

Algorísmica Avançada

Programació dinàmica

Sergio Escalera

A series of horizontal lines of varying lengths and colors (teal, light blue, and white) extending from the left side of the slide towards the right, positioned below the author's name.

Programació dinàmica

- La programació dinàmica ens permet resoldre una gran quantitat de problemes (més dels que hem vist fins ara)
- Això, com veurem, té un cost en la complexitat

Teorema de Optimalidad de Mitten

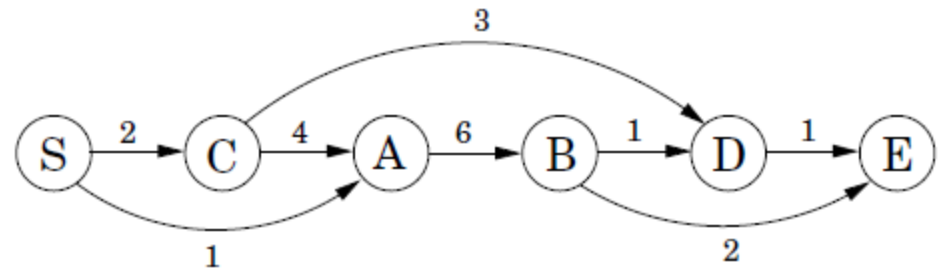
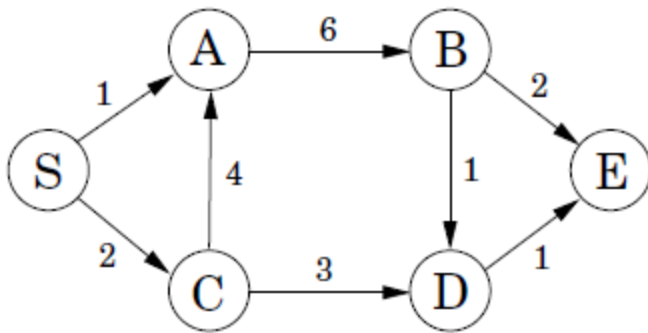
El objetivo básico en la programación dinámica consiste en ‘descomponer’ un problema de optimización en k variables a una serie de problemas con menor número de variables más fáciles de resolver.

La solución óptima obtenida se ajusta al **principio de optimalidad** establecido por R. Bellman en 1957.

Una política óptima tiene la propiedad de que, cualquiera que sea el estado inicial y las decisiones iniciales, las restantes decisiones deben constituir una política óptima con respecto al estado resultante de la primera decisión

Programació dinàmica

- Relació amb la linearització de grafs



- Distància més curta a D:

$$\text{dist}(D) = \min\{\text{dist}(B) + 1, \text{dist}(C) + 3\}$$

Programació dinàmica

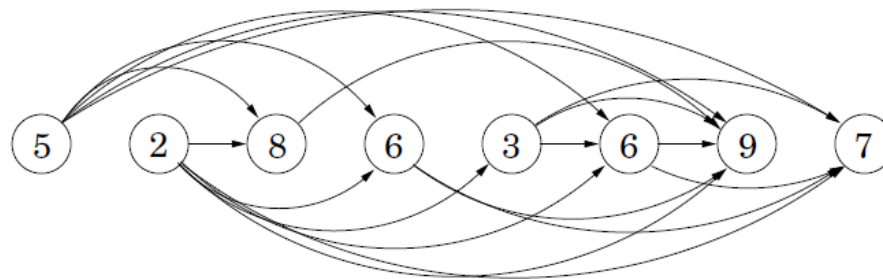
- Podem calcular la distància a tots els nodes en un pas:

```
initialize all dist(.) values to  $\infty$   
dist(s) = 0  
for each  $v \in V \setminus \{s\}$ , in linearized order:  
    dist(v) =  $\min_{(u,v) \in E} \{ \text{dist}(u) + l(u,v) \}$ 
```

