Partizione di un insieme di interi

Esercizio Programmazione Dinamica

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Partizione di un insieme di interi

- Supponiamo di avere due supporti di capacità s byte ciascuno, e n file indivisibili grandi in totale 2s byte

- Problema: è possibile dividere gli n file in due gruppi di s byte ciascuno?
  - Problema della partizione

Formulazione generale della partizione (***partition***):

- Insieme di interi positivi $A = \{a_0, a_1, .., a_{n-1}\}$ t.c. $\sum_{i=0,1,..,n-1} a_i = 2s$
- Esiste sottoinsieme $B = \{b_0, b_1, .., b_{k-1}\} \subseteq A$ t.c. $\sum_{j=0,1,..,k-1} b_j = s$?
  - Soluzione banale: tempo esponenziale $O(2^n\ n)$ se generiamo tutti i sottoinsiemi
  - Domanda: perché la complessità è $O(2^n\ n)$

Programmazione dinamica

Seguiamo le solite regole:

1. definiamo per $0 \leq i \leq n$ e $0 \leq j \leq s$

   $T(i,j)$ = true sse
   esiste $S \subseteq \{a_0, a_1, .., a_{i-1}\}$ t.c. $\sum_{y \in S} y = j$

2. Domanda: come possiamo formulare il problema originario??

Programmazione dinamica

Seguiamo le solite regole:

1. definiamo per $0 \leq i \leq n$ e $0 \leq j \leq s$

   $T(i,j)$ = true sse
   esiste $S \subseteq \{a_0, a_1, .., a_{i-1}\}$ t.c. $\sum_{y \in S} y = j$

2. Domanda: come possiamo formulare il problema originario??
   1. T(n,s)

Programmazione dinamica

Seguiamo le solite regole:

1. definiamo per $0 \leq i \leq n$ e $0 \leq j \leq s$

   $T(i,j)$ = true sse
   esiste $S \subseteq \{a_0, a_1, .., a_{i-1}\}$ t.c. $\sum_{y \in S} y = j$

2. Domanda: come possiamo formulare il problema originario??
   1. $T(n,s)$

[convenzione: $\{a_0, a_1, ..., a_{-1}\}$ = insieme vuoto]

Programmazione Dinamica

- Domanda: quali sono i sotto-problemi elementari (quelli per cui possiamo dare subito la soluzione)?

Programmazione Dinamica

- Domanda: quali sono i sotto-problemi elementari (quelli per cui possiamo dare subito la soluzione)?
  - T(0,0)=true
  - T(0,j)=false, $0<j\leq s$

Programmazione Dinamica

- Domanda: quali sono i sotto-problemi elementari (quelli per cui possiamo dare subito la soluzione)?
  - T(0,0)=true
  - T(0,j)=false, $0<j\leq s$

- Cerchiamo ora di definire il caso generale T(i,j) in termini dei casi aventi valori i e j più piccoli (programmazione dinamica….)
  - Seguiamo l'approccio seguito per il problema LCS
    - a_{i-1} fa parte della soluzione immediatamente più piccola o no?

Programmazione Dinamica

- Cerchiamo ora di definire il caso generale T(i,j) in termini dei casi aventi valori i e j più piccoli (programmazione dinamica….)
  - Seguiamo l'approccio seguito per il problema LCS
    - a_{i-1} fa parte della soluzione o no?
    - Provate ad analizzare i casi

Programmazione Dinamica

- Cerchiamo ora di definire il caso generale T(i,j) in termini dei casi aventi valori i e j più piccoli (programmazione dinamica….)
  - Seguiamo l'approccio seguito per il problema LCS
    - a_{i-1} fa parte della soluzione o no?
    - Provate ad analizzare i casi

esiste già un sotto-insieme (che non contiene a_{i-1}) la cui somma è j

Programmazione Dinamica

- Cerchiamo ora di definire il caso generale T(i,j) in termini dei casi aventi valori i e j più piccoli (programmazione dinamica….)
  - Seguiamo l'approccio seguito per il problema LCS
    - a_{i-1} fa parte della soluzione o no?
    - Provate ad analizzare i casi

esiste già un sotto-insieme (che non contiene a_{i-1}) la cui somma è j

non esiste già un sotto-insieme la cui somma è j, e che permette l'aggiunta di a_{i-1}

Programmazione Dinamica

- Cerchiamo ora di definire il caso generale T(i,j) in termini dei casi aventi valori i e j più piccoli (programmazione dinamica….)
  - Seguiamo l'approccio seguito per il problema LCS
    - a_{i-1} fa parte della soluzione o no?
    - Provate ad analizzare i casi

esiste già un sotto-insieme (che non contiene a_{i-1}) la cui somma è j

non esiste già un sotto-insieme la cui somma è j, e che permette l'aggiunta di a_{i-1}

else

Formulazione ricorsiva

$$T(i,j) = \begin{cases} \texttt{true} & \text{se } i = 0 \text{ e } j = 0 \\ \texttt{true} & \text{se } i > 0 \text{ e } T(i-1,j) = \texttt{true} \\ \texttt{true} & \text{se } i > 0, j \geqslant a_{i-1} \text{ e } T(i-1,j-a_{i-1}) = \texttt{true} \\ \texttt{false} & \text{altrimenti} \end{cases}$$

Formulazione ricorsiva

$$T(i,j) = \begin{cases} \texttt{true} & \text{se } i = 0 \text{ e } j = 0 \\ \texttt{true} & \text{se } i > 0 \text{ e } T(i-1,j) = \texttt{true} \\ \texttt{true} & \text{se } i > 0, j \geqslant a_{i-1} \text{ e } T(i-1,j-a_{i-1}) = \texttt{true} \\ \texttt{false} & \text{altrimenti} \end{cases}$$

se l'insieme è vuoto allora la somma j=0 è l'unica possibile

Formulazione ricorsiva

$$T(i,j) = \begin{cases} \texttt{true} & \text{se } i = 0 \text{ e } j = 0 \\ \texttt{true} & \text{se } i > 0 \text{ e } T(i-1,j) = \texttt{true} \\ \texttt{true} & \text{se } i > 0, j \geqslant a_{i-1} \text{ e } T(i-1,j-a_{i-1}) = \texttt{true} \\ \texttt{false} & \text{altrimenti} \end{cases}$$

Caso 1. Il sottoinsieme di $\{a_0, a_1, .., a_{i-1}\}$ la cui somma è pari a j **non** include a_{i-1}: tale insieme è dunque sottoinsieme anche di $\{a_0, a_1, .., a_{i-2}\}$ e vale T(i-1,j)=true

Formulazione ricorsiva

$$T(i,j) = \begin{cases} \texttt{true} & \text{se } i = 0 \text{ e } j = 0 \\ \texttt{true} & \text{se } i > 0 \text{ e } T(i-1,j) = \texttt{true} \\ \texttt{true} & \text{se } i > 0, j \geqslant a_{i-1} \text{ e } T(i-1,j-a_{i-1}) = \texttt{true} \\ \texttt{false} & \text{altrimenti} \end{cases}$$

Caso 2. Il sottoinsieme di $\{a_0, a_1, .., a_{i-1}\}$ la cui somma è pari a j **include** a_{i-1}: esiste quindi in $\{a_0, a_1, .., a_{i-2}\}$ un sottoinsieme di somma pari a $j - a_{i-1}$ (ovviamente se $j \geq a_{i-1}$) e vale $T(i-1,j-a_{i-1})$=true

Programmazione dinamica per partizione

Tabella di programmazione dinamica di taglia (n+1)*(s+1)

parti[i][j] = T(i,j)

Ordine riempimento: per righe

Pseudocodice

```
1  Partizione( a ):          ⟨pre: a è un array di n interi positivi
2    FOR (i = 0; i <= n; i = i+1)
3      FOR (j = 0; j <= s; j = j+1) {
4        parti[i][j] = FALSE;
5      }
6    parti[0][0] = TRUE;
7    FOR (i = 1; i <= n; i = i+1)
8      FOR (j = 0; j <= s; j = j+1) {
9        IF (parti[i-1][j]) {
10         parti[i][j] = TRUE;
11       }
12       IF (j >= a[i-1] && parti[i-1][ j-a[i-1] ]) {
13         parti[i][j] = TRUE;
14       }
15     }
16   RETURN parti[n][s];
```

$$T(i,j) = \begin{cases} \texttt{true} & \text{se } i = 0 \text{ e } j = 0 \\ \texttt{true} & \text{se } i > 0 \text{ e } T(i-1,j) = \texttt{true} \\ \texttt{true} & \text{se } i > 0,\ j \geqslant a_{i-1} \text{ e } T(i-1,j-a_{i-1}) = \texttt{true} \\ \texttt{false} & \text{altrimenti} \end{cases}$$

Analisi di complessità per partizione

- Tempo O(ns)

- Spazio O(ns): O(s) usando solo le ultime due righe compilate iterativamente

Nota: in generale è possibile ottenere delle prestazioni simili a quelle della programmazione dinamica usando **ricorsione + memoization**

