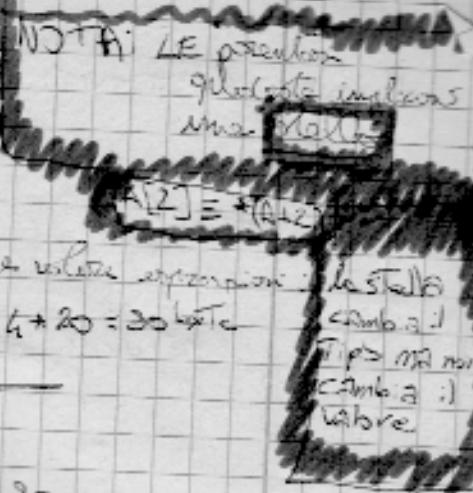
else

# for (int i = 0; i < nnp && i < lims; i++)

out << A[i \* lims] << endl;

if (nnp <= c && npf > 0)

out << A[c \* lims] << endl;



COLLEZIONE PROBLEMI TORN 1 Trova la colonna con il minimo numero di valori assenti

```
int F(int x[10][10], int dim, int &Kol) {
```

```
    if (dim <= 10) {
```

```
        Kol = 0;
```

```
        return 0;
```

} // Tutte le colonne non sono vuote.  
int min = M, N;

```
f (int i = 0, : < 10; i++) { // R = {0 ≤ i ≤ 10} & (controlla colonne 0..i-1)
```

```
    V = &x[i][0..dim-1];
```

~~le colonne vuote hanno tutti gli elementi diversi da zero e quindi valore zero in tutto le colonne 0..i-1)~~

```
    if (V < min) {
```

```
        min = V;
```

```
        Kol = i;
```

}

```
return min;
```

}

```
int C (int x[10][10], int dim, int c) {
```

```
    int nrof = dim / 10, ncf = dim % 10, L = nrof;
```

```
    f (c < ncf) L++;
```

```
    int cont = 0;
```

```
    for (int i = 0; i < L; i++) {
```

~~for bool presente = false;~~

~~del 'presente'~~

```
        for (int j = 0; j < 10; j++) {
```

```
            f (x[i][j][c] == x[i][j])) {
```

presente = true;

}

```
        if (!presente) {
```

cont++;

}

```
    return cont;
```

}



ogni riga contiene solo  
elementi diversi dall'zero

PARTIE 1

```

void F(int* X, ifstream &INP, ofstream &OUT){
    int a, i;
    INP>>a;
    while(a != -2){
        X[i%10] = a;
        if(i%10 == 9)
            OUT<<X[i%10]<<endl;
        else
            OUT<<X[i%10]<<',';

        i++;
    }
}

```

INVARIANTE INSERIMENTO WHILE

$R = (i \text{ è un intero} \geq 0, l'ultimo è} \neq -2) \wedge (i \leq 10 \Rightarrow X[0..i-1]$   
 contiene gli interi dalla prima all'ultima) \wedge (não i > 10 \Rightarrow \text{l'ultimo non è}  
 $\in X[0..9, 0..i\%10-1])$

PARTIE 2

```

void SEL(int* X, int bdim, int k){
    int ok = 0;
    for(int i = 0; i < bdim; i++){
        if(X[i] == k){
            X(ok) = X(i);
            ok++;
        }
    }
}

```

## ES TEORIA

(1)

```

nodo* F(nodo* L1, nodo* L2) {
    if(L1) return L1;
    if(L2) return L2;
    if(L1->info >= L2->info) {
        L1->next = F(L1->next, L2);
        return L1;
    } else {
        L2->next = F(L1, L2->next);
        return L2;
    }
}

```

3

PRES (L1 + L2 sono liste corrette e secolate in modo decrescente)

POST = {F restituisce una lista secolata in modo decrescente da contiene tutti i nodi di L1 e L2}

(2) void G(int x){ cout<<x+y; } // y non è dichiarata, la variabile globale viene definita dopo la definizione della funzione G  
 $\text{int } y = 10;$   
 $\text{main() \{ int } y=20, x=30, \text{G}();\}$

(3) void G(int x){ ... }  
 $\text{int } G(\text{char } x)\{ ... \}$  // il compilatore non sa quale funzione sceglierà perché non ci è nessuna funzione con precedenza double.  
 $\text{main() \{ double } y, G(y);\}$  NB: void G(int x)  $\equiv$  int G(int x)  
 $\text{void } G(\text{int } x) \neq \text{void } G(\text{char } x)$   
 La differenza tra le funzioni è nei parametri.

