Ottimizzazione Combinatoria

Algoritmi e Complessità

ANTONIO SASSANO

Università di Roma"La Sapienza"
Dipartimento di Informatica e Sistemistica
Corso di Laurea in "Ingegneria Gestionale"

Roma, Ottobre 2009

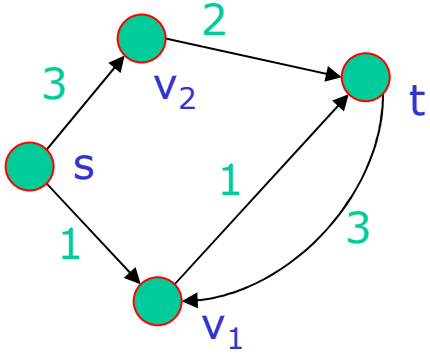
Problemi, Istanze e Soluzioni

- **PROBLEMA:** *Funzione F:* $\mathcal{I} \rightarrow \mathcal{S}$ *:*
  $\mathcal{I}$ *: Insieme delle Istanze (**INPUT**)*
  $\mathcal{S}$ *: Insieme delle Soluzioni (**OUTPUT**)*

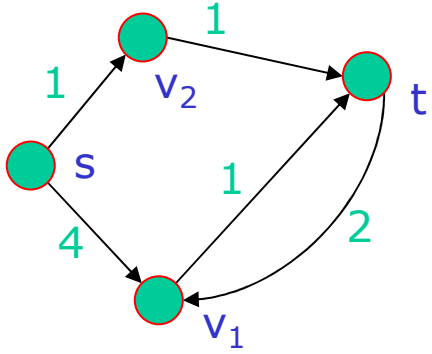
  F(I): Soluzione associata all'Istanza $I \in \mathcal{I}$

- **ISTANZA:** *Insieme delle informazioni di ingresso (dati):*
  Es. Un grafo G(N,A), le "lunghezze" degli archi e due nodi {s,t}

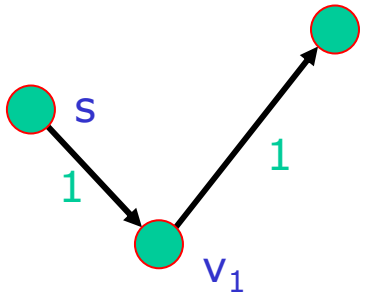
- **SOLUZIONE:** *Una particolare "proprietà" F(I) dell'istanza I*
  Es. Un Cammino di "lunghezza" minima da s a t su G(N,A)



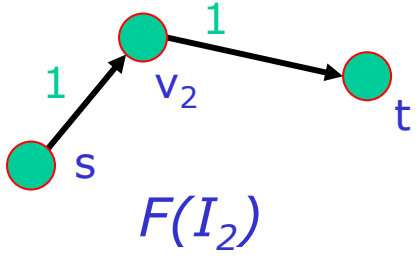
I_1



I_2



$F(I_1)$



$F(I_2)$

Algoritmi

- ALGORITMO:
  Procedura in grado di individuare, in PASSI SUCCESSIVI, una SOLUZIONE per una generica ISTANZA di un PROBLEMA.

✓ Istanza e Soluzione debbono essere opportunamente codificate
(ad es. univocamente associate a stringhe di "bit" (0,1))

- PROPRIETA' DI UN ALGORITMO
  - *CORRETTEZZA: applicato ad $I \in \mathcal{I}$ produce sempre F(I)*
  - *EFFICIENZA: in termini di SPAZIO e TEMPO*

✓ SPAZIO: Numero di bit necessari alla codifica di $I \in \mathcal{I}$

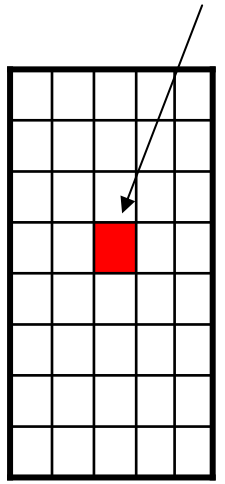
✓ TEMPO: Numero di passi necessari a produrre *F(I)*

Come si contano i "passi" di un algoritmo ?