Ricorsione + memoization

```
1  PartizioneMemoization( ):
2    FOR (j = 0; j <= s; j = j+1)
3      parti[0][j] = FALSE;
4    parti[0][0] = TRUE;
5    FOR (i = 1; i <= n; i = i+1)
6      FOR (j = 0; j <= s; j = j+1) {
7        parti[i][j] = ∅;
8      }
9    RETURN PartizioneRicNota( n, s );
```

```
1  PartizioneRicNota( i, j ):                    ⟨pre: 0 ⩽ i ⩽ n, 0
2    IF (parti[i][j] == ∅) {
3      parti[i][j] = PartizioneRicNota( i-1, j );
4      IF (!parti[i][j] && (j >= a[i-1])) {
5        parti[i][j] = PartizioneRicNota( i-1, j-a[i-1] );
6      }
7    }
8    RETURN parti[i][j];
```

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??
  - Sequenza ottima per la moltiplicazione di matrici

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??
  - Sequenza ottima per la moltiplicazione di matrici
    - NO
  - LCS

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??
  - Sequenza ottima per la moltiplicazione di matrici
    - NO
  - LCS
    - NO
  - Partizione di n interi di somma totale 2s

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??
  - Sequenza ottima per la moltiplicazione di matrici
    - NO
  - LCS
    - NO
  - Partizione di n interi di somma totale 2s
    - SI
  - Zaino 0-1

Algoritmi pseudo-polinomiali

- Test: quali dei seguenti algoritmi sono pseudo-polinomiali e quali no??
  - Sequenza ottima per la moltiplicazione di matrici
    - NO
  - LCS
    - NO
  - Partizione di n interi di somma totale 2s
    - SI
  - Zaino 0-1
    - SI

Edit Distance

Esercizio Programmazione Dinamica

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Confronto di sequenze

Il confronto tra sequenze è fondamentale in campi come la biologia computazionale, l'analisi dei testi e molti altri.

Possibili obiettivi:

- misurare la "similarità" tra le sequenze
  - **allineamento**
- misurare la "diversità" tra le sequenze
  - **distanza di edit**
- trovare parti comuni alle sequenze
  - **pattern discovery**
  - **allineamento locale**

Allineamento tra due sequenze

Allineamento tra due sequenze

Misuriamo la similarità

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Vogliamo trovare **l'allineamento** ottimo fra le due sequenze, cioè quello minimizza la **distanza** (**edit distance**) fra di esse:

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Vogliamo trovare **l'allineamento** ottimo fra le due sequenze, cioè quello minimizza la **distanza** (**edit distance**) fra di esse:

1. Possiamo inserire spazi nelle sequenze (allo scopo di allinearle)

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Vogliamo trovare **l'allineamento** ottimo fra le due sequenze, cioè quello minimizza la **distanza** (**edit distance**) fra di esse:

1. Possiamo inserire spazi nelle sequenze (allo scopo di allinearle)

2. La distanza è la somma delle distanze fra coppie di caratteri allineati:

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Vogliamo trovare **l'allineamento** ottimo fra le due sequenze, cioè quello minimizza la **distanza** (**edit distance**) fra di esse:

1. Possiamo inserire spazi nelle sequenze (allo scopo di allinearle)

2. La distanza è la somma delle distanze fra coppie di caratteri allineati:

   - **caratteri uguali** sono a distanza 0

Allineamento tra due sequenze

Misuriamo la similarità

Distanza di edit tra due sequenze S_1 e S_2:

Vogliamo trovare **l'allineamento** ottimo fra le due sequenze, cioè quello minimizza la **distanza** (**edit distance**) fra di esse:

1. Possiamo inserire spazi nelle sequenze (allo scopo di allinearle)

2. La distanza è la somma delle distanze fra coppie di caratteri allineati:

   - **caratteri uguali** sono a distanza 0

   - **caratteri diversi** o caratteri allineati a spazio hanno distanza 1

Perché?

Molte applicazioni

- Biologia Computazionale:

  …ATGCATACGATCGATT…
  …TGCAATGGCTTAGCTA…

  Specie animali della stessa famiglia hanno DNA simili

  ….oppure biologia evolutiva

Distanza di edit — ALBERO vs LABBRO

A L B E R O

L A B B R O

- Le stringhe hanno già lunghezza uguale, quindi sono già allineate
- **Costo della distanza?**

Distanza di edit — ALBERO vs LABBRO

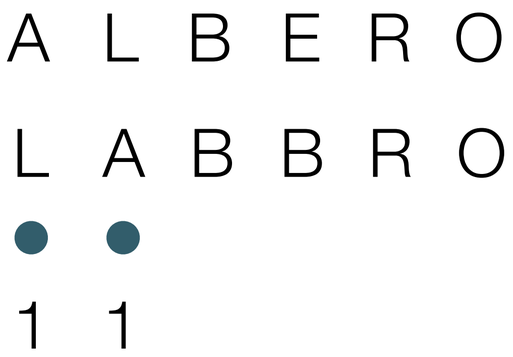
A L B E R O

L A B B R O

1

- Le stringhe hanno già lunghezza uguale, quindi sono già allineate
- **Costo della distanza?**

Distanza di edit — ALBERO vs LABBRO



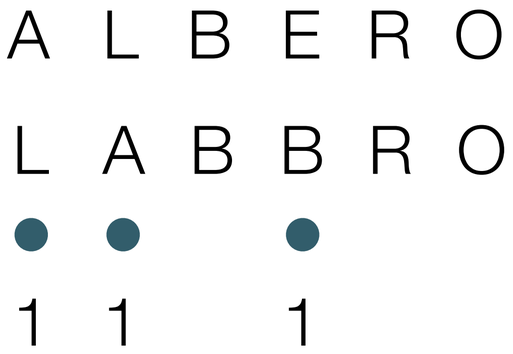
- Le stringhe hanno già lunghezza uguale, quindi sono già allineate
- **Costo della distanza?**

Distanza di edit — ALBERO vs LABBRO

A	L	B	E	R	O
L	A	B	B	R	O
●	●		●		
1	1		1		

- Le stringhe hanno già lunghezza uguale, quindi sono già allineate
- **Costo della distanza?**

Distanza di edit — ALBERO vs LABBRO



- Le stringhe hanno già lunghezza uguale, quindi sono già allineate
- **Costo della distanza?**

la distanza è 3

Distanza di edit — ALBERO vs LABBRO

A L B E R O

L A B B R O

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

A L B E R O

\- L A B B R O

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

A L - B E R O

- L A B B R O

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

A	L	-	B	E	R	O
-	L	A	B	B	R	O
●						
1						

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

	A	L	-	B	E	R	O
-	L	A	B	B	R	O	
●		●					
1		1					

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

A	L	-	B	E	R	O
-	L	A	B	B	R	O
●		●		●		
1		1		1		

- **In alternativa?**

Ricordate la possibilità di usare spazi

Distanza di edit — ALBERO vs LABBRO

A	L	-	B	E	R	O
-	L	A	B	B	R	O
●		●		●		
1		1		1		

- **In alternativa?**

Ricordate la possibilità di usare spazi

la distanza è 3

Formalmente

- X = x1, x2, …., xn e Y = y1, y2, …., ym
- Vogliamo usare Programmazione Dinamica
- **Idee?**
  - **Cosa sono i sotto-problemi?**
  - **Tabella di programmazione dinamica?**

Sotto-problemi

- X = x1, x2, …., xn e Y = y1, y2, …., ym
- Vogliamo usare Programmazione Dinamica
- **Cosa sono i sotto-problemi?**
  - **Prefissi delle due sequenze**
- **Tabella di programmazione dinamica?**

Tabella

- X = x1, x2, …., xn e Y = y1, y2, …., ym
- Vogliamo usare Programmazione Dinamica
- **Tabella di programmazione dinamica?**
  - **Matrice M[i,j] che memorizza la edit distance dei prefissi di x e y lunghi i e j rispettivamente**
  - Quindi le righe della tabella rappresentano i caratteri di X e le colonne quelli di Y

Tabella

- **Tabella di programmazione dinamica?**
  - **Matrice M[i,j] che memorizza la edit distance dei prefissi di x e y lunghi i e j rispettivamente**
  - Quindi le righe della tabella rappresentano i caratteri di X e le colonne quelli di Y

Valori iniziali della matrice?

Tabella

- **Tabella di programmazione dinamica?**
  - **Matrice M[i,j] che memorizza la edit distance dei prefissi di x e y lunghi i e j rispettivamente**
  - Quindi le righe della tabella rappresentano i caratteri di X e le colonne quelli di Y

Valori iniziali della matrice?

M[0, j] = ?
M[i, 0] = ?