- Solucionem un conjunt de subproblemes fins que arribem a una solució final: tècnica molt general!!
- **Tenim una funció sobre els nodes anteriors per actualitzar una resposta al node actual:**
 - → Aquesta funció podria ser qualsevol! (i.e. Màxim en lloc de mínim)

Programació dinàmica

- Per obtenir una representació de programació dinàmica, suposem que el graf disposa de totes les connexions entre un node i els seus predecessors:



- Exemple: trobar el camí de longitud màxima

```

for  $j = 1, 2, \dots, n$ :
     $L(j) = 1 + \max\{L(i) : (i, j) \in E\}$ 
return  $\max_j L(j)$ 

```

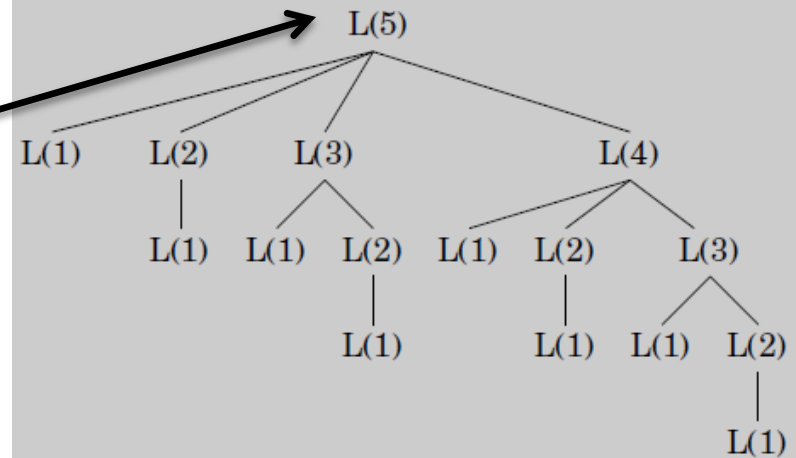
Programació dinàmica

- Complexitat: llista adjacència (en temps lineal)
 - Càlcul de relacions entre els subproblemes: **$O(n^2)$**
 - Simple i eficient

$$L(j) = 1 + \max\{L(1), L(2), \dots, L(j-1)\}.$$

recursion for $L(5)$

En aquest cas la recursivitat no seria eficient: exponencial, es recalculen solucions prèviament estimades (en enumeratiu: ramificació i poda ho solucionarem).



Programació dinàmica

- Aplicació: distància d'edició

SNOWY						SUNNY						
S	—	N	O	W	Y	—	S	N	O	W	—	Y
S	U	N	N	—	Y	S	U	N	—	—	N	Y
Cost: 3						Cost: 5						

- Tenim inserció, eliminació i substitució
- Hi ha moltes combinacions!!!
- Donem una solució amb programació dinàmica

Programació dinàmica

- Taula de subproblemes

```
for  $i = 0, 1, 2, \dots, m$ :
```

```
     $E(i, 0) = i$ 
```

```
for  $j = 1, 2, \dots, n$ :
```

```
     $E(0, j) = j$ 
```

```
for  $i = 1, 2, \dots, m$ :
```

```
    for  $j = 1, 2, \dots, n$ :
```

```
         $E(i, j) = \min\{E(i-1, j) + 1, E(i, j-1) + 1, E(i-1, j-1) + \text{diff}(i, j)\}$ 
```

```
return  $E(m, n)$ 
```

