

Metodi: Euristicci

- Euristiche
- Meta-Euristiche

Persone emere:

- Costruttivi/Distruttivi (Ants Colony Optimiz.)
- Exchange (Simulated Annealing)
- Ricombinazione (Genetic Alg)

Esempio di Applicaz. Euristiche al problema viaggiatore:

- Rappresentazione delle soluzioni

TSP, soluzione: $\boxed{\cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot}$
1 permutazione dell'input (di $(n-1)!$)

- Costo/guadagno

.

1) RANDOM-SAMPLING (TSP-istanza $\star t$, int m-sample, tsp * best-solution)

CODICE DA SKIENA

2) Metodo delle discese/salita del grediente

Local Search

SKIENA

3) Simulated Annealing

SKIENA

Evolutionary Computation Algorithm

$P \leftarrow \text{RANDOM}$

WHILE (! STOP CONDITION)

FOR $i = 1 ; i \leq |P|/2 ; i++$

offspring[i] = select(P)

offspring[i+1] = select(P)

Crossover(offspring[i], offspring[i+1])

Mutation(" ", " ")

Programmazione Dinamica

- "Ricerca Brute force Intelligente"

- Ricerca sulla divisione dei problemi in sottoproblemi
 - ↳ È intelligente perché NON risolve 2 volte lo stesso sottoproblema
- Riuso delle soluzioni e utilizzo di ricorsione.

Esempio Applicativo:

Fibonacci sequence:

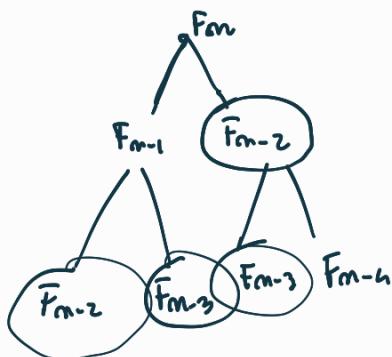
$$\frac{F_m}{F_{m-1}} = \varphi \approx \frac{1 + \sqrt{5}}{2} \quad] \text{ numero aureo}$$

$$F_m \approx \varphi^m$$

NAIVE ALG.

```
int f(m)
    if (m ≤ 2)
        return 1;
    else
        return f(m-1) + f(m-2);
```

$$T(m) = T(m-1) + T(m-2) + \Theta(1) \geq 2T(m-2) + \Theta(1) = 2^{m/2}$$



DOPPIONI! DA EVITARE

V2:

```
memo = {}  
f(m)  
if (m in memo)  
    return memo[m];  
  
else if (m ≤ 2)  
    f = 1  
else  
    f = f(m-1) + f(m-2)  
memo[m] = f;  
return f;
```

- m chiamate MEMOIZED
 - ogni chiamata è $\Theta(1)$
- In Totale : $\Theta(m)$

In generale, usano le ricorsione + memorization:

$T(m) = \# \text{ sottoproblemi} \cdot \text{tempo/sottoproblema}$

Questo approccio si chiama TOP-DOWN DP

V3: Bottom up DP

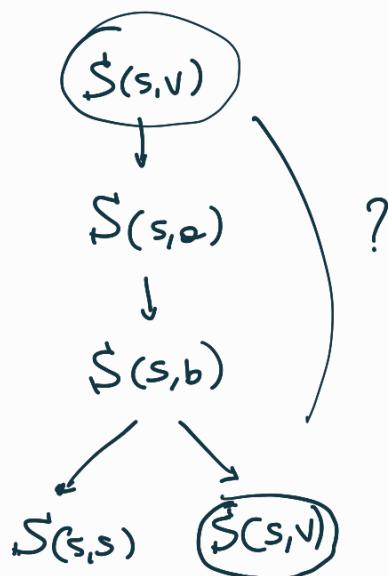
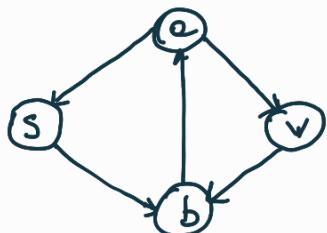
```
fib = {}  
f(m)  
  
for k=1...m  
    if k≤2  
        f = 1;  
    else  
        f = fib[k-1] + fib[k-2]  
  
    fib[k] = f;  
  
return fib[k];
```