.. e come si misura l'efficienza ?

Contare i passi

- MODELLO (IDEALIZZATO) DI CALCOLO:
  RANDOM ACCESS MACHINE



✓ La macchina (computer) effettua operazioni primitive su *celle* (parole) che contengono un numero costante di "bit" (es. 32).

✓ Ogni *operazione "primitiva"* richiede 1 passo

OPERAZIONI PRIMITIVE

- Operazioni logico-aritmetiche (+,-,*,/,if,..)
- Acccesso a matrici e vettori (A[23], B[12,23])
- Assegnazione di valori a variabili (A[i]:=34)
- Chiamate di funzioni e "sub-routine"
- Confronti tra numeri (A[i]>A[j])
- ...

✓ I "loop" (for,while,repeat,..) richiedono un numero di passi *che e' funzione della configurazione dei dati trattati*

Contare i passi - Esempio

- PROBLEMA: CALCOLO DELLA COMPONENTE DI MASSIMO VALORE IN UN VETTORE CON *d* COMPONENTI INTERE

INSIEME DELLE ISTANZE $\mathcal{I}$: Vettori A[ ] con d componenti intere
INSIEME DELLE SOLUZIONI $\mathcal{S}$: Insieme degli interi

ALGORITMO

```
Max := A[1]                          [2 passi – accesso vettore+assegnazione]
For i:=2 to d                        [2 passi – confronto (i con d)+incremento di i]
    do begin
      if A[i]> Max                   [2 passi – accesso vettore+confronto]
      then Max := A[i]               [2 passi – accesso vettore+assegnazione]
    end;
```

Solo se A[i]>Max

d-1 volte !

NUMERO TOTALE PASSI (TEMPO) T :
$2+4(d-1) \leq T \leq 2+6(d-1)$

Dipende dall'Istanza !!

Tempo di esecuzione di un Algoritmo

- TEMPO DI ESECUZIONE = *Numero di passi necessari ad un algoritmo per determinare la soluzione associata ad un'istanza*

DIPENDE DALLA:

- ✓ *Dimensione size(I) dell'Istanza* (es. # componenti del vettore)
- ✓ *Specifica Istanza* (es. Valore delle componenti del vettore)

- COMPLESSITA' DI UN ALGORITMO = *Numero di passi necessari per determinare la soluzione di una generica istanza avente dimensione size(I).*

FUNZIONE DELLA SOLA DIMENSIONE size(I) : $c(size(I))$

Come definire formalmente la dimensione size(I) ?

Come definire la funzione complessita` c(size(I)) ?

Definizione di *size(I)*

- DIMENSIONE *size(I)* = *Numero di celle necessarie a rappresentare i dati di ingresso (istanza I).*

- *size(I)* e` una *funzione f(x,y,..)* dei parametri dell'istanza I.

ESEMPIO:

- PROBLEMA DI PROGRAMMAZIONE LINEARE:
  $\min c^T x: \{x: Ax=b, x \geq 0\}$

Parametri di una generica istanza I :

n: numero variabili
m: numero vincoli
$A = [a_{hk}]$: matrice $(m \times n)$
$b = [b_h]$: vettore con m componenti
$c = [c_h]$: vettore con n componenti