Depenen de l'objectiu del problema



E X P O N E N T I A L
- - P O L Y N O M I A L

$O(mn)$

Programació dinàmica

- Taula de subproblemes

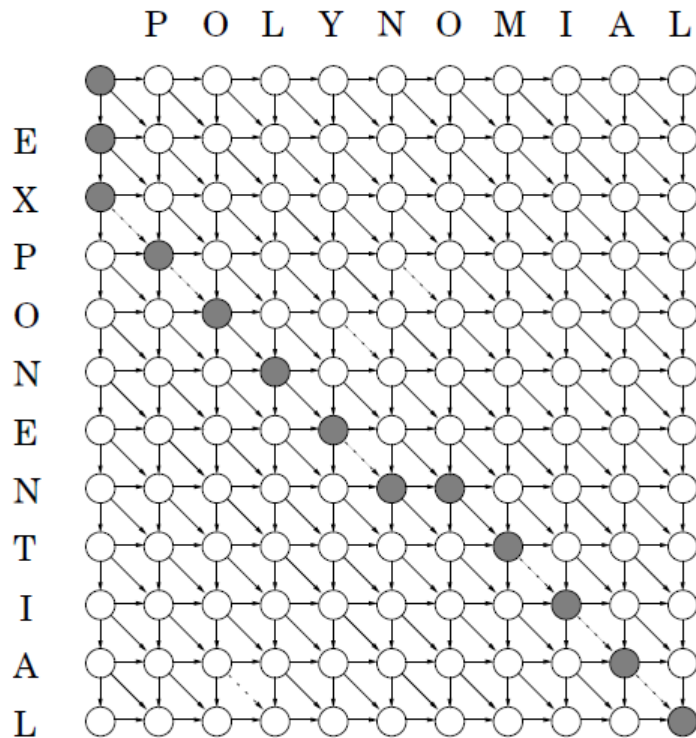
$$E(i, j) = \min\{E(i-1, j) + 1, E(i, j-1) + 1, E(i-1, j-1) + \text{diff}(i, j)\}$$

			$j-1$	j			n
$i-1$							
i							
m							GOAL

		P	O	L	Y	N	O	M	I	A	L
	0	1	2	3	4	5	6	7	8	9	10
E	1	1	2	3	4	5	6	7	8	9	10
X	2	2	2	3	4	5	6	7	8	9	10
P	3	2	3	3	4	5	6	7	8	9	10
O	4	3	2	3	4	5	5	6	7	8	9
N	5	4	3	3	4	4	5	6	7	8	9
E	6	5	4	4	4	5	5	6	7	8	9
N	7	6	5	5	5	4	5	6	7	8	9
T	8	7	6	6	6	5	5	6	7	8	9
I	9	8	7	7	7	6	6	6	6	7	8
A	10	9	8	8	8	7	7	7	7	6	7
L	11	10	9	8	9	8	8	8	8	7	6

Programació dinàmica

- Taula de subproblemes



E X P O N E N – T I A L
– – P O L Y N O M I A L

Hi han diferents camins amb el mateix cost associat

Els camins varien segons fixem costos diferents a diferents operacions de inserció, eliminació i substitució

Programació dinàmica

- Exercici

```

for  $i = 0, 1, 2, \dots, m$ :
     $E(i, 0) = 2i$ 
for  $j = 1, 2, \dots, n$ :
     $E(0, j) = j$ 
for  $i = 1, 2, \dots, m$ :
    for  $j = 1, 2, \dots, n$ :
         $E(i, j) = \min\{E(i-1, j) + 2, E(i, j-1) + 1, E(i-1, j-1) + \text{diff}(i, j)\}$ 
return  $E(m, n)$ 

```

Associar la paraula ALGORISMICA amb la paraula AVANÇADA fent ús d'aquesta inicialització i funció de programació dinàmica

Programació dinàmica

- Exercici

[illegible]

Programació dinàmica

- Exercici

		A	L	G	O	R	I	S	M	I	C	A
	0	1	2	3	4	5	6	7	8	9	10	11
A	2	0	1	2	3	4	5	6	7	8	9	10
V	4	2	1	2	3	4	5	6	7	8	9	10
A	6	4	3	2	3	4	5	6	7	8	9	9
N	8	6	5	4	3	4	5	6	7	8	9	10
Ç	10	8	7	6	5	4	5	6	7	8	9	10
A	12	10	9	8	7	6	5	6	7	8	9	9
D	14	12	11	10	9	8	7	6	7	8	9	10
A	16	14	13	12	11	10	9	8	7	8	9	9