Shortest Paths

Problema del minimo percorso tra s e v , $\forall v$: $S(s, v)$ $\forall v$.

$$S(s, v) = \min_u \left\{ S(s, u) + w(u, v) \mid (u, v) \in E \right\} \quad (E \text{ insieme degli Edge})$$

Ese:



Questo algoritmo è applicabile solo se il grafo è ACICLICO (DAG)

$$[T(v) = \Theta(V+E)]$$

E se ho cicli nel grafo?

Shortest path con maximo K edge finiti:

$$S_K(s, v) = \min_u \left\{ S_{K-1}(s, u) + w(u, v) \mid (u, v) \in E \right\}$$

C.B.:

- $S_0(s, v) = \infty, \forall v \neq s$

- $S_K(s, s) = 0, \forall K$

(In assenza di cicli/autosanelli negativi)

$$\Rightarrow S(s, v) = \underbrace{S_{|V|-1}(s, v)}_{\substack{\uparrow \\ \text{Il problema di prima}}} \quad \downarrow \quad \text{Il problema ottimale} \\ \text{con } K = \# \text{vertici} - 1$$

Applico una riduzione con copie del grafo

$$\text{Tempo complessivo} = |V| \cdot |V| \cdot \theta(v) = \Theta(|V|^3)$$

↑
indegree(v)

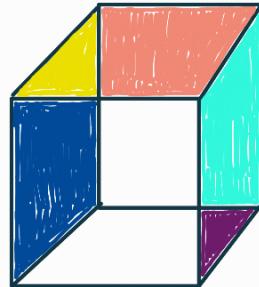
$$\text{Ma in realtà : } \theta\left(M \sum_{v \in V} \text{indegree}(v)\right) = \Theta(V \cdot E)$$

↓
ALG. BELLMAN FORD

LEzione 2 sulle DP

DP \Rightarrow Ricerca del minimo percorso nel DAG dei sottoproblemi.

DP \Rightarrow Ricorrenza + Memoization + Guessing



5 passi per la risoluzione di un problema usando la DP

1. Definizione dei sottoproblemi $\rightarrow \#$ Combinazioni di sottoproblemi
2. Guessing (Il modo in cui uso i sottoproblemi per costruire il problema più esterno) $\rightarrow \#$ Possibili scelte
3. "Collegare" i sottoproblemi $\rightarrow \#$ Tempo per sottoproblemi
 - a. Scrivere le ricorrenze
4. Ricorrenza + Memoization / Bottom-up
 - a. Check del grafo dei sottoproblemi
sia aciclico
5. Risolvere il problema originario $\rightarrow \#$ Eventuale tempo extra

	FIBONACCI	SHORTEST PATHS $S(s, v), \forall v$
Sottoproblemi # sottoproblemi	$F_k, k=1 \dots n$ n	$S_k(s, v), \forall v \in V, \forall k=0 \dots V -1$ V^2
Guessing # possibili scelte	No 1	# etchi entranti in v $\text{indegree}(v) + 1$
Ricorrenze	$F_k = F_{k-1} + F_{k-2}$	$S_k(s, v) = \min_u \{ S_{k-1}(s, u) + w(u, v) \mid (u, v) \in E \}$
Tempo/Sottoproblema	$\Theta(1)$	$\Theta(\text{indegree}(v) + 1)$
Check DAG Tempo Totale	Nessun ciclo $\Theta(n)$	Nessun ciclo $\Theta(VE) + \Theta(V^2) \approx \Theta(VE)$
Problema Ottimizzo (Extreme Time)	F_m $\Theta(1)$	$S_{\text{in}-d}(s, v), \forall v \in V$ $\Theta(V)$

