Algoritmi e Strutture Dati
8 crediti

Calendario: 2 Marzo – 12 Giugno
Aula: LuM250
Orario: Mer, Gio, Ven 15.30-17.30

Numero crediti = 8 (~ 64 ore)
~ 48 ore di teoria, ~16 ore di esercizi

Modalità d'Esame

a. **Prova scritta**
   (5 appelli - indispensabile iscriversi nella lista di esame che verrà attivata su UNIWEB)

b. **Registrazione**, con possibile colloquio
   (discussione dello scritto con qualche domanda di teoria)

Materiale didattico

Testo: Introduzione agli Algoritmi e Strutture Dati (3° ed). T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein. McGraw-Hill.

Trad. di: Introduction to Algorithms and Data Structures (3° ed). T.H.Cormen, C.E.Leiserson, R.L.Rivest, C.Stein. MIT Press.

Pagina del corso con altro materiale (note, link, ecc.:

www.math.unipd.it/~baldan/Algoritmi

Programma

Le prime 5 parti del testo (con qualche omissione):

I. Fondamenti: notazione per gli algoritmi e primi esempi di algoritmi e di analisi degli algoritmi

II. Ordinamento e statistiche d'ordine

III. Strutture dati fondamentali

IV. Tecniche avanzate di progettazione ed analisi degli algoritmi

V. Strutture dati avanzate

INTRODUZIONE

I problemi computazionali, gli algoritmi che li risolvono e le tecniche per sviluppare tali algoritmi