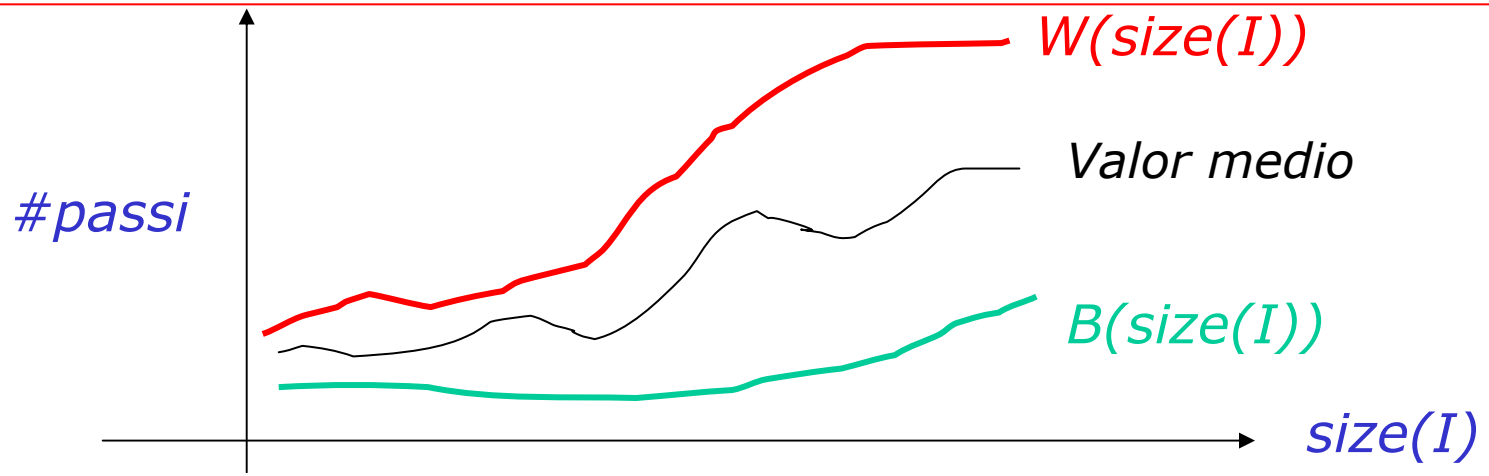
$size(I) = mn+n+m$

Definizione di c(size(I))

- Per definire *c(size(I))* *dovremmo considerare il valor medio del tempo di esecuzione su tutte le istanze di dimensione size(I).*
  Ma quale probabilita` hanno le diverse istanze di presentarsi ?
  Difficile determinarla (ma non impossibile)!
  Noi considereremo i casi estremi:

- COMPLESSITA' NEL *CASO PEGGIORE* *W(size(I))*
  - Funzione di *size(I)*
  - *Numero di passi necessari ad un algoritmo per determinare la soluzione della piu` difficile istanza di dimensione size(I).*

- COMPLESSITA' NEL *CASO MIGLIORE* *B(size(I))*
  - Funzione di *size(I)*
  - *Numero di passi necessari ad un algoritmo per determinare la soluzione della piu` facile istanza di dimensione size(I).*