Programació dinàmica

		A	L	G	O	R	I	S	M	I	C	A
	0	1	2	3	4	5	6	7	8	9	10	11
A	2	0	1	2	3	4	5	6	7	8	9	10
V	4	2	1	2	3	4	5	6	7	8	9	10
A	6	4	3	2	3	4	5	6	7	8	9	9
N	8	6	5	4	3	4	5	6	7	8	9	10
Ç	10	8	7	6	5	4	5	6	7	8	9	10
A	12	10	9	8	7	6	5	6	7	8	9	9
D	14	12	11	10	9	8	7	6	7	8	9	10
A	16	14	13	12	11	10	9	8	7	8	9	9

ALGORISMICA
AAA AVANÇADA

Molts camins de cost 9

ALGORISMICA

 AVANÇADA

Programació dinàmica

- Un problema de transport:
- Necessitem transportar el màxim de valor, transportant ítems si el nostre pes màxim es W :

Item	Weight	Value
1	6	\$30
2	3	\$14
3	4	\$16
4	2	\$9

- Escenari:
 - La quantitat de cada ítem és il·limitada

Programació dinàmica

- Quantitat ítem = il·limitada
 - Quins són els subproblemes?
 - Incloure un nou producte
 - Quin és l'objectiu? maximitzar $K(w) = \max_{i:w_i \leq w} \{K(w - w_i) + v_i\}$
 - Es pot solucionar en una dimensió

```

K(0) = 0
for w = 1 to W:
    K(w) = max{K(w - w_i) + v_i : w_i ≤ w}
return K(W)
  
```

Programació dinàmica

- Quantitat ítem = il·limitada

Item	Weight	Value
1	6	\$30
2	3	\$14
3	4	\$16
4	2	\$9

$$K(w) = \max_{i:w_i \leq w} \{K(w - w_i) + v_i\}$$

$$K(0) = 0$$

for $w = 1$ to W :

$$K(w) = \max\{K(w - w_i) + v_i : w_i \leq w\}$$

return $K(W)$

$$W=5$$

$$K(0)=0$$

$$K(1)=0$$

$$K(2)=9$$

$$K(3)=14$$

$$K(4)=$$

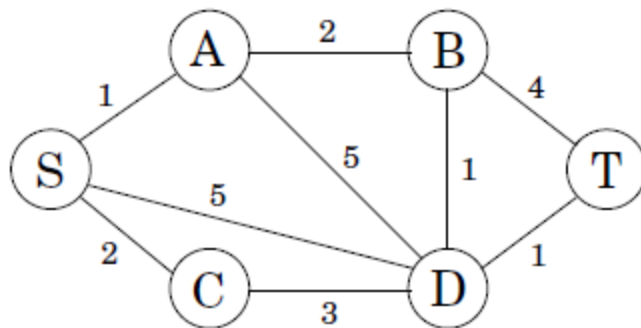
$$\text{MAX}\{K(2)+9, K(1)+14, K(0)+16\}=18$$

$$K(5)=\text{MAX}\{K(3)+9, K(2)+14, K(1)+16\}$$

$$=23$$

Programació dinàmica

- Camins curts: compliquem el problema!
- Volem camí de cost reduït però passant per poques arestes \rightarrow Dijkstra no guarda aquesta informació!



$i \leq k, \text{dist}(v, i)$ Distància del camí més curt de s a v que passa per i nodes

$$\text{dist}(v, i) = \min_{(u,v) \in E} \{ \text{dist}(u, i-1) + \ell(u, v) \}$$

Programació dinàmica

- I com podem guardar la informació dels camins mínims entre totes les parelles possibles de nodes?
- Si fem ús dels algorismes dels temes anteriors per totes les possibles parelles tenim una complexitat $O(|V|^2|E|)$
- Normalment $|E| \gg |V|$
- Si guardem informació de subproblemes podem aconseguir una complexitat $O(|V|^3)$