Tabella

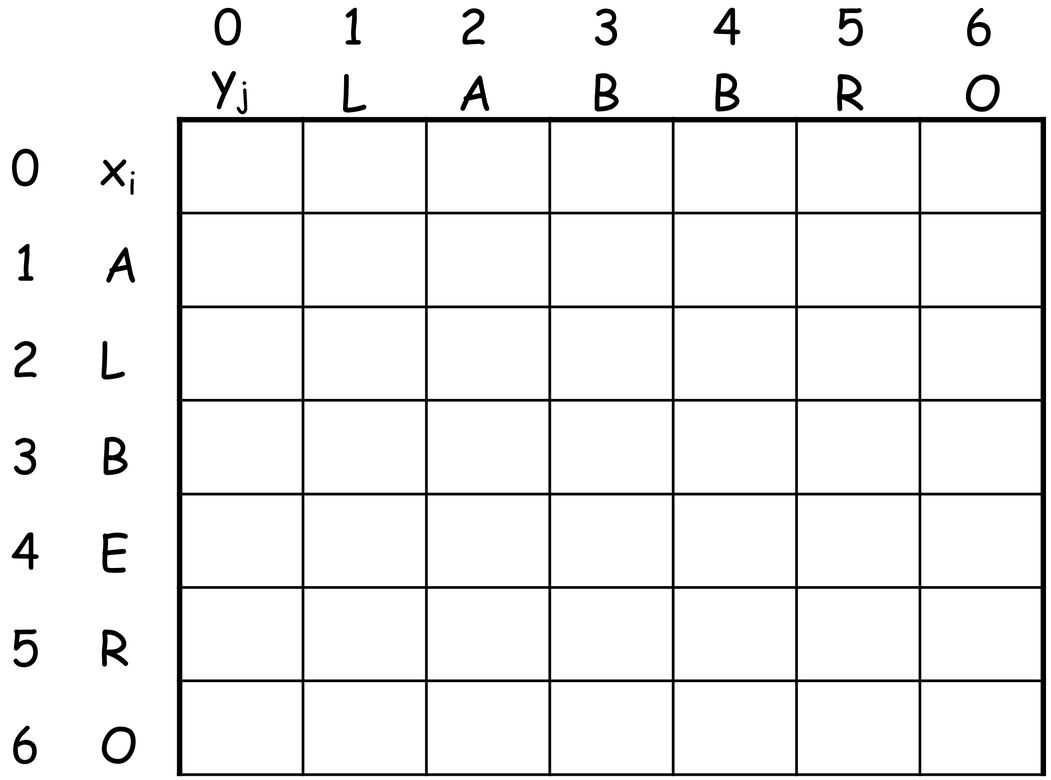
- **Tabella di programmazione dinamica?**
  - **Matrice M[i,j] che memorizza la edit distance dei prefissi di x e y lunghi i e j rispettivamente**
  - Quindi le righe della tabella rappresentano i caratteri di X e le colonne quelli di Y

Valori iniziali della matrice?

M[0, j] = j
M[i, 0] = i

Programmazione Dinamica

- Indichiamo con p(i, j) la distanza locale dei caratteri xi e yj
  - Se xi = yj, **p(i, j) =0**
  - Altrimenti, **p(i, j) = 1**



Programmazione Dinamica

- Indichiamo con p(i, j) la distanza locale dei caratteri xi e yj
  - Se xi = yj, **p(i, j) =0**
  - Altrimenti, **p(i, j) = 1**

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0						
1	A	1						
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica

- Indichiamo con p(i, j) la distanza locale dei caratteri xi e yj
  - Se xi = yj, **p(i, j) =0**
  - Altrimenti, **p(i, j) = 1**

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1						
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

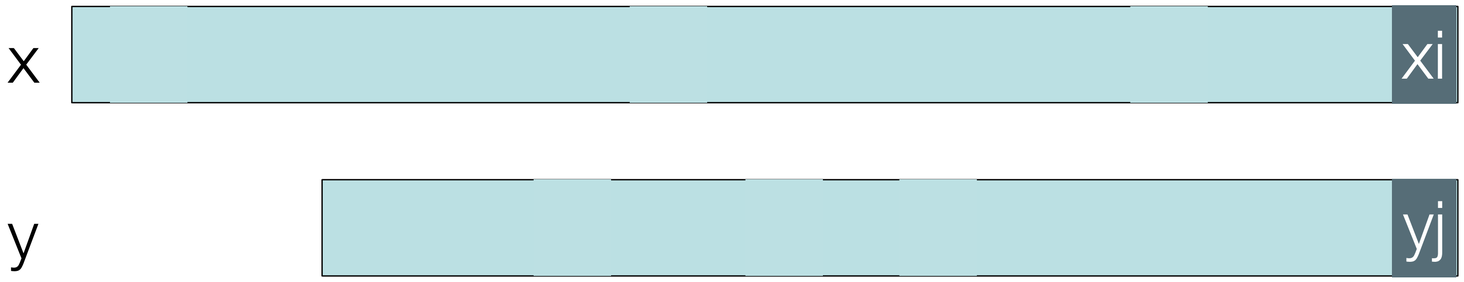
- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1						
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

  Quali possibilità vanno considerate, basandosi sull'ipotesi che conosciamo le edit distance per le sotto-sequenze?

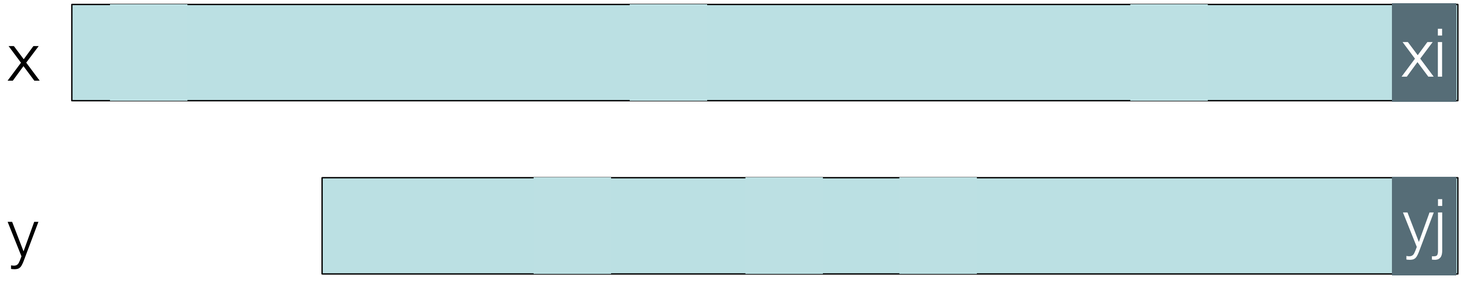


Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

Quali possibilità vanno considerate, basandosi sull'ipotesi che conosciamo le edit distance per le sotto-sequenze?

x1, x2, …., xi **vs** y1, y2, …., yj



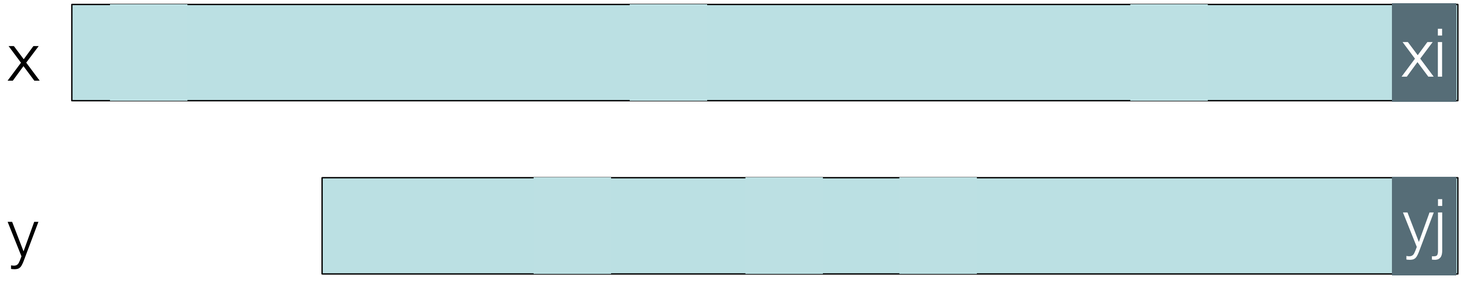
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

Quali possibilità vanno considerate, basandosi sull'ipotesi che conosciamo le edit distance per le sotto-sequenze?

x1, x2, …., xi **vs** y1, y2, …., yj

x1, x2, …., - **vs** y1, y2, …., yj



Programmazione Dinamica — formulazione ricorsiva

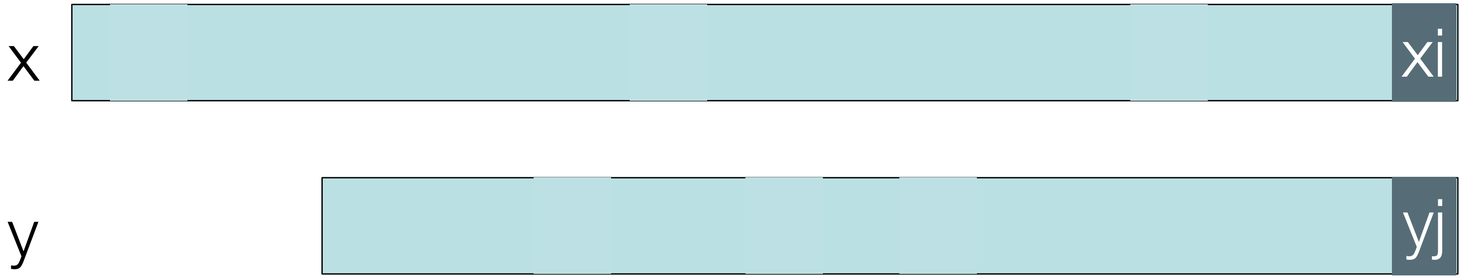
- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