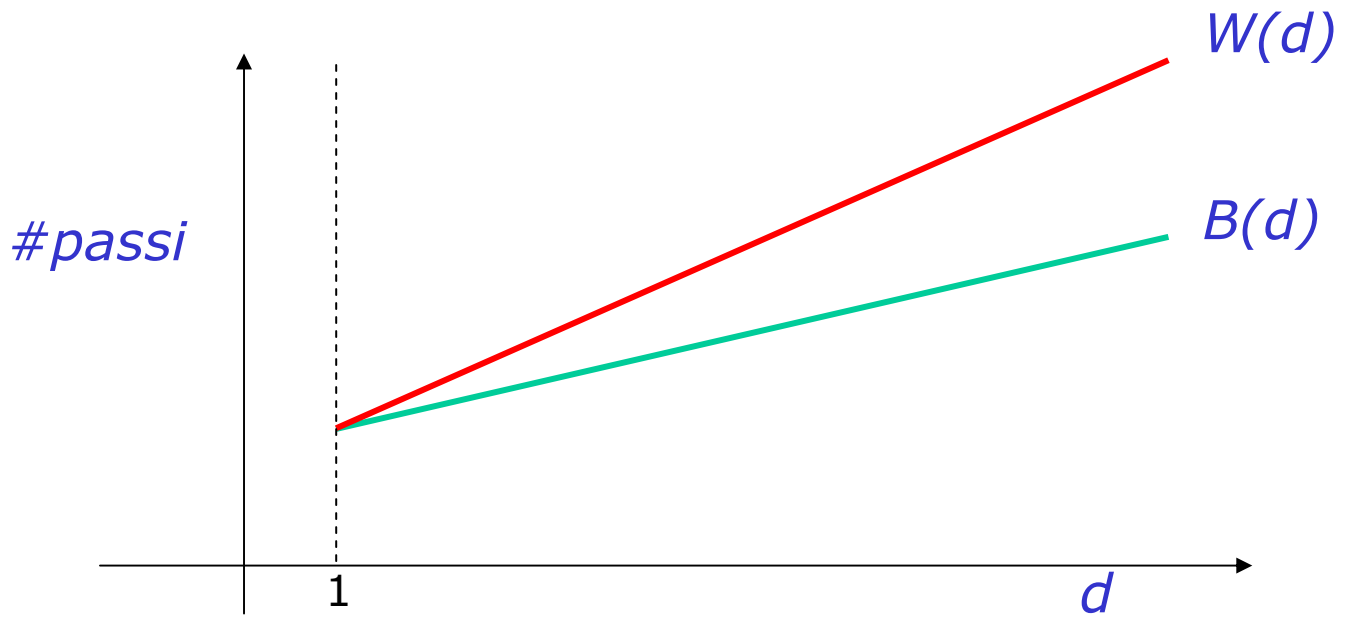
Caso Peggiore e Caso Migliore (Esempio)

