Aniello Murano

# Altri problemi NP-Completi

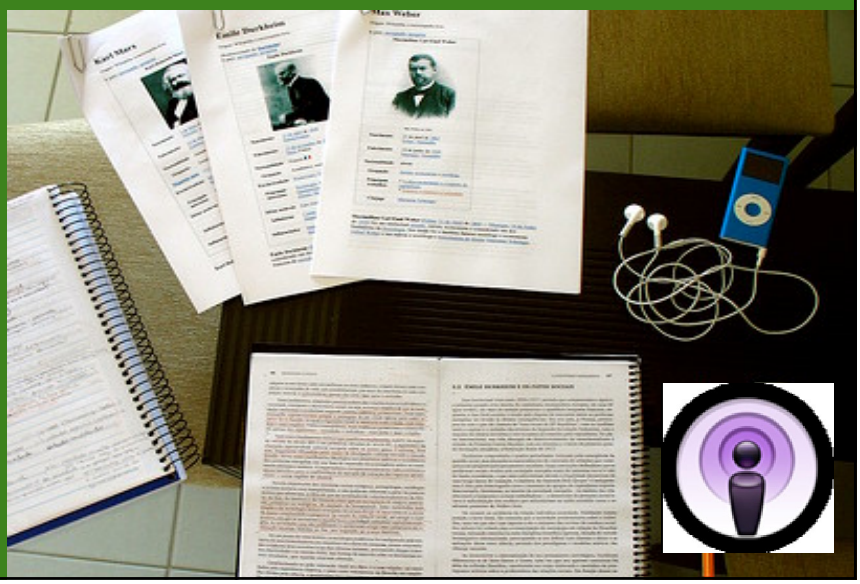
**Lezione n.**16
**Parole chiave:**
Np-complete

**Corso di Laurea:**
Informatica

**Codice:**

**Email Docente:**
murano@ na.infn.it

**A.A.** 2008-2009



## Perché vedere altri problemi Np-complete?

- Per ragioni che non sono ben conosciute, la maggior parte dei problemi NP presenti in natura sono noti per essere o in P o NP-complete.
- Se si cerca un algoritmo in tempo polinomiale per un nuovo problema NP, spendere una buona parte degli sforzi nel cercare di dimostrare la NP-completezza del problema è una cosa sensata perché, se così fosse, l'esperienza insegna che molto probabilmente un algoritm polinomiale per P non sarà mai trovato.
- Questo spiega perché sono importanti le prove di NP-completezza.
- Tali prove consistono nel provare oltre l'appartenenza a P di un problema, una riduzione polinomiale ad esso di un altro problema NP-Completo.
- Per i problemi che vedremo in questa lezione, un buon candidato alla riduzione è 3SAT (l'insieme delle formule booleane 3CNF soddisfacibili).
  - In tali riduzioni, cerchiamo di strutturare il linguaggio sotto esame in modo che possa simulare le variabili e le clausole delle formule Booleane.
  - Ad esempio, quando riduciamo 3SAT a CLIQUE, le variabili sono simulate da vertici, e le clausole da triple.
  - Tali strutture sono spesso chiamati **gadget**.

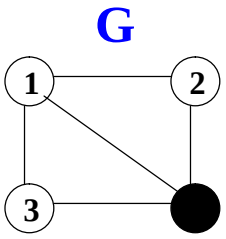
## Np-completezza per Clique

- **Corollario:** **CLIQUE** è NP-completo.
- **Dimostrazione:**
  - Come è già stato (facilmente) dimostrato, CLIQUE è in NP.
  - Con un precedente teorema abbiamo dimostrato che 3SAT è riducibile in tempo polinomiale a CLIQUE, ed inoltre abbiamo anche dimostrato che 3SAT è NP-completo.
  - Per il teorema della NP-completezza abbiamo che se un linguaggio B NP-completo è riducibile in tempo polinomiale ad un linguaggio C in NP, allora C è anchesso NP-completo.
  - Per questo motivo CLIQUE è NP-completo.

- **Si Ricordi che** un linguaggio A NP-completo se solo se:
  - A è in NP.
  - Ogni altro linguaggio NP è riducibile ad A (se prendiamo un linguaggio A NP-completo, allora basta solo questo per provare questa proprietà).

## Problema di definire un Vertex Cover

- Se G è un grafo, una copertura dei vertici di G (**Vertex Cover**) è un sottoinsieme di nodi in cui ogni arco di G tocchi (almeno) uno di questi nodi.