Quali possibilità vanno considerate, basandosi sull'ipotesi che conosciamo le edit distance per le sotto-sequenze?

x1, x2, …., xi **vs** y1, y2, …., yj

x1, x2, …., - **vs** y1, y2, …., yj

x1, x2, …., xi **vs** y1, y2, …., -



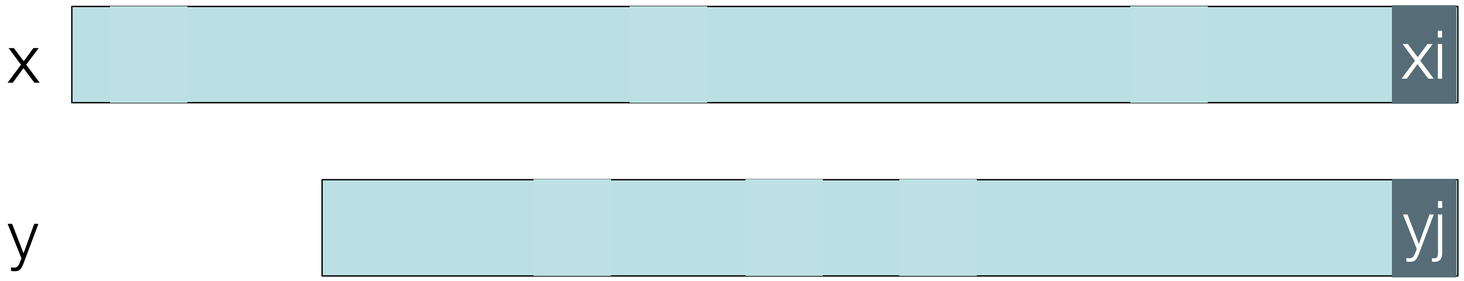
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

x1, x2, …., xi **vs** y1, y2, …., yj → M[?, ?] + ?

x1, x2, …., - **vs** y1, y2, …., yj →

x1, x2, …., xi **vs** y1, y2, …., - →



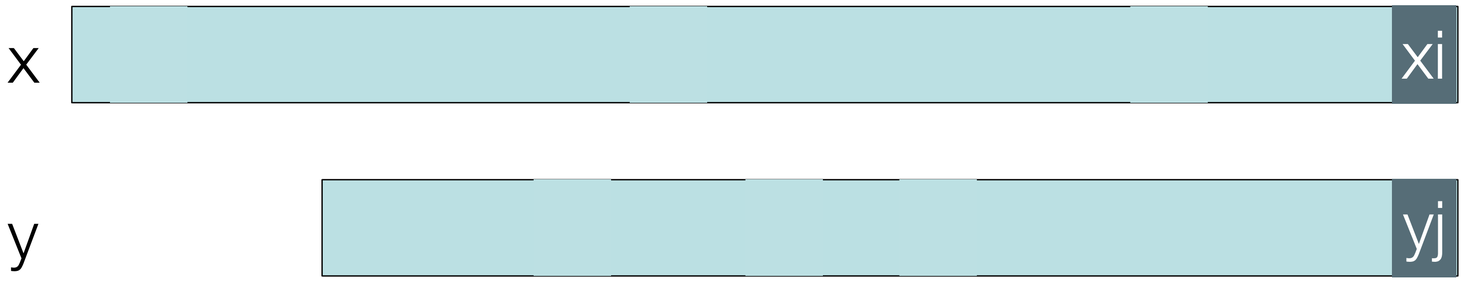
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

x1, x2, …., xi **vs** y1, y2, …., yj → **M[i-1, j-1] + p(i,j)**

x1, x2, …., - **vs** y1, y2, …., yj → M[?, ?] + ?

x1, x2, …., xi **vs** y1, y2, …., - →



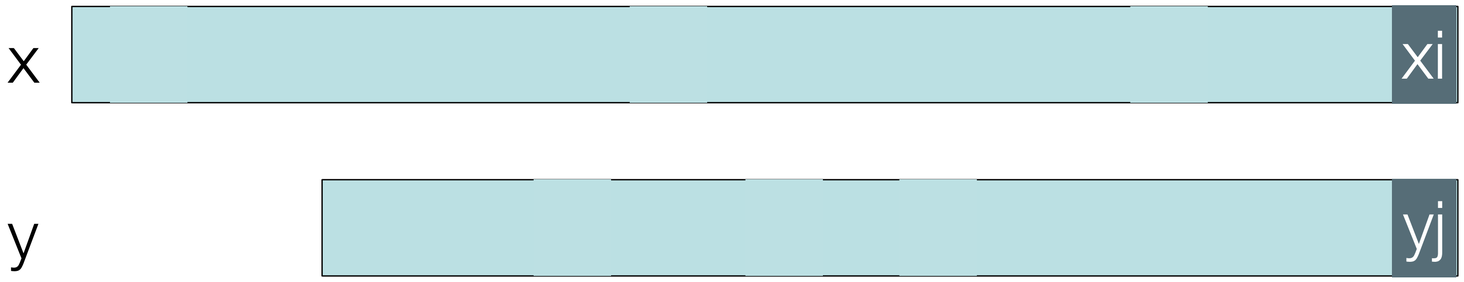
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

x1, x2, …., xi **vs** y1, y2, …., yj → **M[i-1, j-1] + p(i,j)**

x1, x2, …., - **vs** y1, y2, …., yj → **M[i-1, j] + 1**

x1, x2, …., xi **vs** y1, y2, …., - → M[?, ?] + ?



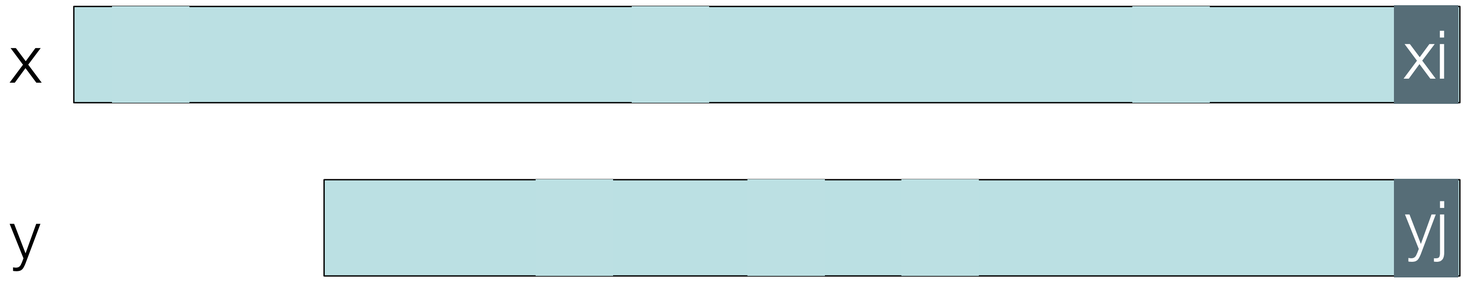
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

x1, x2, …., xi **vs** y1, y2, …., yj → **M[i-1, j-1] + p(i,j)**

x1, x2, …., - **vs** y1, y2, …., yj → **M[i-1, j] + 1**

x1, x2, …., xi **vs** y1, y2, …., - → **M[i, j-1] + 1**



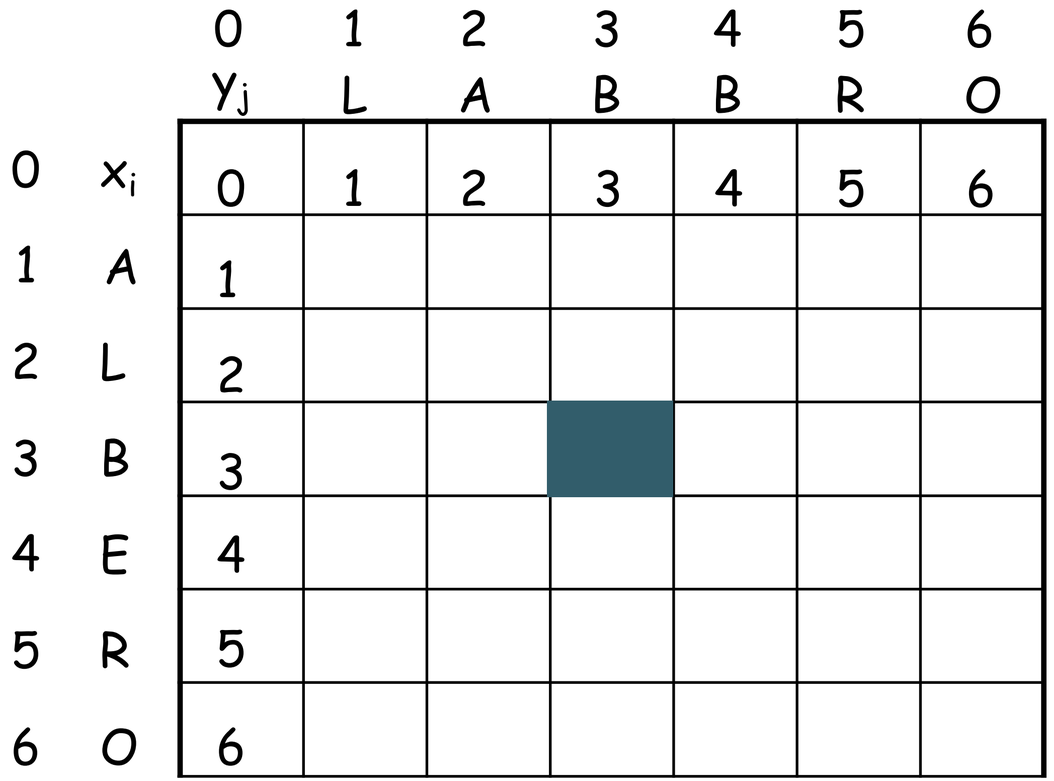
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }



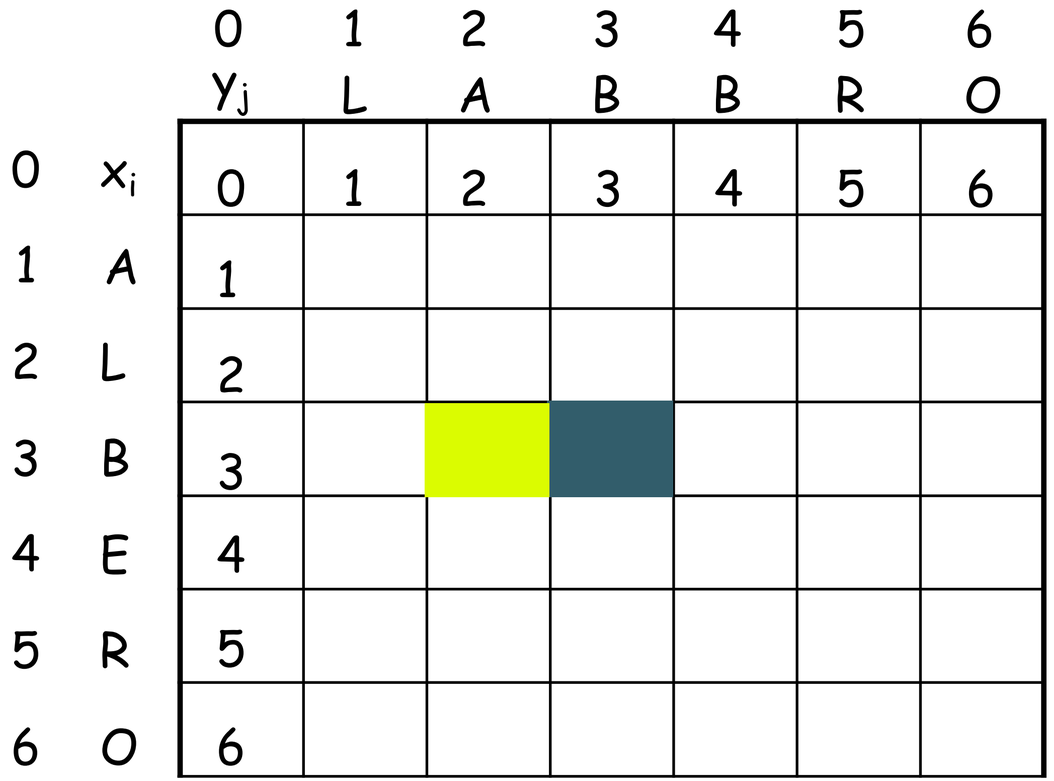
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }



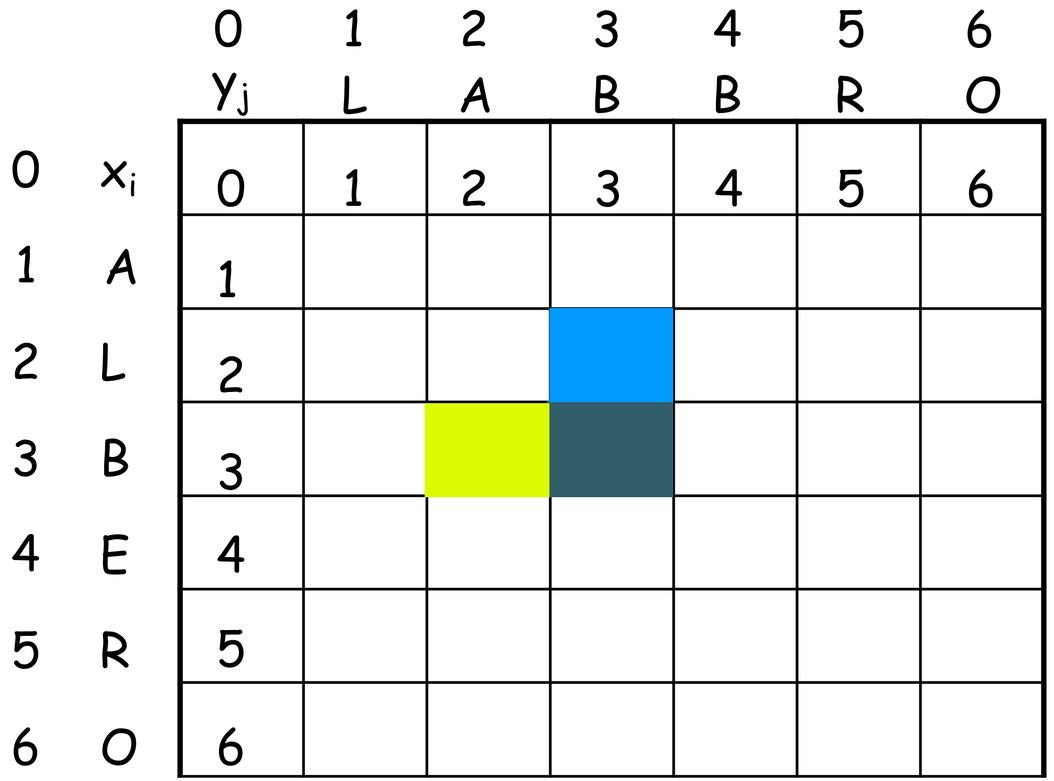
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }



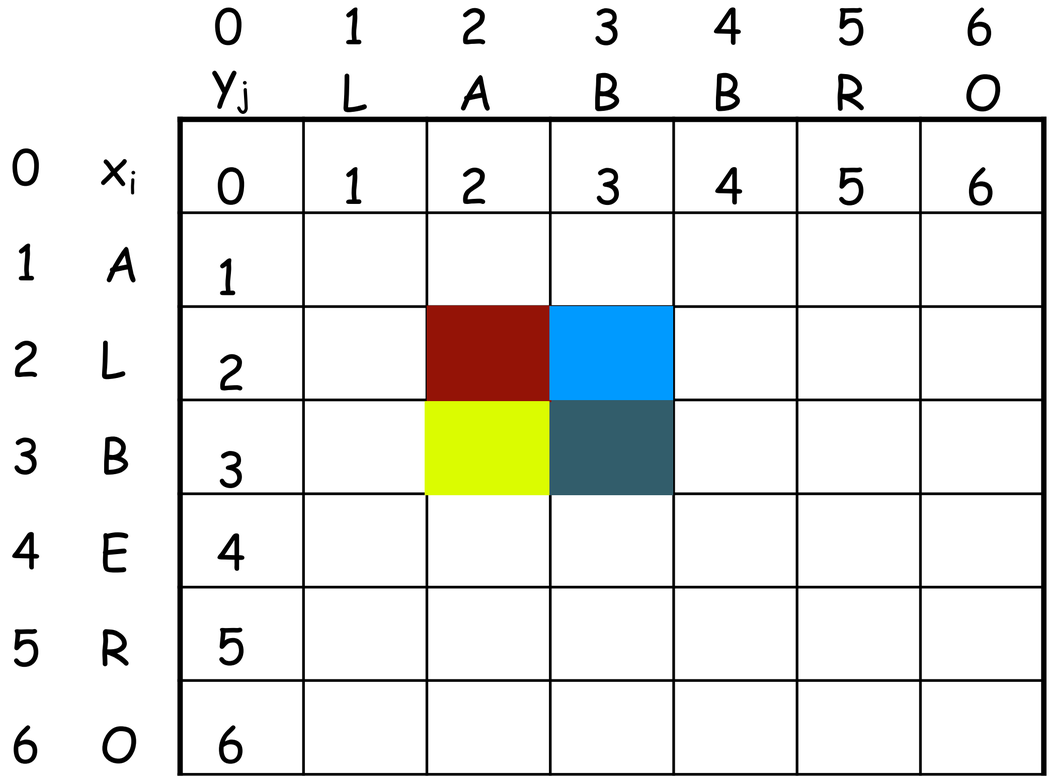
Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }



Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1						
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1					
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1				
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2			
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3		
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2						
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1					
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j)}

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2				
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j)}

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2			
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3		
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3						
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4						
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5						
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6						

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6	5	5	5	5	4	3

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6	5	5	5	5	4	3

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

Complessità: **O(????)**

		0 y_j	1 L	2 A	3 B	4 B	5 R	6 O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6	5	5	5	5	4	3

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

Complessità: **O(nm)**

		0 y_j	1 L	2 A	3 B	4 B	5 R	6 O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6	5	5	5	5	4	3

Programmazione Dinamica — formulazione ricorsiva

- Come possiamo formulare ricorsivamente il calcolo di M[i,j]?