- PROBLEMA: CALCOLO DELLA COMPONENTE DI MASSIMO VALORE IN UN VETTORE CON *d=size(I)* COMPONENTI INTERE

NUMERO TOTALE PASSI T :
$2+4(d-1) \leq T \leq 2+6(d-1)$

- COMPLESSITA' NEL *CASO PEGGIORE* $W(d)=2+6(d-1)$

- COMPLESSITA' NEL *CASO MIGLIORE* $B(d)=2+4(d-1)$



Complessita': Caso Peggiore

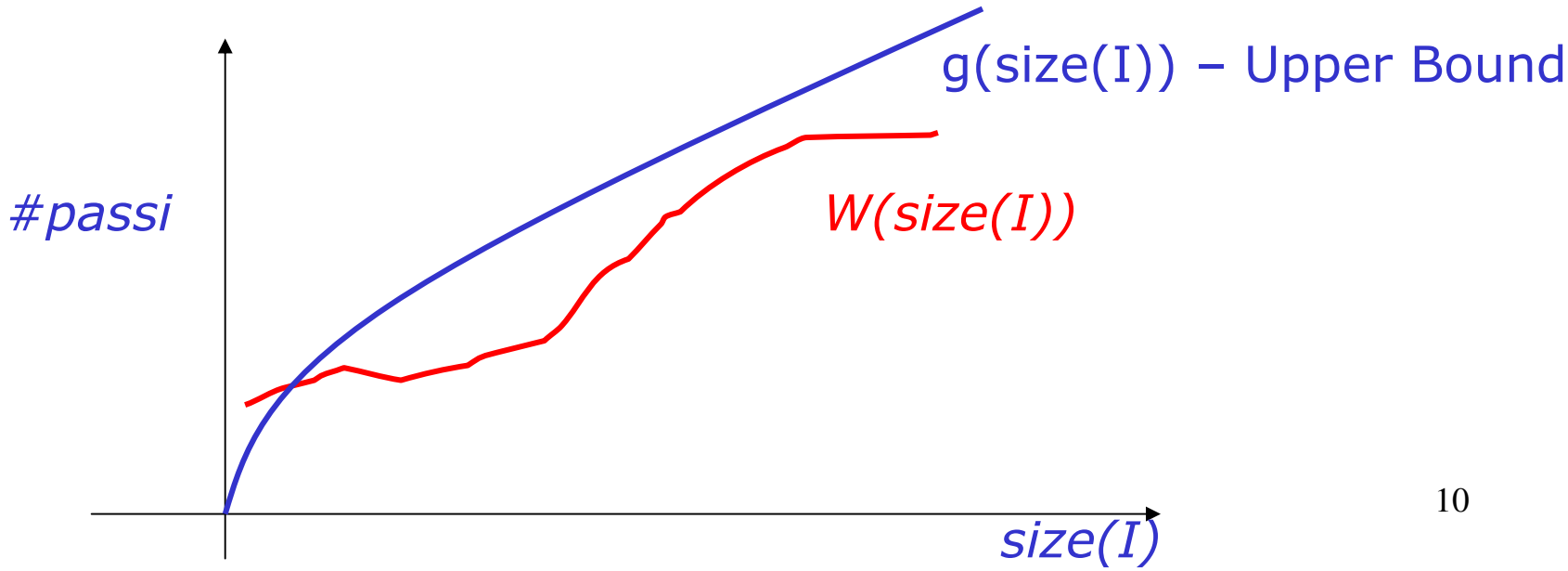
- COMPLESSITA' NEL *CASO PEGGIORE* *maggiormente utilizzata*

✓ *Il caso migliore si ottiene per istanze poco significative*

✓ *Il caso peggiore si ottiene per istanze "patologiche"*

✓ *Misura prudenziale della complessita`*

✓ *Di facile determinazione utilizzando un'approssimazione superiore ("Upper Bound") g(size(I)) della funzione* W(size(I))



Upper Bound: Funzione $O()$ ("ordine di")

$f(x)$ e` $O(g(x)) \Leftrightarrow \exists\, x_0$: per $x \geq x_0$ $f(x) \leq c_1 * g(x)$

$f(x)$ e`ordine di $g(x)$ $[O(g(x))]$ se, per valori di $x$ abbastanza grandi, il valore di $f(x)$ e` definitivamente inferiore a quello di $g(x)$.

$f(x) = x^2 + 100*x + 10000 \rightarrow O(x^2)$

costante c_1

NOTA: se $x \leq 3 \rightarrow f(x) > x^2$ ma... se $x \geq 1000 \rightarrow f(x) < 2x^2$

Regola pratica per calcolare $g(x)$ dato $f(x)$:
- elimina coefficienti e costanti
- conserva l'addendo di $f(x)$ che "cresce" piu` rapidamente

$f(x) = 20*x^2 + \log(x) + 10000 \rightarrow O(x^2)$

$f(x) = 2^x + 20*x^2 + \log(x) + 10000 \rightarrow O(2^x)$

Complessita' Polinomiale

- *Un Algoritmo ha Complessita`nel Caso Peggiore* $O(g(size(I)))$ *se e solo se la funzione* $W(size(I))$ *e`ordine di* $g(size(I))$ *[ovvero:* $g(size(I))$ *e` un "Upper Bound" di* $W(size(I))$*].*

ESEMPIO:

- PROBLEMA: CALCOLO DELLA COMPONENTE DI MASSIMO VALORE IN UN VETTORE CON $d=size(I)$ COMPONENTI INTERE
- COMPLESSITA' NEL *CASO PEGGIORE* $W(d)=2+6(d-1)$ $(O(d))$

- Un Algoritmo ha *Complessita` Polinomiale* se e solo se la sua Complessita`nel Caso Peggiore e` $O(size(I)^k)$ *con* k *costante*