- **VERTEX-COVER =**
  **{<G,k> | G è un grafo non direzionato che ha un sottoinsime G′ di cardinalità k che è un vertex cover}**

**<G,4> ∈ VERTEX-COVER?**

**<G,3> ∈ VERTEX-COVER?**

**<G,2> ∈ VERTEX-COVER?**

**<G,1> ∈ VERTEX-COVER?**

- Nota: Si può facilmente dimostrare che se <G,k> è un vertex cover allora lo è anche <G,m> per ogni m>k.

## Np-completezza di Vertex Cover

- **Teorema:** VERTEX-COVER è NP-completo.
- **Dimostrazione:**
  - È facile dimostrare che Vertex-cover è in NP [esercizio per gli studenti].
  - Per provare l'hardness, costruiamo una riduzione in tempo polinomiale da 3SAT a VERTEX-COVER.
  - La riduzione deve quindi "mappare" una formula 3CNF $\phi$ in un grafo G e un valore k, tale che $\phi$ è soddisfacibile sse G ha un Vertex Cover di taglia k.
  - Per ogni variabile x in $\phi$, costruiamo un arco orizzontale che connette due nodi che marchiamo con i "gadget" x e $\bar{x}$.
  - Chiamiamo ognuno di questi sottografi "**variabile-gadget**".
  - Intuitivamente, settare x a true corrisponde a selezionare il nodo di sinistra per il Vertex Cover, mentre falso corrisponde a selezionare il nodo di destra.
  - Per esempio, se

$$\phi = (x \vee x \vee z) \wedge (\bar{x} \vee \bar{z} \vee \bar{z}) \wedge (\bar{x} \vee z \vee z),$$

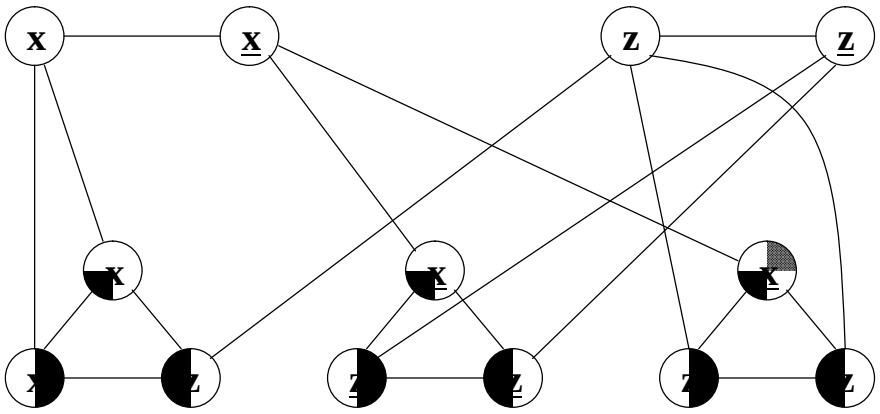
la riduzione produce i seguenti due archi:


## Np-completezza di Vertex Cover 2

$$\phi = (x \vee x \vee z) \wedge (\bar{x} \vee \bar{z} \vee \bar{z}) \wedge (\bar{x} \vee z \vee z)$$

- Per ciascuna clausola, si produce un "triangolo" di interconnessione tra i nodi.
- Ciascun nodo del triangolo è marcato con un letterale della clausola.
- Ogni triangolo rappresenta una "**clausola gadget**".

Infine, connettiamo ciascun nodo "variabile-gadget" con ciascun nodo "clausola-gadget" che hanno identica marcatura.

