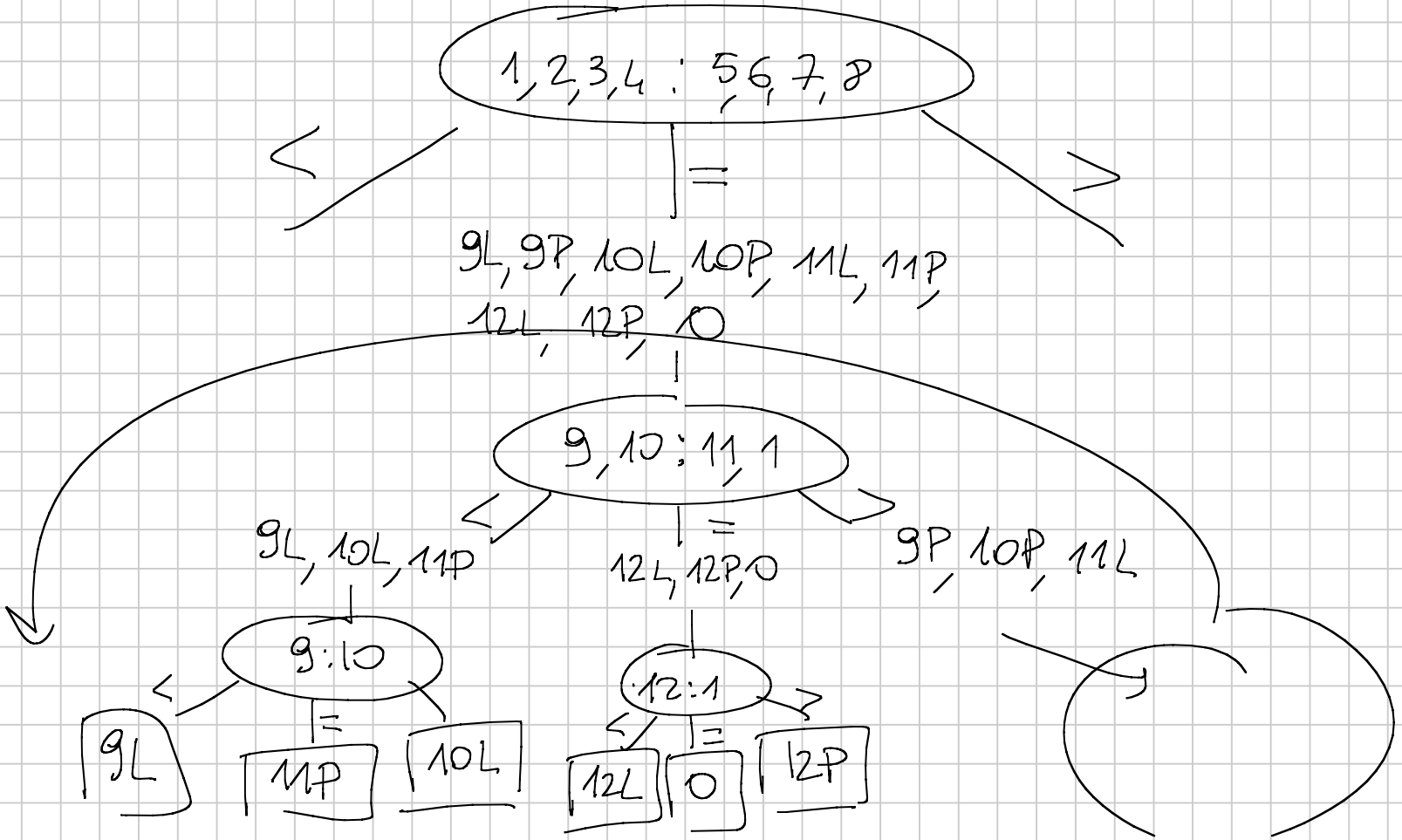
Ricevimento Studenti: Giovedì 9-11

base b $(b=2,\ b=10)$

cifre per rappresentare N

$$n = \lfloor \log_b N \rfloor + 1$$

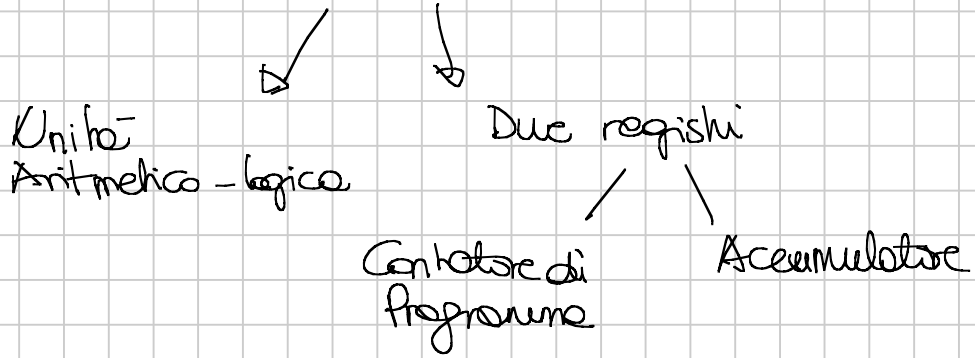
2


1,2,3,4 : 5,6,7,8
<
=
>
9L, 9P, 10L, 10P, 11L, 11P
12L, 12P, 0
9, 10 : 11, 1
<
=
>
9L, 10L, 11P
12L, 12P, 0
9P, 10P, 11L
9 : 10
12 : 1
9L
11P
10L
12L
0
12P

Modello RAM "Random Access Machine"

"Macchina ad accesso diretto"

PROCESSORE — memoria di dim. illimitata



Istruzioni Elementari

- operazioni aritmetiche (+, −, *, /)
- operazioni logiche (AND, OR, NOT)
- operazioni di confronto (<, >, =)
- operazioni di trasferimento (memoria ⟷ accumulatore)
- operazioni di controllo

Le istruzioni elementari richiedono un tempo di esecuzione costante

Costo Computazionale di un algoritmo

TEMPO → # istruzioni elementari

SPAZIO → # celle di memoria utilizzate durante l'esecuzione
(senza contare le celle occupate dai dati di input)

↳ si esprimono come funzioni matematiche della dimensione dei dati di input (ISTANZA DI INPUT)

↳ # di bit da servono per scrivere in i dati di input
(o una misura equivalente)

→ valutazione del tasso di crescita

Analisi di un algoritmo

CASO PESSIMO

A: algoritmo I: istanza di input $|I|$ = dim. input

$$\max_{I,\ |I|=n} \{T_A(n)\}$$

Limite superiore per il costo in tempo di A
(valido per tutte le istanze di dim. n)