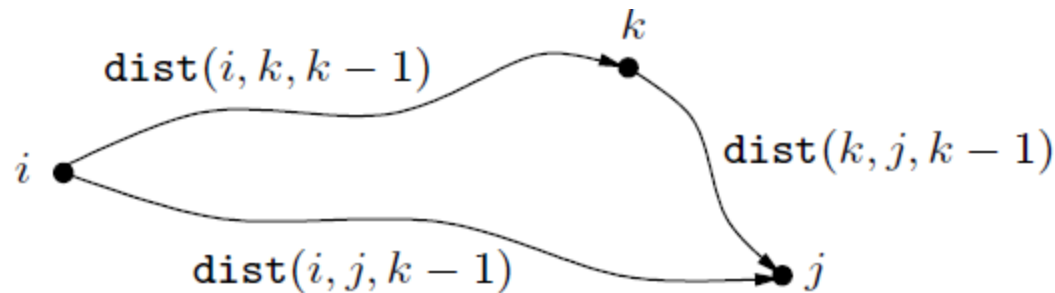
Programació dinàmica

- Floyd-Warshall

- Fem ús d'una matriu tridimensional, on

$$\text{dist}(i, j, k) \quad \{1, 2, \dots, k\}$$

és la distància del camí més curt entre i i j tenint en compte només els nodes $1, \dots, k$



$$\text{dist}(i, k, k-1) + \text{dist}(k, j, k-1) < \text{dist}(i, j, k-1),$$

Programació dinàmica

- Floyd-Warshall

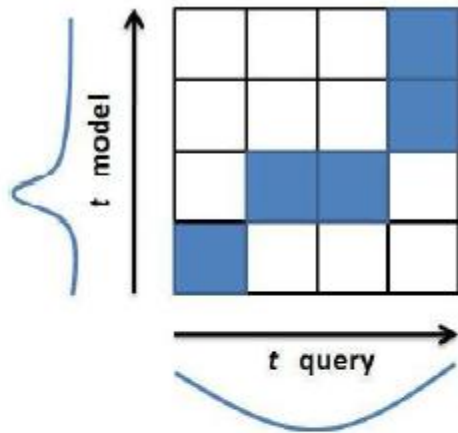
```
for  $i = 1$  to  $n$ :  
    for  $j = 1$  to  $n$ :  
         $\text{dist}(i, j, 0) = \infty$   
  
for all  $(i, j) \in E$ :  
     $\text{dist}(i, j, 0) = \ell(i, j)$   
for  $k = 1$  to  $n$ :  
    for  $i = 1$  to  $n$ :  
        for  $j = 1$  to  $n$ :  
             $\text{dist}(i, j, k) = \min\{\text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1), \text{dist}(i, j, k - 1)\}$ 
```

Programació dinàmica

- Dynamic Time Warping
 - El temps és dinàmic per definició
 - Es poden definir subproblemes temporals per resoldre problemes temporals més grans
 - → Un altre cop tenim la programació dinàmica!!
- Veiem el problema de reconeixement de gestos i accions