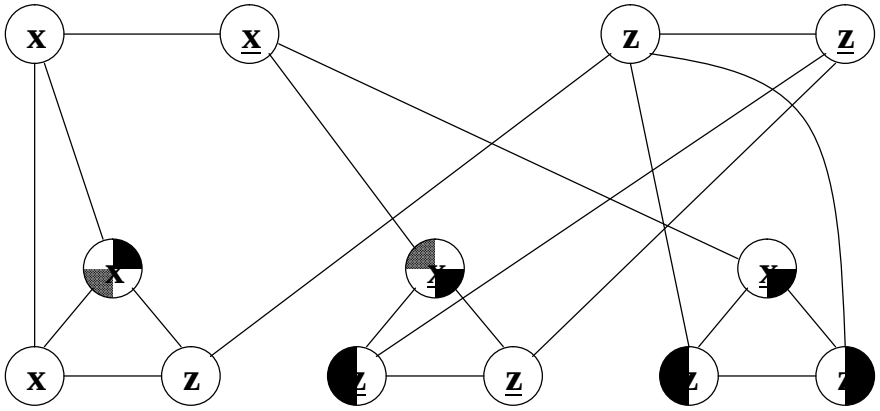


## Np-completezza di Vertex Cover 3

- Si noti che il grafo G così costruito avrà 2m+3i nodi, dove m è il numero di variabili in $\phi$ e i è il numero di clausole.
- k sarà invece m+2i. Nell'esempio visto avremo che k=2+(2*3)=8.

- **Claim:** $\phi$ è soddisfacibile sse **G** ha un Vertex Cover di taglia **k**.

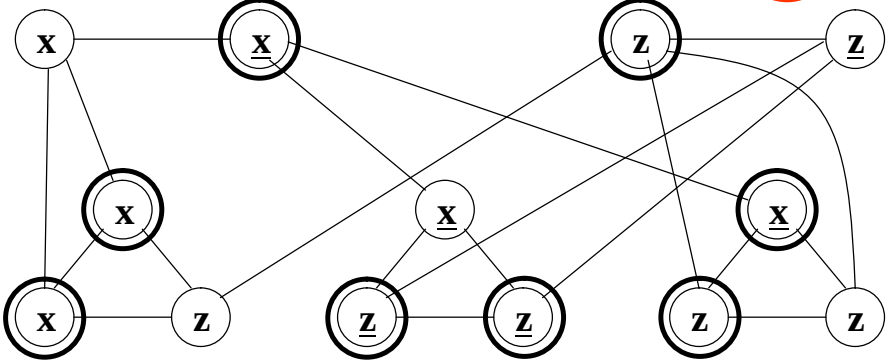
$$\phi = (x \vee x \vee z) \wedge (\underline{x} \vee \underline{z} \vee \underline{z}) \wedge (\underline{x} \vee z \vee z)$$




## Np-completezza di Vertex Cover 4

- Supponiamo che $\phi$ sia soddisfacibile.
- Si considerino i nodi di variabili-gadget nel Vertex Cover che corrispondono a letterali "veri" nella formula.
- Per ciascuna "clausola-gadget", si selezioni un nodo letterale "marcato vero", e si includino gli altri due nodi nel Vertex Cover. Otteniamo così un numero totale di k nodi.
- Tutti gli archi di queste variabili e clausole gadget sono ovviamente "cover" (coperti). Inoltre, ciascun arco tra i gadget è coperto da un nodo (vero) di variabile gadget, o da uno dei due nodi della clausola gadget che è stata inclusa nel Vertex Cover.

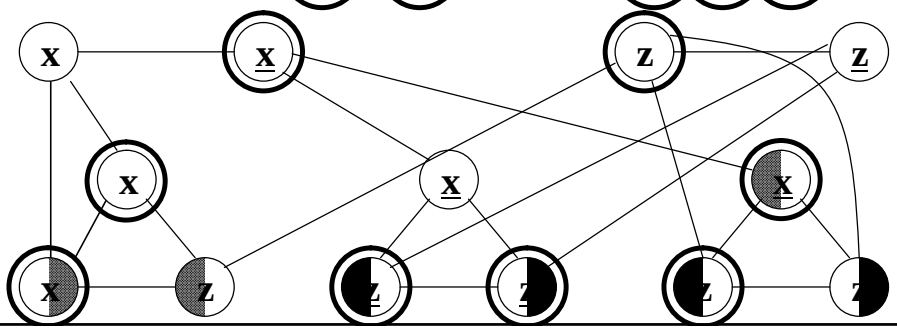
$$\phi = (x \vee x \vee z) \wedge (\underline{x} \vee \underline{z} \vee \underline{z}) \wedge (\underline{x} \vee z \vee z)$$



## Np-completezza di Vertex Cover 5

- Supponiamo ora che G ha un VC di taglia k. Chiaramente, deve includere almeno un nodo da ciascuna variabile gadget (per coprire l'arco corrispondente), e almeno due nodi per ciascuna clausola gadget (altrimenti un arco rimane non coperto).
- Siccome la somma di questi numeri è pari a k, questi nodi sono **esattamente** i nodi del vertex-cover (non servono altri nodi e non possiamo prendere nodi differenti).
- Prendiamo i nodi delle variabili gadget che sono nel VC e assegniamo a true i corrispondenti letterali nella formula. Questo è un assegnamento soddisfacibile per $\phi$ perchè, se si prende una qualsiasi clausola gadget, il nodo che non è in VC è sicuramente connesso tramite un arco a un nodo letterale vero di una variabile gadget, altrimenti quell'arco tra gadget non sarebbe coperto.