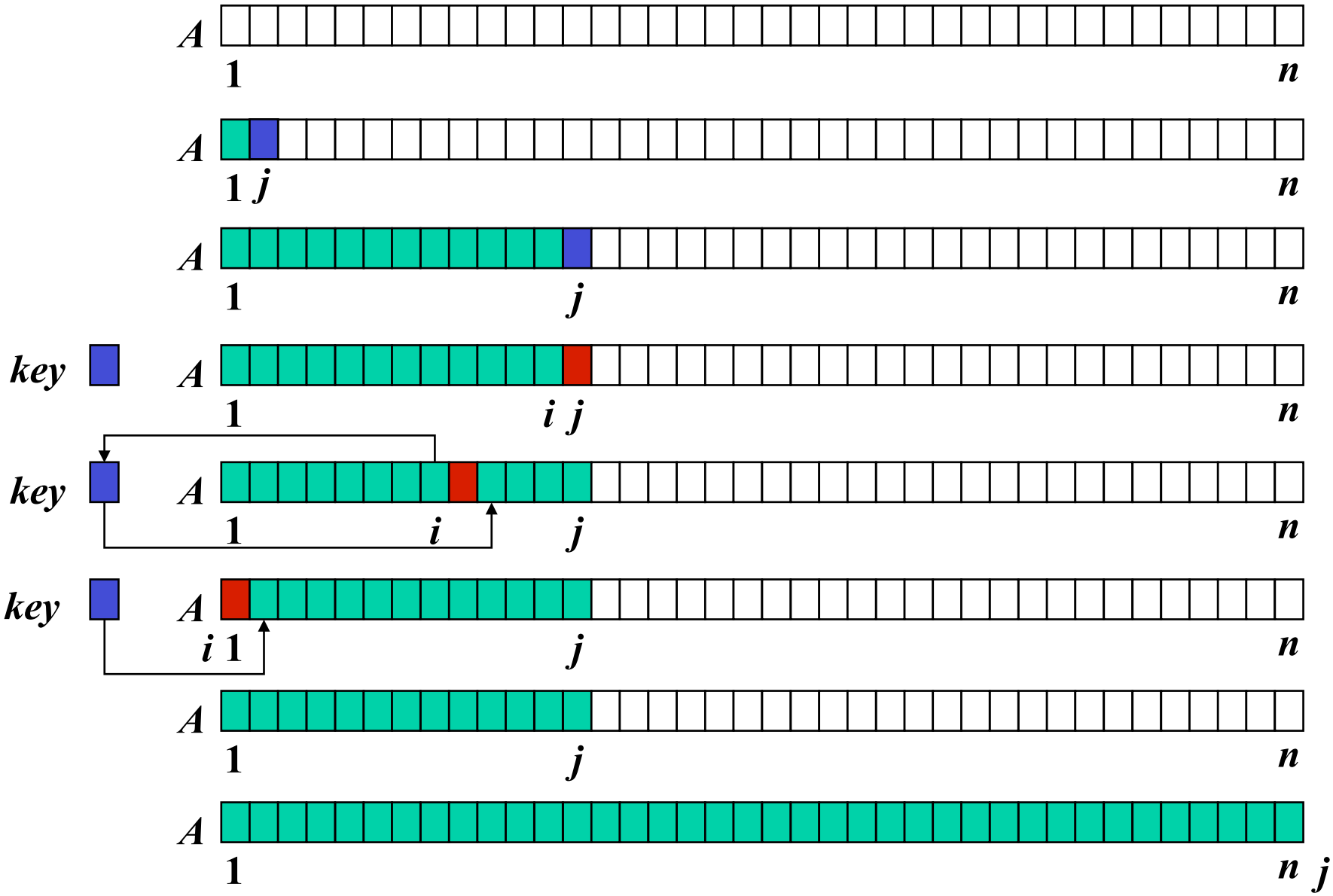
Esempio1: problema dell'ordinamento

Input: $a_1, a_2, \ldots, a_n$

Output: $a'_1, a'_2, \ldots, a'_n$ permutazione (riarrangiamento) di $a_1, a_2, \ldots, a_n$ tale che $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

TECNICA INCREMENTALE

Soluzione1: Algoritmo Insertion-Sort.



```
Insertion-Sort(A)
    n = A.length
    for j = 2 to n
        key = A[j]
        // inseriamo A[j] nella sequenza
        // ordinata A[1..j-1]
        i = j - 1
        while i > 0 and A[i] > key
            A[i+1] = A[i]
            i = i – 1
        A[i+1] = key
```

```
Insertion-Sort(A)

n = A.length
for j = 2 to n
    key = A[j]
    // inseriamo A[j] nella
    // sequenza ordinata
    // A[1..j-1]
    i = j – 1
    while i > 0 and A[i] > key
        A[i+1] = A[i]
        i = i – 1
    A[i+1] = key
```

```
void Insertion-Sort(vector<T> A)
{
 int i, j, n = A.size();  T key;
 for (j = 1; j < n; j++)
  {
   key = A[j];
   // inseriamo A[j] nella
   // sequenza ordinata
   // A[0..j-1]
   i = j – 1;
   while (i >= 0 && A[i] > key)
    {
     A[i+1] = A[i];
     i--;
    }
   A[i+1] = key;
  }
}
```

A								*key*
5	2	8	4	7	1	3	6	
5	#	8	4	7	1	3	6	2
#	5	8	4	7	1	3	6	
2	5	8	4	7	1	3	6	
2	5	#	4	7	1	3	6	8
2	5	#	4	7	1	3	6	
2	5	8	4	7	1	3	6	
2	5	8	#	7	1	3	6	4
2	#	5	8	7	1	3	6	
2	4	5	8	7	1	3	6	
2	4	5	8	#	1	3	6	7
2	4	5	#	8	1	3	6	
2	4	5	7	8	1	3	6	

```
Insertion-Sort(A)
  n = A.length
  for j = 2 to n
    // inserisce A[j] nella
    // sequenza ordinata
    // A[1..j-1]
    key = A[j]
    i = j – 1
    while i > 0 and A[i] > key
      A[i+1] = A[i]
      i = i – 1
    A[i+1] = key
```

A								*key*
2	4	5	7	8	1	3	6	
2	4	5	7	8	#	3	6	1
#	2	4	5	7	8	3	6	
1	2	4	5	7	8	3	6	
1	2	4	5	7	8	#	6	3
1	2	#	4	5	7	8	6	
1	2	3	4	5	7	8	6	
1	2	3	4	5	7	8	#	6
1	2	3	4	5	#	7	8	
1	2	3	4	5	6	7	8	

```
Insertion-Sort(A)
  n = A.lenght
  for j = 2 to n
    // inserisce A[j] nella
    // sequenza ordinata
    // A[1..j-1]
    key = A[j]
    i = j – 1
    while i > 0 and A[i] > key
      A[i+1] = A[i]
      i = i – 1
    A[i+1] = key
```

Analisi di Insertion-Sort: correttezza

Insertion-Sort(A)