ESEMPIO: **PROGRAMMAZIONE LINEARE**

$size(I) = mn+ n+ m$

Se esiste un algoritmo ***A*** *che risolve la PL in:*
$W(size(I)) = m^4 + n^4 + 4mn + m^3$ *passi*

$W(size(I)) \leq c_1(mn+ n+ m)^4$

$O(size(I)^4)$

A *polinomiale*

Complessita' ed Efficienza

- J. EDMONDS (1965): Un Algoritmo e` *EFFICIENTE ("GOOD")* se e solo se ha *Complessita` Polinomiale.*

- Il valore *k* in $O(size(I)^k)$ e` un *indice di efficienza:*
  [un algoritmo di complessita` $O(size(I)^2)$ e` migliore di un algoritmo di complessita` $O(size(I)^3)$]

- Un Algoritmo ha *Complessita` Esponenziale se la Complessita` nel caso peggiore cresce piu` velocemente di* $size(I)^k$ *per ogni possibile k (ALGORITMO INEFFICIENTE).*

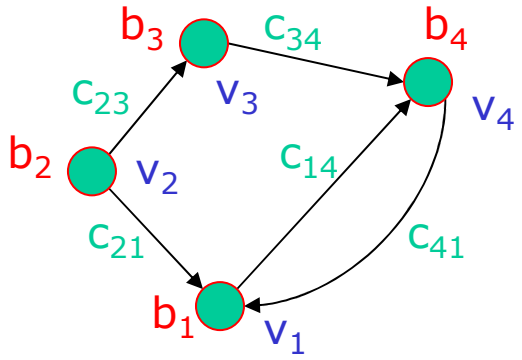
log(n)	n	n^2	n^5	2^n
1	2	4	32	4
2	4	16	1024	16
3	8	64	32768	256
4	16	256	1048576	65536
5	32	1024	33554432	4,29E+09
6	64	4096	1,07E+09	1,84E+19
7	128	16384	3,44E+10	3,4E+38
8	256	65536	1,1E+12	1,16E+77
9	512	262144	3,52E+13	1,3E+154
10	1024	1048576	1,13E+15	#NUM!

$size(I)=n$

Il Caso dei Grafi

ISTANZA

G(N,A) *Grafo orientato*
$[b_h]_{h \in N}$ *parametri di nodo*
$[c_{hk}]_{hk \in A}$ *parametri di arco*



PROBLEMA DELLA RAPPRESENTAZIONE di un Grafo Orientato G(N,A) e dei parametri (lunghezze,capacità,costi...) associati a nodi e archi

IPOTESI: Ogni componente di $[c_{hk}]$ ($[b_h]$) *richiede 1 cella*

Rappresentazione della Struttura delle adiacenze di G(N,A)
$\rightarrow$ *Rappresentazione delle Stelle Uscenti di tutti i nodi di N*

Due modi possibili:

MATRICE DI ADIACENZA:

LISTE DI ADIACENZA:

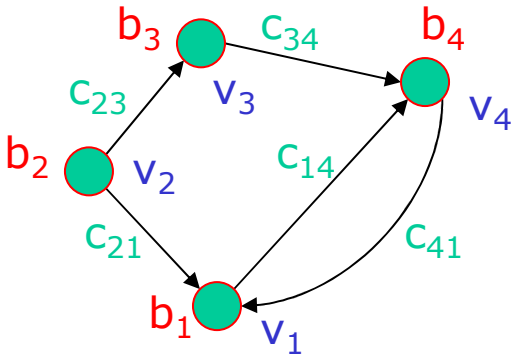
Matrice di Adiacenza

ISTANZA

$G(N,A)$ Grafo orientato

$[b_h]_{h \in N}$ parametri di nodo

$[c_{hk}]_{hk \in A}$ parametri di arco



Rappresentazione del Grafo:
MATRICE DI ADIACENZA:

$\mathcal{A} = [a_{hk}]$ $(N \times N)$

La componente a_{hk} è associata alla coppia ordinata $\{v_h, v_k\}$ di nodi:

$$\begin{cases} a_{hk}=1 & (v_h,v_k) \in A \\ a_{hk}=0 & (v_h,v_k) \notin A \end{cases}$$

	v_1	v_2	v_3	v_4
v_1	-	**0**	**0**	**1**
v_2	**1**	-	**1**	**0**
v_3	**0**	**0**	-	**1**
v_4	**1**	**0**	**0**	-

$\mathcal{A} = [a_{hk}]$

	v_1	v_2	v_3	v_4
v_1	-	-	-	c_{14}
v_2	c_{21}	-	c_{23}	-
v_3	-	-	-	c_{34}
v_4	c_{41}	-	-	-

$[c_{hk}]$

v_1	v_2	v_3	v_4
b_1	b_2	b_3	b_4

$[b_h]$

OCCUPAZIONE DI MEMORIA (celle):

$$\left.\begin{array}{l} \mathcal{A} \rightarrow |N|^2 \\ [c_{hk}] \rightarrow |N|^2 \\ [b_h] \rightarrow |N| \end{array}\right\} \Rightarrow |N|^2 + |N|^2 + |N|$$

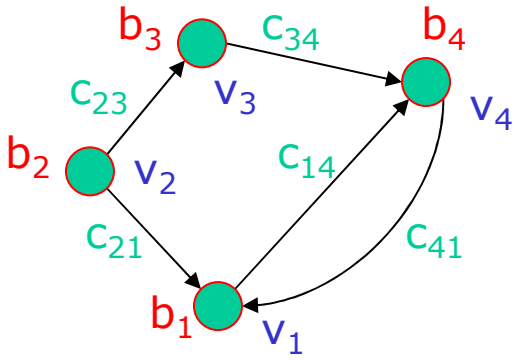
Liste di Adiacenza

ISTANZA

$G(N,A)$ Grafo orientato

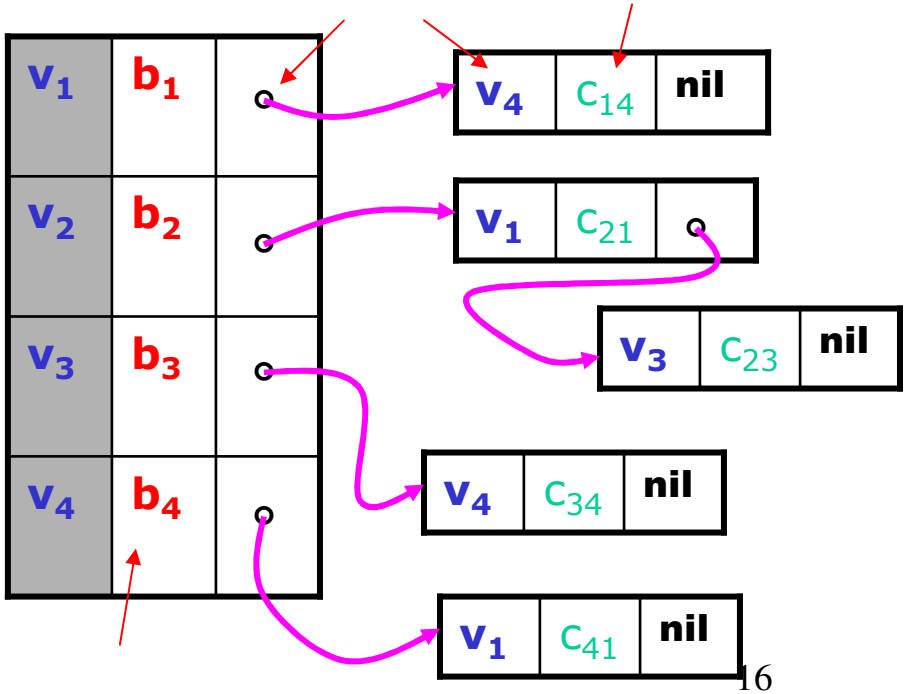
$[b_h]_{h \in N}$ parametri di nodo

$[c_{hk}]_{hk \in A}$ parametri di arco



Tutte le informazioni relative agli archi (i parametri c_{hk} e le adiacenze) sono rappresentate da $|N|$ liste (una per nodo)