$$\phi = (x \vee x \vee z) \wedge (\underline{x} \vee \underline{z} \vee \underline{z}) \wedge (\underline{x} \vee z \vee z)$$

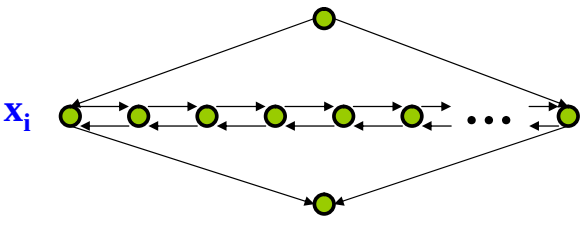



## Np-completezza di Hampath 1

- **Teorema:** **HAMPATH** è NP-completo.
  - **Dimostrazione:** Che Hampath è in NP è già stato dimostrato. Mostriamo ora una riduzione in tempo polinomiale da 3SAT ad Hampath. Consideriamo la seguente formula 3cnf generica $\phi$ con k clausole:

$$\phi = \underbrace{(p_1 \vee q_1 \vee r_1)}_{c_1} \wedge \underbrace{(p_2 \vee q_2 \vee r_2)}_{c_2} \wedge \ldots \wedge \underbrace{(p_k \vee q_k \vee r_k)}_{c_k}$$

Assumiamo che $\phi$ contiene **l** variabili $x_1,\ldots,x_l$. Ciascuna variabile sarà rappresentata con una struttura a forma di diamante con 3k+5 nodi (più avanti vi sarà chiaro perchè questo numero):

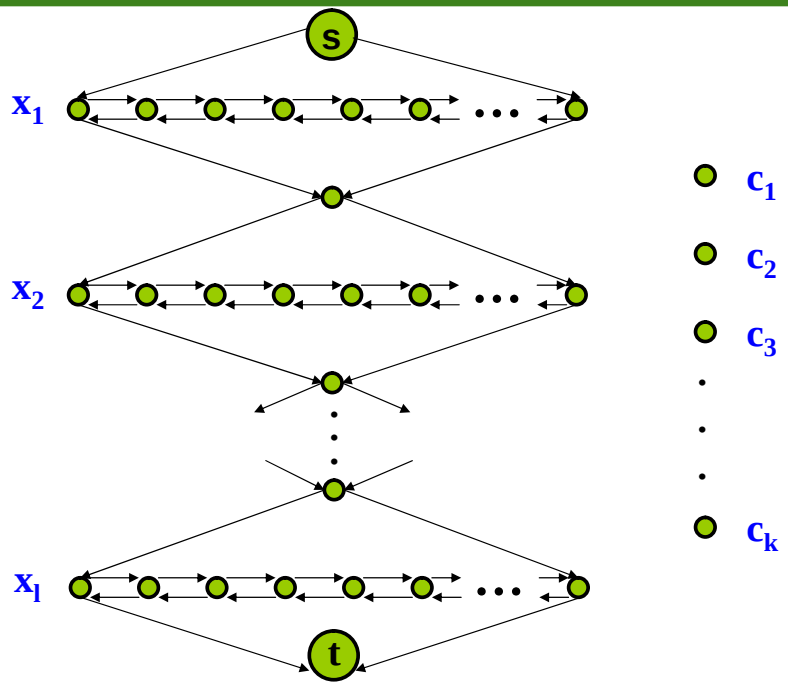



Ciascuna clausola sarà rappresentata con un singolo nodo:

## Np-completezza di Hampath 2

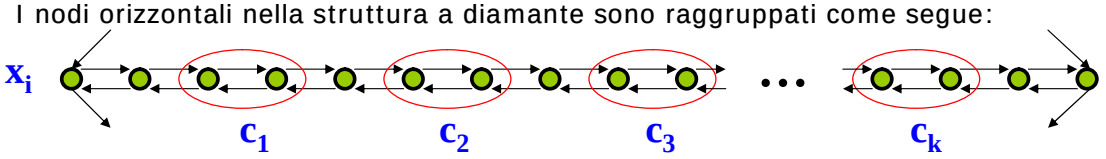
La struttura complessiva di G avrà una forma del tipo mostrato a lato.