CASO OTTIMO

$$\min_{I,\ |I|=n} \{T_A(n)\}$$

CASO MEDIO

Costo medio sulle istanze di dim. n

Insertion Sort

PROBLEMA

Input: array a di n numeri interi

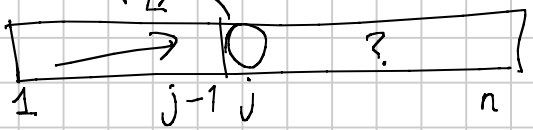
Output: array ordinato

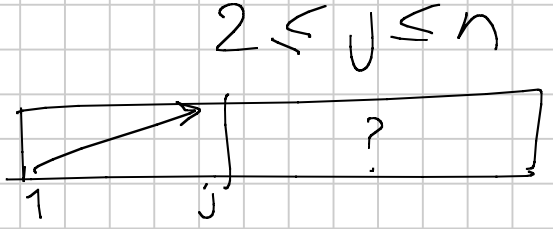
$$a[1] \leq a[2] \leq \dots \leq a[n]$$

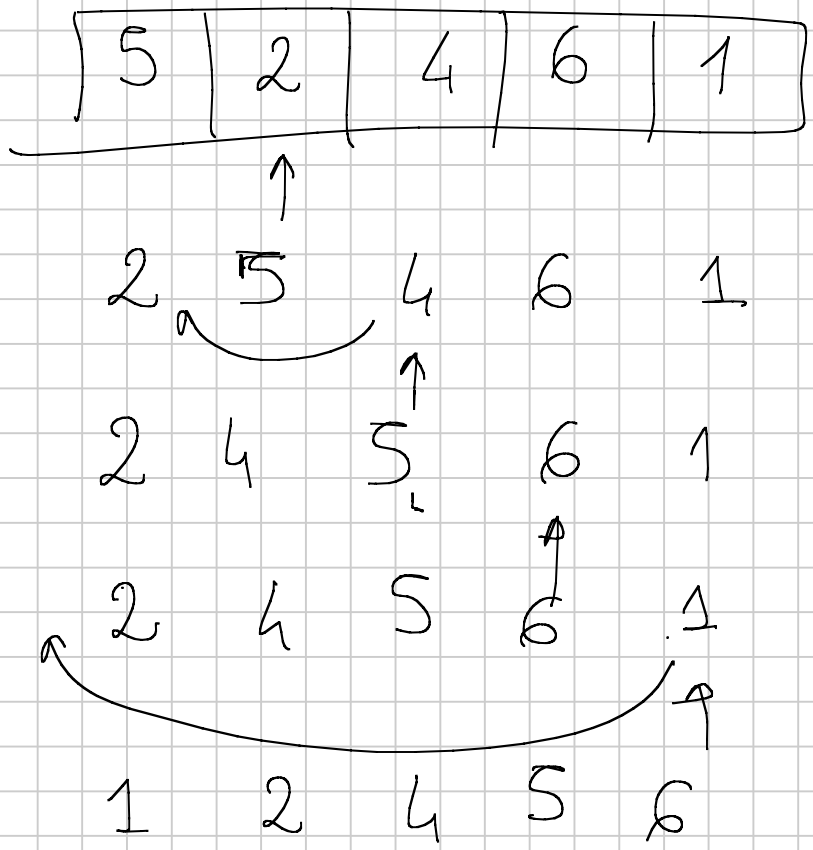
Incrementale

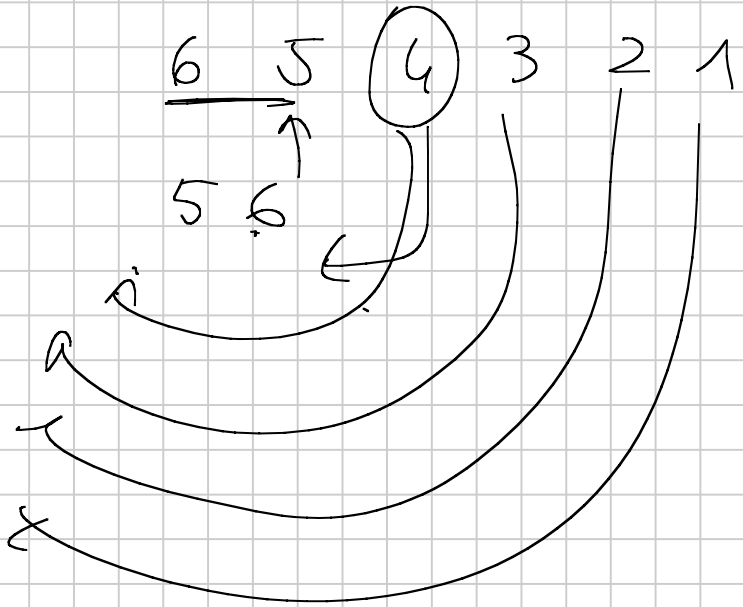
$(n-1)$ passi

j-esimo passo






5 2 4 6 1
2 5 4 6 1
2 4 5 6 1
2 4 5 6 1
1 2 4 5 6


6 5 4 3 2 1
5 6

InsertionSort(a)

```
for j=2 to n {
    k=a[j];
    i=j-1
    while (i>0 && a[i]>k) {
        a[i+1]=a[i]
        i--;
    }
    a[i+1]=k;
}
```