Lista $\mathcal{L}(u)$: $u \in N$ rappresenta gli archi della stella uscente di u



OCCUPAZIONE TOTALE DI MEMORIA:

- *Indici di nodo* $\rightarrow |A|$
- $[c_{hk}] \rightarrow |A|$
- *Puntatori arco* $\rightarrow |A|$
- $[b_h] \rightarrow |N|$

$\Rightarrow 3|A| + |N|$

Dimensione di un'Istanza Grafo

ISTANZA

$G(N,A)$ *Grafo orientato*

$[b_h]_{h \in N}$ *parametri di nodo*

$[c_{hk}]_{hk \in A}$ *parametri di arco*

PONIAMO: $|N|=n$ *e* $|A|=m$

Ipotesi: $n \leq m$

DIMENSIONE: size(G(N,A))= f(n, m)

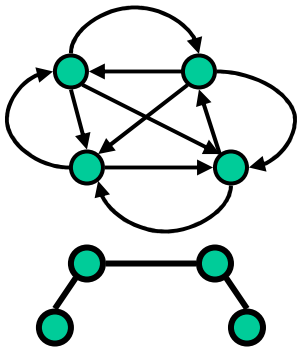
MATRICE DI ADIACENZA: $size_A = 2n^2 + n$

LISTE DI ADIACENZA: $size_L = 3m + n$

OSSERVAZIONI ED ASSUNZIONI

$m \leq n^2$; $m \approx n^2$ *grafo denso*

$m \approx n$ *grafo sparso*



QUALE RAPPRESENTAZIONE E` DA PREFERIRE?

DIPENDE DAL PUNTO DI VISTA e DALLE CARATTERISTICHE DEL GRAFO

Dimensione di un'Istanza Grafo (Complessita`)

MATRICE DI ADIACENZA: $size_A = 2n^2 + n$

LISTE DI ADIACENZA: $size_L = 3m + n$

1. Punto di vista della valutazione della COMPLESSITA`

$size_A = 2n^2 + n \rightarrow O(n^2)$

$size_L = 3m + n \rightarrow O(m)$

ABBIAMO CHE:

Un'istanza grafo ha sempre dimensione (size(G)) dell'ordine di n^2

PONIAMO QUINDI: $size(G)=n^2$

DI CONSEGUENZA:

Un algoritmo su grafi ha Complessita` Polinomiale se e solo se:
$W(size(G))=W(n^2) \leq c1(n^2)^k$ *con k costante*

OVVERO:

*Un algoritmo su grafi ha Complessita` Polinomiale* se e solo se la sua Complessita`nel Caso Peggiore e` $O(n^k)$ *con k costante*

Dimensione di un'Istanza Grafo (in pratica)

MATRICE DI ADIACENZA: $size_A = 2n^2 + n$

LISTE DI ADIACENZA: $size_L = 3m + n$

2. Punto di vista della valutazione dell'EFFICIENZA nella soluzione di istanze specifiche

Se $m \approx n^2$ *(grafo denso)* $\rightarrow$ *MATRICE DI ADIACENZA*

$size_A = O(m) = O(n^2)$; $size_L = O(n^2)$

Se $m \approx n$ *(grafo sparso)* $\rightarrow$ *LISTE DI ADIACENZA*

ESEMPIO: $n = 10^4$ $m = 10^5$

$size_A \approx 10^8 \approx$ *100 milioni di celle*

$size_L \approx 10^5 \approx$ *100.000 celle*