M[i,j] = min { **M[i, j-1] + 1**

M[i-1, j] + 1

M[i-1, j-1] + p(i,j) }

Complessità: **O(nm)**

Dispense per come ricostruire l'allineamento ottimo

		0	1	2	3	4	5	6
		y_j	L	A	B	B	R	O
0	x_i	0	1	2	3	4	5	6
1	A	1	1	1	2	3	4	5
2	L	2	1	2	2	3	4	5
3	B	3	2	2	2	2	3	4
4	E	4	3	3	3	3	3	4
5	R	5	4	4	4	4	3	4
6	O	6	5	5	5	5	4	3

Come passare l’esame!

Esercizio Programmazione Dinamica

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Problema

- Si consideri un **esame** con n problemi, ognuno dei quali vale v[i] punti e richiede t[i] minuti per essere risolto
- Il vostro obiettivo è di passare l'esame con il minimo sforzo possibile
- Trovate quindi un algoritmo che dati **v[1 . . . n]**, **t[1 . . . n]** e un valore **V** che rappresenta la sufficienza, trova il sottoinsieme di problemi che richiede il minor tempo possibile e permette di passare l'esame

Problema

Come strutturare la tabella (e quindi la formulazione ricorsiva)?

- Si consideri un **esame** con n problemi, ognuno dei quali vale v[i] punti e richiede t[i] minuti per essere risolto
- Il vostro obiettivo è di passare l'esame con il minimo sforzo possibile
- Trovate quindi un algoritmo che dati **v[1 . . . n]**, **t[1 . . . n]** e un valore **V** che rappresenta la sufficienza, trova il sottoinsieme di problemi che richiede il minor tempo possibile e permette di passare l'esame

Cose da fare

1. Sia T(i, v) il numero minimo di minuti richiesti per ottenere almeno v risolvendo un qualunque sottoinsieme dei primi i problemi
   - **Scrivere un'espressione ricorsiva** per T(i, v)
2. Scrivere **l'algoritmo** basato su programmazione dinamica
3. Valutare la **complessità** dell'algoritmo proposto

Programmazione Dinamica — Casi Base

- Per ottenere almeno 0 o un voto negativo, posso uscire subito dall'aula:
  - $T[i, v] = 0 : \forall i, \forall v \leq 0$
- Se non ho esercizi da risolvere, ma voglio ottenere $v > 0$, resterò chiuso dentro l'aula in eterno
  - $T[0, v] = ???? : \forall v > 0$

Programmazione Dinamica — Casi Base

- Per ottenere almeno 0 o un voto negativo, posso uscire subito dall'aula:
  - **T[i, v] = 0** : $\forall i, \forall v \leq 0$
- Se non ho esercizi da risolvere, ma voglio ottenere $v > 0$….
  - T[0, v] = ???? : $\forall v > 0$

Programmazione Dinamica — Casi Base

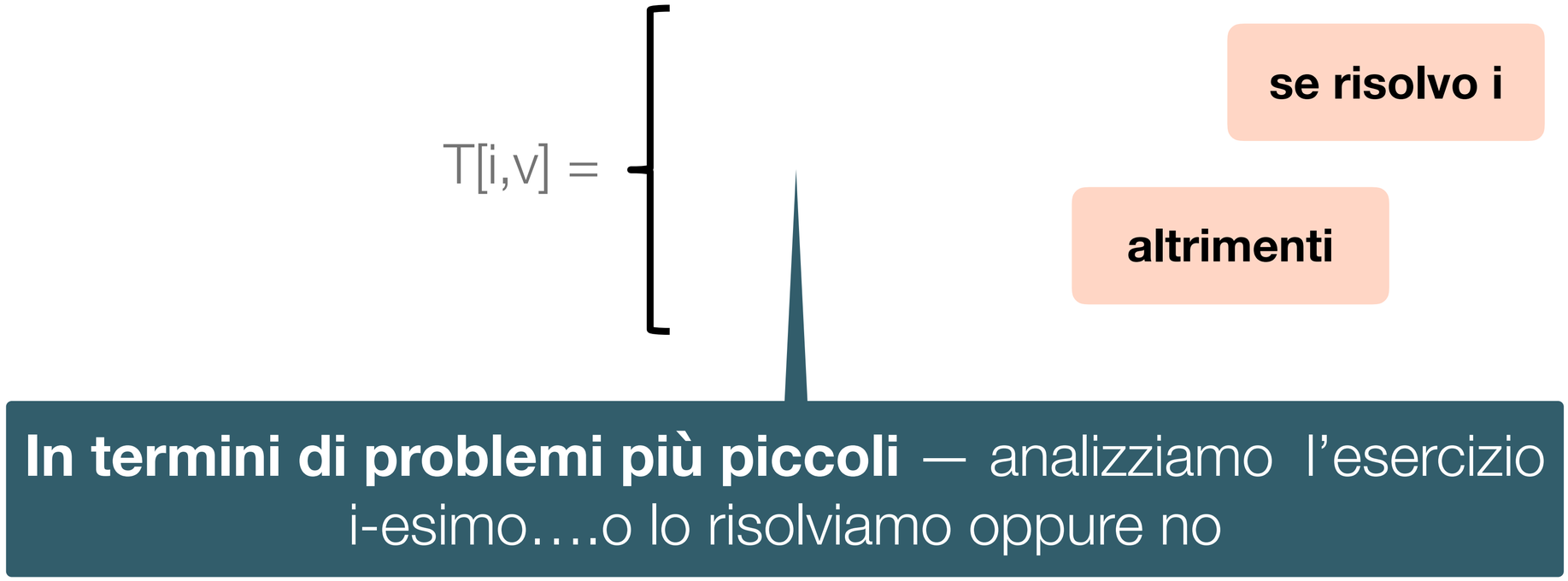
- Per ottenere almeno 0 o un voto negativo, posso uscire subito dall'aula:
  - **T[i, v] = 0** : $\forall i, \forall v \leq 0$
- Se non ho esercizi da risolvere, ma voglio ottenere $v > 0$, resterò chiuso dentro l'aula in eterno:
  - T[0, v] = ???? : $\forall v > 0$

Programmazione Dinamica — Casi Base

- Per ottenere almeno 0 o un voto negativo, posso uscire subito dall'aula:
  - **T[i, v] = 0** : $\forall i, \forall v \leq 0$
- Se non ho esercizi da risolvere, ma voglio ottenere $v > 0$, resterò chiuso dentro l'aula in eterno:
  - **T[0, v] = +∞** : $\forall v > 0$

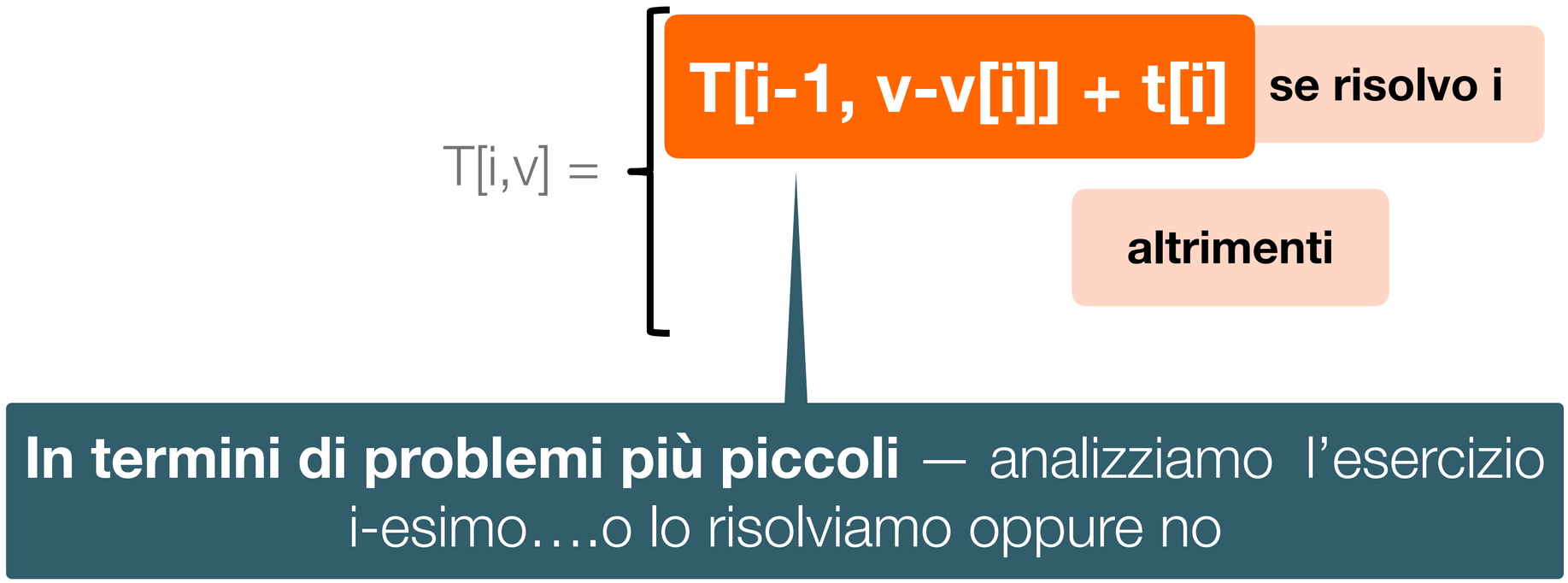
Programmazione Dinamica — formulazione ricorsiva