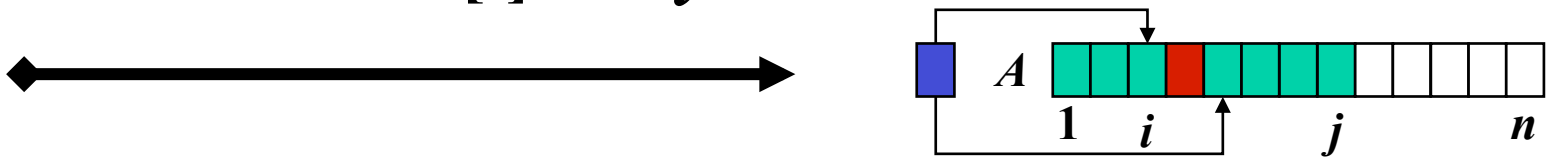
$n = A.length$

for $j = 2$ **to** n

$key = A[j]$

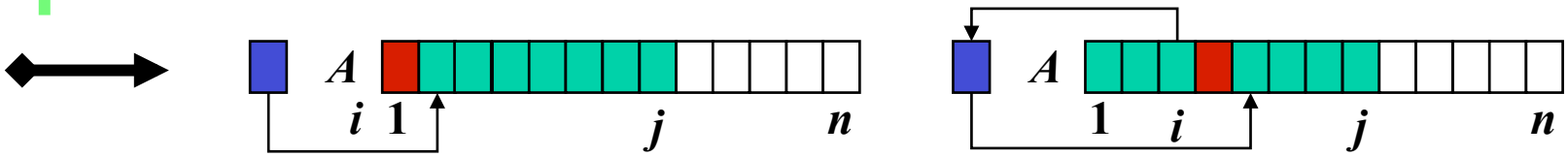
$i = j - 1$

while $i > 0$ **and** $A[i] > key$



$A[i+1] = A[i]$

$i = i - 1$



$A[i+1] = key$

Analisi di Insertion-Sort: complessità

Insertion-Sort(*A*)	//	c_0
$n = A.length$	//	c_1
for $j = 2$ **to** n	//	$c_2 n$
$key = A[j]$	//	$c_3(n-1)$
$i = j - 1$	//	$c_4(n-1)$
while $i > 0$ **and** $A[i] > key$	//	$c_5 \sum_{j=2}^{n}(t_j + 1)$
$A[i+1] = A[i]$	//	$c_6 \sum_{j=2}^{n} t_j$
$i = i - 1$	//	$c_7 \sum_{j=2}^{n} t_j$
$A[i+1] = key$	//	$c_8(n-1)$

$$0 \le t_j < j$$

$$T^{IS}(n) = c_0 + c_1 + c_2 n + (c_3 + c_4 + c_8)(n-1) + c_5 \sum_{j=2}^{n}(t_j + 1) + (c_6 + c_7)\sum_{j=2}^{n} t_j$$

caso migliore: $t_j = 0$ $\boxed{0 \le t_j < j}$

$$T_{\min}^{IS}(n) = c_0 + c_1 + c_2 n + (c_3 + c_4 + c_8)(n-1)$$
$$+ c_5 \sum_{j=2}^{n} 1 + (c_6 + c_7) \sum_{j=2}^{n} 0$$

$$T_{\min}^{IS}(n) = (c_2 + c_3 + c_4 + c_5 + c_8)n + (c_0 + c_1 - c_3 - c_4 - c_5 - c_8)$$
$$= bn + a$$

caso peggiore: $t_j = j - 1$ $\boxed{0 \le t_j < j}$

$$\boxed{\begin{aligned} T_{\max}^{IS}(n) &= c_0 + c_1 + c_2 n + (c_3 + c_4 + c_8)(n-1) \\ &\quad + c_5 \sum_{j=2}^{n} j + (c_6 + c_7) \sum_{j=2}^{n} (j-1) \end{aligned}}$$

$$\begin{aligned} T_{\max}^{IS}(n) &= \frac{1}{2}(c_5 + c_6 + c_7)n^2 \\ &\quad + (c_2 + c_3 + c_4 + \frac{1}{2}c_5 - \frac{1}{2}c_6 - \frac{1}{2}c_7 + c_8)n \\ &\quad + (c_0 + c_1 - c_3 - c_4 - c_8) \\ &= c' n^2 + b' n + a' \end{aligned}$$

caso medio: $t_j = \frac{j-1}{2}$ $\boxed{0 \le t_j < j}$

$$T_{\text{med}}^{IS}(n) = c_0 + c_1 + c_2 n + (c_3 + c_4 + c_8)(n-1)$$

$$+ c_5 \sum_{j=2}^{n} \frac{j+1}{2} + (c_6 + c_7) \sum_{j=2}^{n} \frac{j-1}{2}$$

$$\boxed{\begin{aligned} T_{\text{med}}^{IS}(n) &= \frac{1}{4}(c_5 + c_6 + c_7)n^2 \\ &+ (c_2 + c_3 + c_4 + \frac{3}{4}c_5 - \frac{1}{4}c_6 - \frac{1}{4}c_7 + c_8)n \\ &+ (c_1 - c_3 - c_4 - c_5 - c_8) \\ &= c''n^2 + b''n + a'' \end{aligned}}$$