Ci sono degli archi aggiuntivi che connettono i nodi delle clausole con nodi interni alla struttura a diamante.Nella prossima diapositiva vedremo il perchè.

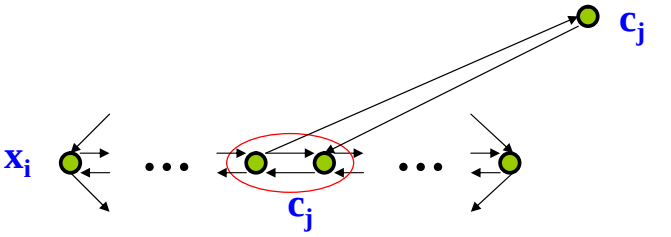



## Np-completezza di Hampath 3

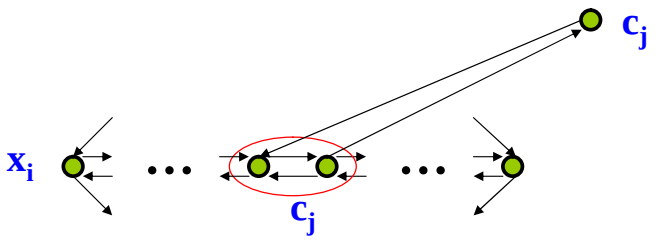
I nodi orizzontali nella struttura a diamante sono raggruppati come segue:




Questi sono gli archi aggiuntivi relativi a quando le clausole $c_j$ contengono $x_i$ :




Questi sono gli archi aggiuntivi corrispondenti a quando le clausole $c_j$ contengono $\underline{x_i}$ :



## Np-completezza di Hampath 4

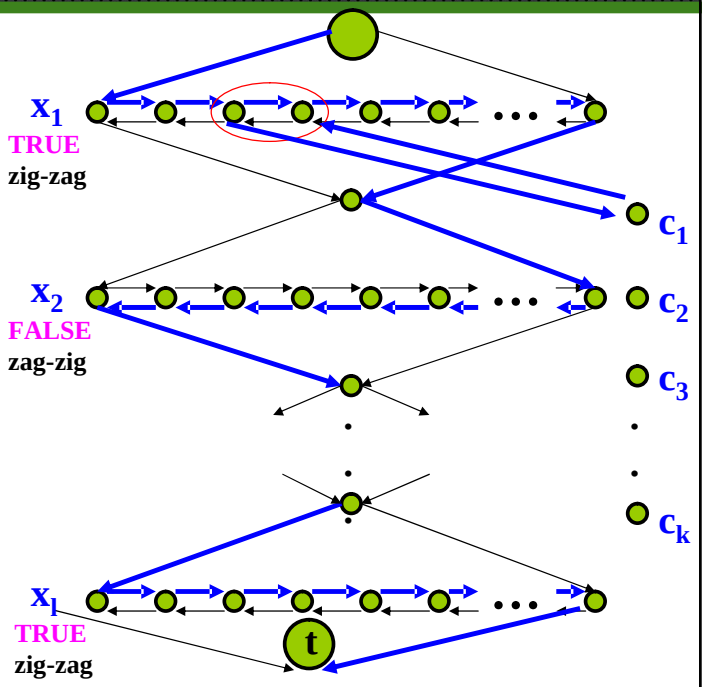
Si supponga che $\phi$ sia soddisfacibile

Si consideri il percorso che fa "zig-zag" attraverso i nodi true del diamante, e "zag- zig" attraverso i nodi false del diamante.

Questo percorso copre tutti i nodi eccetto i nodi della clausola $c_1,\ldots,c_k$.