- Formulazione ricorsiva del problema può essere espressa come:



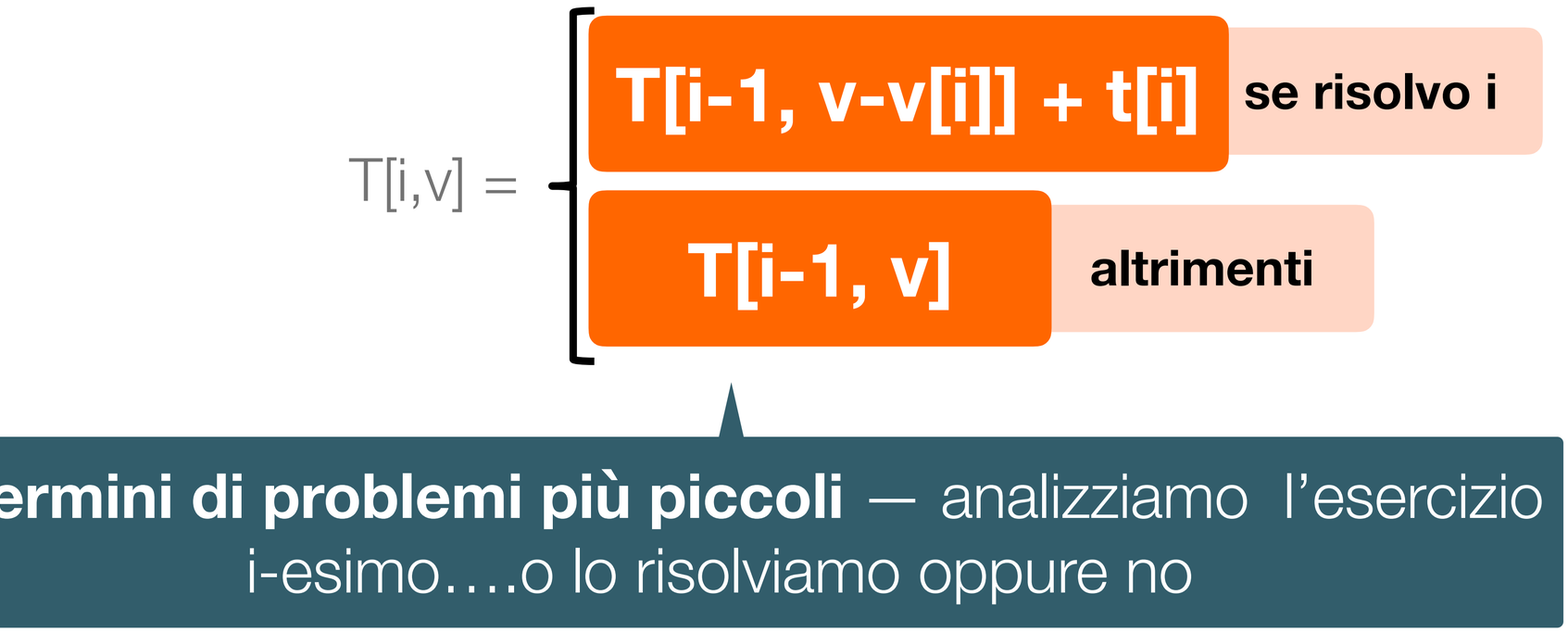
Programmazione Dinamica — formulazione ricorsiva

- Formulazione ricorsiva del problema può essere espressa come:



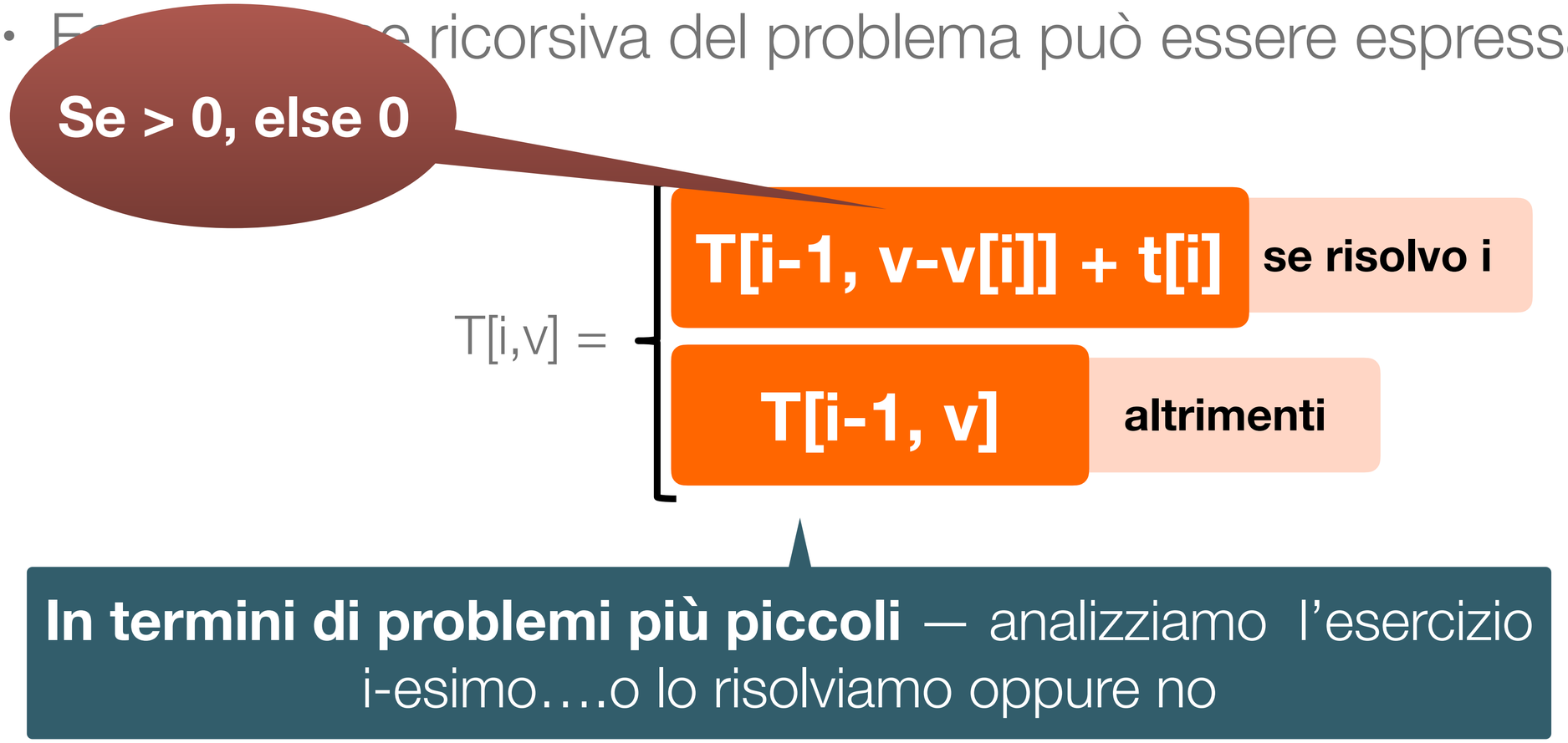
Programmazione Dinamica — formulazione ricorsiva

- Formulazione ricorsiva del problema può essere espressa come:



Programmazione Dinamica — formulazione ricorsiva

- Formulazione ricorsiva del problema può essere espressa come:



Programmazione Dinamica

integer studente-pigro(**integer**$[\,]\ v$, **integer**$[\,]\ t$, **integer** n, **integer** V)

integer$[\,][\,]\ T \leftarrow$ **newinteger**$[0 \ldots n][1 \ldots V]$
for $v \leftarrow 1$ **to** V **do**
  $T[0, v] \leftarrow +\infty$
for $i \leftarrow 1$ **to** n **do**
  $T[i, v] \leftarrow \mathsf{min}(T[i-1, v], t[i] + \mathsf{iif}(v - v[i] > 0, M[i-1, v - v[i]], 0))$
return $T[n, V]$

Programmazione Dinamica

integer studente-pigro(**integer**$[\,]$ v, **integer**$[\,]$ t, **integer** n, **integer** V)

integer$[\,][\,]$ $T \leftarrow$ **newinteger**$[0 \ldots n][1 \ldots V]$
for $v \leftarrow 1$ **to** V **do**
  $T[0, v] \leftarrow +\infty$
for $i \leftarrow 1$ **to** n **do**
  $T[i, v] \leftarrow \mathsf{min}(T[i-1, v], t[i] + \mathsf{iif}(v - v[i] > 0, M[i-1, v - v[i]], 0))$
return $T[n, V]$

Complessità?