5 Step

1. Definizione e conteggio dei sottoproblemi
2. Identificare scelte possibili $\rightarrow \# \text{scelte}$
3. Relazionare i sottoproblemi $\rightarrow t/\text{sottoproblema}$
4. Ricorsione + Memoization / Bottom-UP e
controllo aciclicità del grafo dei sottoproblemi \rightarrow

$$t = \# \text{sottop.} \times \frac{t}{\text{sottop.}}$$

5. Problema Origine (Extra Time)

Ex: Shortest Paths

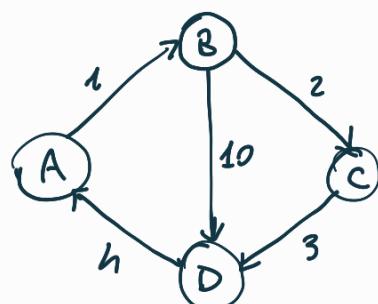
$$1. S_k(s, v) \quad \forall s, \forall v.$$

$$2. S(s, v), \forall v.$$

\hookrightarrow Per risolvere questo nei grafici ciclici risolvendo

$$3. S_{\leq k}(s, v), \forall v$$

Risolvendo il problema \downarrow nel grafo:



$$S_2(A, D) = 11$$

$$S_3(A, D) = 6$$

$$S_4(A, D) = 10$$

$$S_5(A, D) = 26$$

Scrivo la ricorrenza: $S_k(s, v) = \min \left\{ S_{k-1}(s, u) + w(u, v) \mid (u, v) \in E \right\}$

Mi serve una matrice dei risultati parziali: MEMORIZATION

	A	B	C	D
A	∞	1	∞	∞
B	∞	∞	2	10
C	∞	∞	∞	3
D	4	∞	∞	∞

In ogni cella ho $\min S_k(i,j)$

	A	B	C	
A	∞			11
B		∞		
C			∞	
D				∞

Guardando quella precedente.

Complessità: $\Theta(k \cdot v^2 \cdot v)$



Risolu^o prob. 2) MINIMO PERCORSO IN UN GRAFO ACICLICO

$S(s, v)$, $\forall v$.

Senza cicli $\theta(V+E) = \theta\left(\sum_{v \in V} \text{indegree}(v) + 1\right)$

Se il grafo ha ciclo il problema 2 si trasforma nel 3.

$S(s, v) = S_{|V|-1}(s, v)$ IN ASSENZA DI CICLI NEGATIVI

• $|V|-1$ è il numero di ecchi di un grafo inf. del # dei nodi.

A diff. del problema di prima

• $K = |V|$

• Non ho K matrici ma K vettori (poiché sorgente fissa)
quindi complessozza = $\Theta(KV^2) = \Theta(V^3) = \Theta(V \cdot \sum_{v \in V} \text{indegree}(v) + 1) = \Theta(VE + V^2) \approx \Theta(VE)$

TEXT JUSTIFICATION

• words[0:m]

• Bedmax(i:s), $s > i$: $\begin{cases} \infty, & \text{se } \text{pageWidth} < \text{length}(w[i:s]) \\ (\text{pageWidth} - \text{length})^3, & \text{altrimenti} \end{cases}$

• Minimizzare la somma delle Bedmax

Il sotto problema: min bedmax per un suffiso $w[i:]$

$DP(i) = \min_{s > i} \{ \text{bedmax}(i:s) + DP(s) \}$

• Caso base $DP(m) = 0$

• Complessità = $\Theta(m^2) = \Theta(m \cdot m)$
tempo \approx n² problemi

A 6x6 grid of squares. The vertical axis is labeled 'i' at the top-left corner. The horizontal axis is labeled 'j' at the top-right corner.

Nella matrice salvo i valori trovati di DP in funzione di i e j .