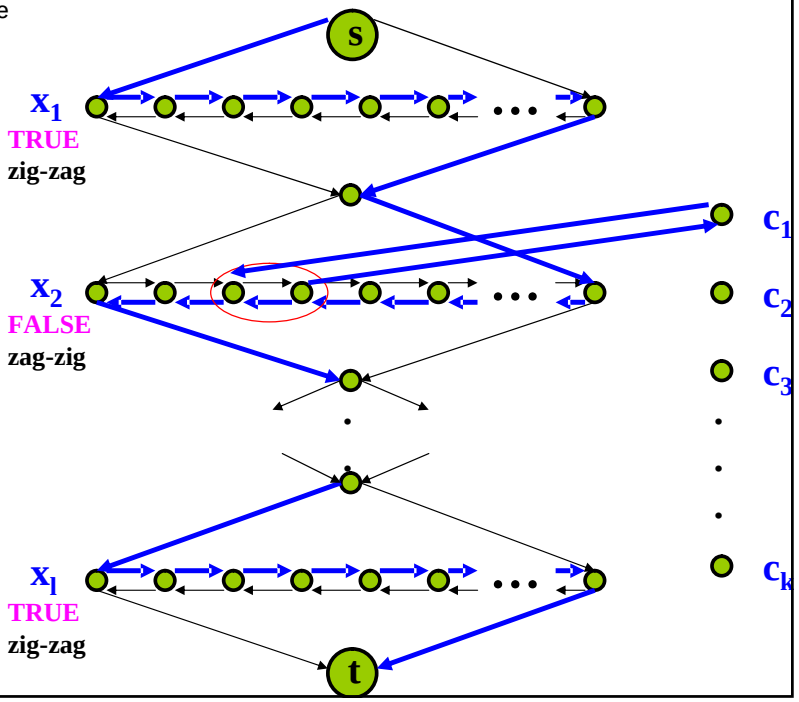
Possiamo facilmente includerli selezionando un letterale vero in ciascuna clausola e aggiungendo una "deviazione" dalla corrispondente coppia di nodi orizzontali del corrispondente diamante.

Per esempio quando $c_1$ include x e $x_1$ è selezionata per $c_1$, abbiamo gli archi dal primo diamante a $c_1$ e viceversa



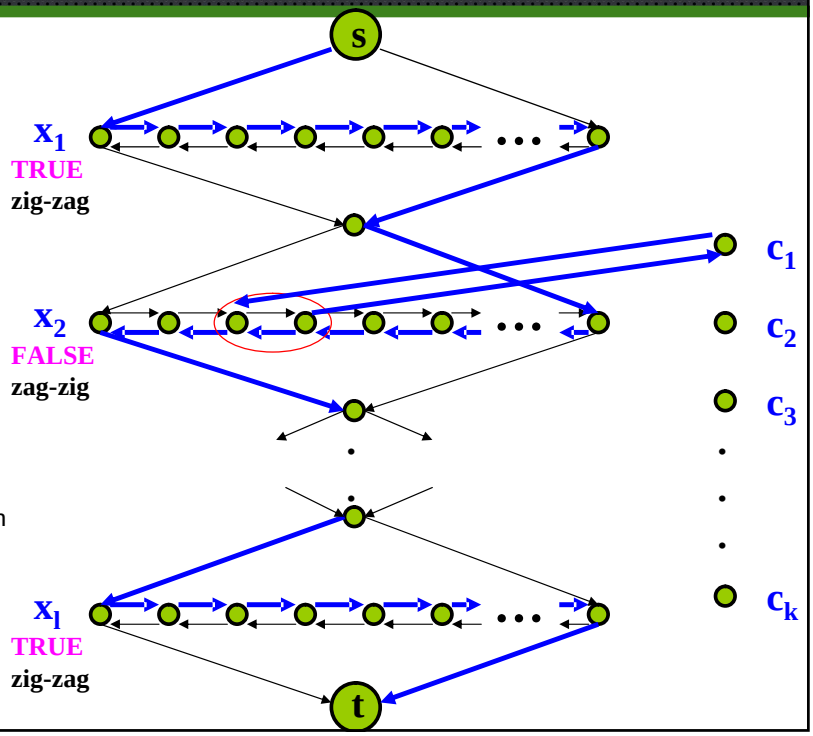

## Np-completezza di Hampath 5

- Dall'altro lato, se $c_1$ include $\underline{x_2}$, selezioneremo $c_1$ con una deviazione, ma in senso opposto
- E così via...
- Il path risultante con le deviazioni è ovviamente Hamiltoniano.



## Np-completezza di Hampath 5 bis

Per la direzione inversa assumiamo che G ha un path Hamiltoniano da S a t. Con un pò di immaginazione, possiamo vedere che il path deve essere "*normalizzato*", significa che va immaginato dal primo in alto all'ultimo in basso, eccetto che per la deviazione dei nodi della clausola. Da questo percorso Normalizzato possiamo facil-mente ottenere un assegnamento soddisfacibile: se attraversiamo il diamante in zig-zag verso sinistra assegnamo true alle corrispondenti varibiabili. Altrimenti le assegnamo false.