Costo	# volte
c_1	n
c_2	$n-1$
c_3	$n-1$
c_4	$\sum_{j=2}^{n} t_j$
c_5	$\sum_{j=2}^{n} (t_j - 1)$
c_6	$=$
c_7	$n-1$

t_j = # volte che si valuta la guardia del while nella j-esima iterazione del for

Dimostrazione di Correttezza

tecnica dell'invariante di ciclo

Invariante:

all'inizio di ogni iterazione del ciclo for il sottoarray $A[1 \ldots j-1]$ è ORDINATO e contiene gli stessi elementi che erano originariamente in $A[1 \ldots j-1]$

PASSI della DIMOSTRAZIONE

1. **Inizializzazione** — l'invariante è vera prima della prima iterazione del ciclo
   ↳ CASO BASE (INDUZIONE)
2. **Conservazione** — ↳ PASSO INDUTTIVO — l'invariante rimane vera prima della successiva iterazione del ciclo

(3) CONCLUSIONE

validazione dell'invariante a fine ciclo
↓
correttezza dell'algoritmo

1. $j = 2$ $a[1]$ ✓
2. ok per ispezione diretta del codice
3. alla fine del ciclo for $j = n+1$
   $a[1 \ldots n]$ è ordinato e contiene gli elementi che erano originariamente in $a(1 \ldots n)$

CORRETTO!

Analisi di complessità

I a[1...n] $|I| = n$

$$T(n) = c_1 \cdot n + (c_2 + c_3)(n-1) + c_4 \sum_{j=2}^{n} t_j + (c_5 + c_6) \sum_{j=2}^{n} (t_j - 1) + c_7 (n-1)$$

CASO OTTIMO (array è già ordinato)

↳ guardia del while esce falsa

$\forall j \quad t_j = 1$

$$T(n) = c_1 n + (c_2 + c_3 + c_7)(n-1) + c_4 \sum_{j=2}^{n} 1 = c_1 n + (c_2 + c_3 + c_7)(n-1) + c_4 (n-1)$$

FUNZIONE LINEARE

CASO PESSIMO array ordinato →

$t_j = j$

$$T(n) = c_1 n + (c_2+c_3+c_7)(n-1) + c_4 \sum_{j=2}^{n} j + (c_5+c_6)\sum_{j=2}^{n}(j-1)$$

$$= c_1 n + (c_2+c_3+c_7)(n-1) + c_4 \cdot \left(\frac{n(n+1)}{2} - 1\right) + (c_5+c_6)\frac{n(n-1)}{2}$$

↳ cresce come una funzione quadratica di n

cresce come n^2

CASO MEDIO

$t_j \simeq \frac{j}{2}$

$T(n)$ cresce come n^2

Conteggio dei confronti

$$C(n) = \sum_{j=2}^{n} d_j$$

d_j = # confronti durante la j-esima iterazione del for

CASO OTTIMO

$d_j = 1 \quad \forall j$

$$C(n) = \sum_{j=2}^{n} 1 = n-1$$

CASO PESSIMO

$d_j = j-1 \quad \forall j$

$$C(n) = \sum_{j=2}^{n} (j-1) = 1+2+3+\dots+(n-1) = \frac{n(n-1)}{2} = \binom{n}{2}$$

↳ fa tutti i confronti possibili !!!

Caso medio $d_j \simeq \frac{j}{2}$

$$C(n) \simeq \frac{1}{2}\binom{n}{2}$$