## ERATICA

// crea Terativa

node\* crea(:int dim, ifstream &INP){

int x;

INP >> x;

node\* init = new node(x), \*fine = init;

for(int i=0; i<dim-1; ++){

INP >> x;

fine->next = new node(x);

fine = fine->next;

}

return init;

}

Match ( $L \cdot P$  = quelli che vengono aggiunti al match di  $P$ ;  $R(LP)$  = nodi del match in  $L$  di  $P$ )

node\* F(node\* L, :int &P, int dimP){

:F(! dimP) return 0;

:F(! L) return 0;

:if(L->inf, == P[0]) {

node\* x = F(L->next, P+1, dimP-1);

node\* y = L;

L = L->next

y->next = x;

} else

: return F(L->next, P, dimP);

}

}

PRE = ( $L$  è corretto,  $P$  ha dimensione,  $\dim P \neq 0$ ,  $\text{init}(L) = \forall L$ )

POST =  $F$  ritorna e restituisce  $R(L, P)$ , con  $L$  pronto per riferire next in  $\forall L$

CASE INDUTTIVO: MATCH TROVATO

$\forall L = a \otimes \forall L'$

$a \in P[0]$

$R(\forall L, P) = a \otimes R(\forall L', P[1, \dots, \dim P - 1])$

# CORREZIONE COMPITO N° 13/3

## TEORIA

ES 1) (2 punti)

void f(:int &ka){ cout << a[1] << endl; }

void g(:int &a){ cout << a[2] << endl; }

main(){ :int a[] = {1, 2, 3}, f(x), g(x); }

costante, f riceve un riferimento di X, mentre g riceve una copia del contenuto

ES 2. (2 punti)

Se compiliamo senza controllare le correttezza delle convenzioni. [CAP 9]

ES 3 (3 punti)

:int i=10

Stampa: 1  
0

for(i=0; i<5; ++){

cout << i << endl;

, f((i+3)/2)  
cout << endl;

else

break;

}

cout << i << endl;

## PROGRAMMAZIONE

void crea(int &inizio, int dim, ifstream &INP){

if(!dim){

inizio=0;

else{

inizio=0;

:int x;

INP >> x;

inizio=nodo(x);

crea(inizio+1, dim-1, INP);

}

}

```

node* match(node* L, int P, int dimP) {
    if (!L) return 0;
    if (L->info == P[0]) {
        if (dimP == 1) { // completo o match.
            node* x = L;
            L = L->next;
            x->next = 0;
            return x;
        } else { // match non completo.
            node* x = match(L->next, P + dimP);
            f(x);
            node* y = L;
            L = L->next;
            y->next = 0;
            return y;
        }
    } else // match non pos. sono completato
    return 0;
}

```

PRE = ( $L$  è corretto,  $P$  ha dim elementi < dim $L$ , dim $P \geq 0$ ,  $L \neq 0$ )

POST = (~~Se match restituisce un valore V ≠ 0 allora esiste un match di  $P$  in  $VL$  e  $V = R(vL, P)$  e  $L = VL - P$~~  Se match, restituisce un valore  $V \neq 0$  allora esiste un match di  $P$  in  $VL$  e  $V = R(vL, P)$  e  $L = VL - P$  ) & (Se non match, restituisce 0 allora  $\forall$  elem match di  $P$  in  $VL$  e  $L = VL$ ).

else  
 return match(L->next, P, dimP);

}