Programació dinàmica

- Valors espai-temporals de la matriu de programació dinàmica



Programació dinàmica

- DTW (Wikipedia) → ja ho hem vist abans no?

```
int DTWDistance(char s[1..n], char t[1..m]) {
    declare int DTW[0..n, 0..m]
    declare int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity
    for i := 1 to n
        DTW[i, 0] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j],           // insertion
                                         DTW[i, j-1],           // deletion
                                         DTW[i-1, j-1])          // match

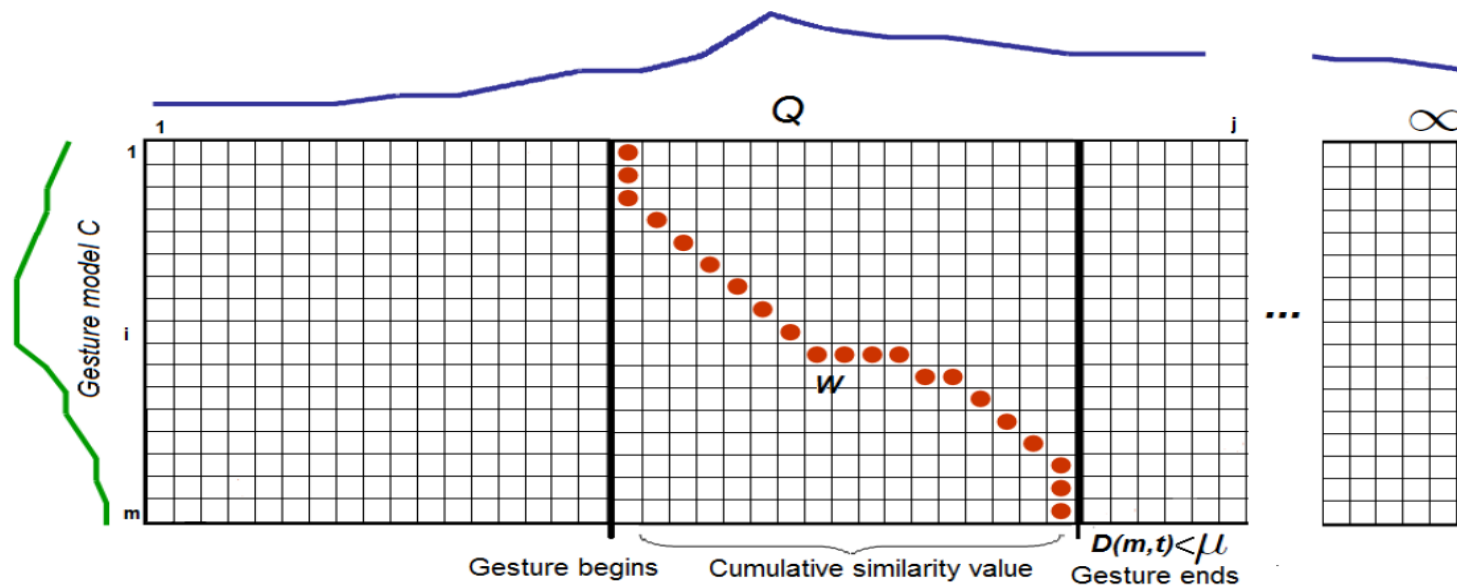
    return DTW[n, m]
}
```

Programació dinàmica

- Resultat – seqüència discreta, trajectòries 2D

Programació dinàmica

- DTW Detecció de patrons en seqüències més



A cada nou instant de temps tenim diferents hipòtesis: possibilitat de néixer primera posició d'un patró temporal (cost mínim amunt) o que estiguem a mig d'un patró (la distància mínima ve propagada de posicions anteriors)

Programació dinàmica

- Resultat – seqüència infinita, trajectòries 3D

Programació dinàmica

- Exercicis
- Tenim una trajectòria definida per vectors binaris amb múltiples candidats:

1 1 1 0 0 1 0 1

0 0 1

1 1 1

Volem fer la correspondència amb el patró

1 1 1 1 0 0 0 1 fent servir DTW i distància de Hamming (número de bits diferents)

$$\gamma(i, j, k) = d(i, j, k) + \min\{\gamma((i-1, j-1), (i-1, j), (i, j-1) \times \{1, \dots, K\})\}$$

→ Fes la taula de programació dinàmica i mostra el cost i el “working path”

Programació dinàmica

1 1 1 0 0 1 0 1

0 0 1

1 1 1

	0	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	1	1
1	2	0	0	0	0	0	0	0	1	1
1	3	0	0	0	0	0	0	0	1	1
1	4	0	0	0	0	0	0	0	1	1
0	5	1	1	1	0	0	0	1	0	1
0	6	2	2	2	0	0	0	1	0	1
0	7	3	3	3	0	0	0	1	0	1
1	8	3	3	3	0	0	0	0	1	0

1 1 1 1 1 1 1 1 0 1
 1 1 1 1 0 0 0 1