## Np-completezza di Uhampath

- **UHAMPATH** è la versione non direzionata di **HAMPATH**
- **Teorema: UHAMPATH** è NP-completo.
  - **Dimostrazione:** Riduciamo Hampath ad Uhampath. Da un grafo direzionato G, costruiamo il suo equivalente G′ non direzionato sostituendo ciascun nodo u di G con tre nodi $u_{in}$ $u_{mid}$ e $u_{out}$; eccetto il nodo s che viene sostituito con $s_{out}$, e il nodo t che viene sostituito dal nodo $t_{in}$. G′ ha due tipi di archi. Un tipo connette ciascun nodo $u_{mid}$, e $u_{out}$. Il secondo connette ciasciun nodo $u_{in}$ con $u_{ou}$, finchè c'è un arco da u a v in G. Supponiamo che G ha un path Hamiltoniano

    **s, u1, u2, …, uk, t**

    Allora ovviamente segue che G′ ha un path Hamiltoniano

    $s_{out}, u_{1in}, u_{1mid}, u_{1out}, u_{2in}, u_{2mid}, u_{2out}, \ldots, u_{kin}, u_{kmid}, u_{kout}, t_{in}$

    Ugualmente vale per l'altra direzione.

## Np-completezza di Subset-Sum 1

- **Teorema:** Subset-Sum è Np-completo.
- **Dimostrazione:**
  - Che Subset-sum appartine a NP è già stato dimostrato.
  - Riduciamo adesso in tempo polinomiale 3SAT a Subset-Sum.
  - Sia $\phi$ una formula 3-cnf booleana con $x_1,\dots,x_l$ variabili e $c_1,\dots,c_k$ clausole.
  - Costruiamo un tabella di decimali con (l+k) colonne e (2l+2k+1) righe .
  - Le colonne sono marcate con $x_1,\dots,x_l,c_1,\dots,c_l$, e le righe sono marcate con $y_1,z_1,\dots,y_l,z_l,g_1,h_1,\dots g_k,h_k$, t.
  - Il contenuto delle righe sono numeri decimali. Le prime 2l+2k righe, con numeri di S, e l'ultima riga è il numero target t.
  - Mostreremo in seguito che S ha un sottoinsieme che sommato da t sse $\phi$ è soddisfacibile.
  - In particolare, la riga/numero t consiste di l uno seguiti da k tre.
  - Ciascuna riga $y_i$ ha un 1 nella colonna $x_i$, e così in ciascuna colonna $c_j$ tale che la clausola $c_j$ contiene xi.
  - Ciascuna riga $z_i$ ha un 1 nelle colonne $x_i$, e così in ciascuna colonna $c_j$ tale che la clausola $c_j$ contiene $\underline{\mathbf{x}}\mathbf{i}$.
  - Infine, ciascuna riga $g_i$ e $h_i$ ha un 1 nelle colonne $c_i$.
  - Le cifre non specificate sono da sottintendersi a 0.

## Np-completezza di Subset-Sum 2

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | … | $x_l$ | $c_1$ | $c_2$ | … | $c_k$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 1 | | | | | | 1 | | | |
| $z_1$ | 1 | | | | | | | | | |
| $y_2$ | | 1 | | | | | | 1 | | |
| $z_2$ | | 1 | | | | | 1 | | | |
| $y_3$ | | | 1 | | | | 1 | 1 | | |
| $z_3$ | | | 1 | | | | | | | 1 |
| … | | | | | | | | | | |
| $y_l$ | | | | | | 1 | | | | |
| $z_l$ | | | | | | 1 | | | | |
| $g_1$ | | | | | | | 1 | | | |
| $h_1$ | | | | | | | 1 | | | |
| $g_2$ | | | | | | | | 1 | | |
| $h_2$ | | | | | | | | 1 | | |
| … | | | | | | | | | | |
| $g_k$ | | | | | | | | | | 1 |
| $h_k$ | | | | | | | | | | 1 |
| t | 1 | 1 | 1 | 1 | … | 1 | 3 | 3 | … | 3 |

$$(x_1 \vee \underline{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \dots) \wedge \dots \wedge (\underline{x}_3 \vee \dots \vee \dots)$$

- Ciascuna riga $y_i$ ha un 1 nella colonna $x_i$ se $c_j$ contiene $x_i$.
- Ciascuna riga $y_i$ ha un 1 nella colonna $c_j$ se $c_j$ contiene $x_i$ vera.
- Ciascuna riga $z_i$ ha un 1 nelle colonne $x_i$, e così in ciascuna colonna $c_j$ tale che la clausola $c_j$ contiene $\underline{\mathbf{x}}\mathbf{i}$.
- Infine, ciascuna riga $g_i$ e $h_i$ ha un 1 nelle colonne $c_i$.
- Le cifre non specificate sono da sottintendersi a 0.