Programmazione Dinamica

integer studente-pigro(**integer**[ ] v, **integer**[ ] t, **integer** n, **integer** V)

integer[ ][ ] $T \leftarrow$ **newinteger**$[0 \ldots n][1 \ldots V]$
for $v \leftarrow 1$ **to** V **do**
  $T[0, v] \leftarrow +\infty$
for $i \leftarrow 1$ **to** n **do**
  $T[i, v] \leftarrow \min(T[i-1, v], t[i] + \text{iif}(v - v[i] > 0, M[i-1, v - v[i]], 0))$
return $T[n, V]$

Complessità? O(nV)

Programmazione Dinamica

integer studente-pigro(**integer**[ ] v, **integer**[ ] t, **integer** n, **integer** V)

integer[ ][ ] $T \leftarrow$ **newinteger**$[0 \ldots n][1 \ldots V]$
for $v \leftarrow 1$ **to** V **do**
  $T[0, v] \leftarrow +\infty$
for $i \leftarrow 1$ **to** n **do**
  $T[i, v] \leftarrow \min(T[i-1, v], t[i] + \text{iif}(v - v[i] > 0, M[i-1, v - v[i]], 0))$
return $T[n, V]$

Complessità? O(nV)

Algoritmo di che tipo?

Programmazione Dinamica

integer studente-pigro(**integer**$[\,]\ v$, **integer**$[\,]\ t$, **integer** n, **integer** V)

integer$[\,][\,]\ T \leftarrow$ **newinteger**$[0 \ldots n][1 \ldots V]$
for $v \leftarrow 1$ **to** V **do**
  $T[0, v] \leftarrow +\infty$
for $i \leftarrow 1$ **to** n **do**
  $T[i, v] \leftarrow \mathsf{min}(T[i-1, v], t[i] + \mathsf{iif}(v - v[i] > 0, M[i-1, v - v[i]], 0))$
return $T[n, V]$

Complessità? O(nV)

Pseudo-polinomiale

Infine….

Supponete ora che risolvere un esercizio in **maniera parziale** dia la possibilità di ottenere una **porzione di voto proporzionale**. Descrivete un algoritmo di costo O(n log n) per risolvere lo stesso problema….

Infine….

Supponete ora che risolvere un esercizio in **maniera parziale** dia la possibilità di ottenere una **porzione di voto proporzionale**. Descrivete un algoritmo di costo O(n log n) per risolvere lo stesso problema….

Un **approccio greedy** può funzionare come con lo zaino frazionario

- **Ordinare** i problemi per m[i]/v[i] crescente (o per v[i]/m[i] decrescente)
- A questo punto, **iteriamo sulla lista ordinata**, fino a quando V non è stato raggiunto
- Solo l'ultimo problema può essere risolto parzialmente

Data Center

Esercizio Programmazione Dinamica

Giuseppe Prencipe — Dipartimento di Informatica — Università di Pisa

Problema

- Siete i sistemisti di un data center in grado di “**macinare**” diversi terabyte di dati al giorno. In un **periodo di n giorni**, vi viene comunicato in anticipo la **quantità x_i di dati da gestire nel giorno i**. I dati non macinati nel giorno i non possono essere gestiti il giorno successivo

- Purtroppo, la scelta del sistema operativo non è stata fra le più felici. Infatti le performance del sistema peggiorano di giorno in giorno (forse per memory leaks?), fino a quando non decidete di fare **reboot**

- Dopo j giorni dall’ultimo reboot, la quantità massima di terabyte “macinabili” è pari a sj, con s1> s2> . . . > sn

- Peggio ancora, per fare reboot ci vuole un giorno intero e durante quel giorno non è possibile gestire alcun dato

Problema

- Dato un vettore **x[1..n]** che misura la **quantità** di dati in input e un vettore **s[1..n]** che misura le **performance** di sistema dall'ultimo reboot, **scrivere un programma che calcola il numero massimo di terabyte che possono essere processati dal sistema** (è ovviamente consentito il reboot, considerando il giorno che si perde)

Problema

- Dato un vettore **x[1..n]** che misura la **quantità** di dati in input e un vettore **s[1..n]** che misura le **performance** di sistema dall'ultimo reboot, **scrivere un programma che calcola il numero massimo di terabyte che possono essere processati dal sistema** (è ovviamente consentito il reboot, considerando il giorno che si perde)

Esempio:

i	1	2	3	4
x	10	1	7	7
s	8	4	2	1

Problema

- Dato un vettore **x[1..n]** che misura la **quantità** di dati in input e un vettore **s[1..n]** che misura le **performance** di sistema dall'ultimo reboot, **scrivere un programma che calcola il numero massimo di terabyte che possono essere processati dal sistema** (è ovviamente consentito il reboot, considerando il giorno che si perde)

Esempio:

- La **soluzione ottimale** consiste nel fare un singolo reboot nel giorno 2
  - Si processano quindi 8TB nel giorno 1, 7TB nel giorno 3, 4TB nel giorno 4 per un totale di 19TB

i	1	2	3	4
x	10	1	7	7
s	8	4	2	1

Programmazione Dinamica — formulazione ricorsiva

- Proviamo a capire le dinamiche in questo problema
  - La quantità di tera che posso processare da oggi dipende da quanti giorni fa ho fatto l'ultimo reboot
    - Come dati ho **giorni a cui sono**, **giorni dall'ultimo reboot**, e **quantità di tera**
    - Vogliamo ottimizzare la quantità di tera, quindi i valori delle celle della tabella saranno quantità di tera
- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso?

Programmazione Dinamica — formulazione ricorsiva

- Proviamo a capire le dinamiche in questo problema
  - La quantità di tera che posso processare da oggi dipende da quanti giorni fa ho fatto l'ultimo reboot
    - Come dati ho **giorni a cui sono**, **giorni dall'ultimo reboot**, e **quantità di tera**
    - Vogliamo ottimizzare la quantità di tera, quindi i valori delle celle della tabella saranno quantità di tera
- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Quali valori possiamo inserire in tabella subito?

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Quali valori possiamo inserire in tabella subito?

P(n, j) = min{????}

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Quali valori possiamo inserire in tabella subito?

$$P(n, j) = \min\{x[n], s[j]\}$$

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Quali valori possiamo inserire in tabella subito?

$$P(n, j) = \min\{x[n], s[j]\}$$

Quindi in quale cella troviamo il risultato finale?

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Quali valori possiamo inserire in tabella subito?

$$P(n, j) = \min\{x[n], s[j]\}$$

Quindi in quale cella troviamo il risultato finale? P(1, 1)

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

P[i,j] = ⎧
⎨
⎩

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

se oggi decido un reboot

P[i,j] = ⎧
⎨
⎩

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

P[i,j] = ⎧ **se oggi decido un reboot**

Da **domani** posso ripartire a pieno regime

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Da **domani** posso ripartire a pieno regime

P[i,j] = { **P[i+1, 1]** **se oggi decido un reboot**

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

Da **domani** posso ripartire a pieno regime

P[i,j] =

P[i+1, 1] **se oggi decido un reboot**

altrimenti

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

P[i,j] =

P[i+1, 1] **se oggi decido un reboot**

altrimenti

Da **domani** posso ripartire a pieno regime

Oggi **faccio quello che posso**, e da domani continuo

Programmazione Dinamica — formulazione ricorsiva

- Cosa usiamo come tabella per rappresentare gli altri due dati che abbiamo in ingresso? Matrice **P(i, j)**

Cosa rappresentano gli indici **i** e **j**?

i — giorno a cui sono **j** — giorni da ultimo reboot

$$P[i,j] = \begin{cases} P[i+1, 1] & \text{se oggi decido un reboot} \\ \min\{x[i], s[j]\} + P[i+1, j+1] & \text{altrimenti} \end{cases}$$

Da **domani** posso ripartire a pieno regime

Oggi **faccio quello che posso**, e da domani continuo

Programmazione Dinamica — formulazione ricorsiva

- Sia **P(i, j)** la massima quantità di terabyte che si possono gestire nei giorni **[i..n]**, nel caso in cui l'ultimo reboot sia stato effettuato **j** giorni prima del giorno **i**

- **Il problema originale corrisponde a calcolare P(1, 1)**
  P(i, j) può essere calcolato ricorsivamente come segue

$$P(i,j) = \begin{cases} \min\{x[i], s[j]\} & i = n \\ \max\{P(i+1,1), \min\{x[i], s[j]\} + P(i+1, j+1)\} & i < n \end{cases}$$

Soluzione

```
integer datacenter(integer[ ] x, integer[ ] s, integer n)
    integer[ ][ ] P ← new integer[1 . . . n][1 . . . n]
    for j ← 1 to n do
        P[n, j] ← min(x[i], s[j])
    for i ← n − 1 downto 1 do
        for j ← 1 to n − 1 do
            P[i, j] = max(P[i + 1, 1], min(x[i], s[j]) + P[i + 1, j + 1])
    return P[1, 1]
```

Complessità: $O(n^2)$

KEEP CALM AND STUDY ALGORITHMS