## Np-completezza di Subset-Sum 3

- Supponi $\phi$ ha un assegnamento soddisfacibile. Costruiamo S come segue:

  Seleziona yi se xi è true, e selziona zi se xi è false. Se sommiamo quello che abbiamo selezionato fin ora otteniamo un 1 in ciascuna delle prime l cifre perché abbiamo selezionato le yi o zi per ciascun i.

  Inoltre, ciascuna delle ultime k cifre è un numero compreso tra 1 e 3 perché ciascuna clausola è soddisfacibile e così contengono litterali veri tra 1 e 3.
  In dipendenza di questo numero, selezioniamo ulteriormente i numeri di g e h per portare ciascuno delle ultime k cifre a 3.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | … | $x_l$ | $c_1$ | $c_2$ | … | $c_k$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 1 | | | | | | 1 | | | |
| $z_1$ | 1 | | | | | | | | | |
| $y_2$ | | 1 | | | | | | 1 | | |
| $z_2$ | | 1 | | | | | 1 | | | |
| $y_3$ | | | 1 | | | | 1 | 1 | | |
| $z_3$ | | | 1 | | | | | | | 1 |
| … | | | | | | | | | | |
| $y_l$ | | | | | | 1 | | | | |
| $z_l$ | | | | | | 1 | | | | |
| $g_1$ | | | | | | | 1 | | | |
| $h_1$ | | | | | | | 1 | | | |
| $g_2$ | | | | | | | | 1 | | |
| $h_2$ | | | | | | | | 1 | | |
| … | | | | | | | | | | |
| $g_k$ | | | | | | | | | | 1 |
| $h_k$ | | | | | | | | | | 1 |
| t | 1 | 1 | 1 | 1 | … | 1 | 3 | 3 | … | 3 |

## Np-completezza di Subset-Sum 3

- Supponiamo che la somma di S dia t. Osserviamo che ciascuna colonna nella tabella che descrive S ha al più cinque 1.
- Per ottenere un 1 in ciascuna delle prime l colonne di t, abbiamo bisogno o delle yi o delle zi per ciascuna i.
- Se abbiamo yi, assegnamo xi a true, altrimenti a false.
- Questo assenamento deve soddisfare $\phi$ perchè in ciascuna delle colonne finali k la somma è sempre 3.
- Nella colonna cj, al massimo 2 possono venire da gj e hj, così almeno 1 in questa colonna dovrebbero provenire da qualche yi o da qualche zi nel sottoinsieme. Se abbiamo yi, allora xi appare a cj e allora cj è assegnata a true, così cj è soddisfacibile. Se abbiamo zi, allora xi appare in cj e xi è assegnato a false, così cj è soddisfacibile.
- Pertanto $\phi$ è soddisfacibile.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | … | $x_l$ | $c_1$ | $c_2$ | … | $c_k$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $y_1$ | 1 | | | | | | 1 | | | |
| $z_1$ | 1 | | | | | | | | | |
| $y_2$ | | 1 | | | | | | 1 | | |
| $z_2$ | | 1 | | | | | 1 | | | |
| $y_3$ | | | 1 | | | | 1 | 1 | | |
| $z_3$ | | | 1 | | | | | | | 1 |
| … | | | | | | | | | | |
| $y_l$ | | | | | | 1 | | | | |
| $z_l$ | | | | | | 1 | | | | |
| $g_1$ | | | | | | | 1 | | | |
| $h_1$ | | | | | | | 1 | | | |
| $g_2$ | | | | | | | | 1 | | |
| $h_2$ | | | | | | | | 1 | | |
| … | | | | | | | | | | |
| $g_k$ | | | | | | | | | | 1 |
| $h_k$ | | | | | | | | | | 1 |
| t | 1 | 1 | 1 | 1 | … | 1 | 3 | 3 | … | 3 |