

Algoritmi e Strutture Dati

Lezione 34

12 dicembre 2025

CLASSE

P

P = Classe dei problemi di decisione risolvibili

in tempo polinomiale (da algoritmi deterministici)

CLASSE NP

NP = Classe dei problemi di decisione risolvibili

in tempo polinomiale da algoritmi nondeterministici

= Classe dei problemi di decisione con certificati

verificabili in tempo polinomiale

SODD, CLIQUE, PARTIZIONE \in NP

PROBLEMI NP

ALGORITMO

FASE NON DETERMINISTICA costruzione del certificato

FASE DETERMINISTICA

verifica del certificato
in tempo polinomiale

$$P \subseteq NP$$

$$P = NP?$$

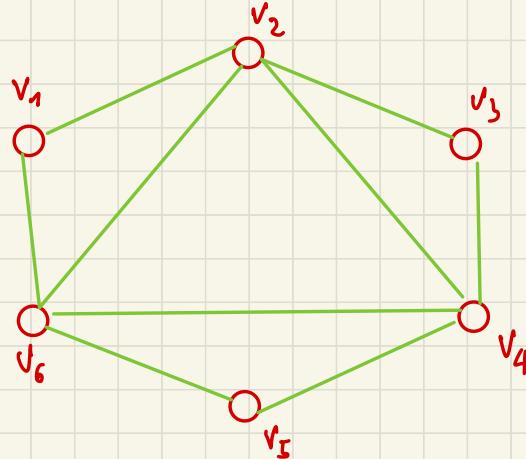
da CLIQUE a INSIEME INDIPENDENTE

Problema CLIQUE

Istanza Grafo non orientato $G = (V, E)$

Intero $K \geq 0$

Quesione \exists sottografo completo di G con K vertici?

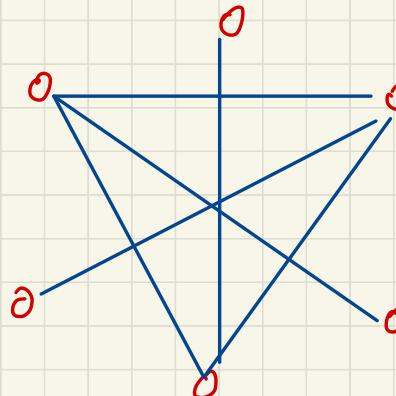


Problema INSIEME INDIPENDENTE

Istanza Grafo non orientato $G = (V, E)$

Intero $K \geq 0$

Quesione $\exists V' \subseteq V$ con $\#V' = K$ t.c. i vertici in V' non hanno archi che li collegano direttamente?



da SODD a CLIQUE

Problema SODDISFACIBILITÀ (SODD)

Istanza Formula booleana ϕ in forma normale congiuntiva
con insieme di variabili V

Quesione \exists un assegnamento alle variabili in V che rende vera ϕ ,
cioè t.c. $\phi(f)=1$?

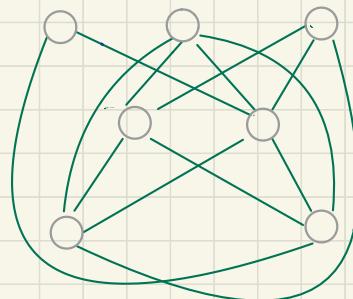
Problema CLIQUE

Istanza Grafo non orientato $G=(V,E)$

Intero $K \geq 0$

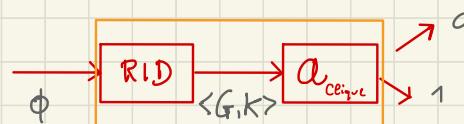
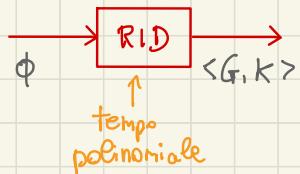
Quesione \exists sottografo completo di G con K vertici?

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



$K = 3$

dai SODD a CLIQUE



a_{Sodd}

risolve SODD !

ALGORITMO che risolve CLIQUE



ALGORITMO $a_{\text{Sodd}} (\phi \text{ formula}) \rightarrow \text{boolean}$
 $\langle G, K \rangle \leftarrow \text{R.I.D.}(\phi)$
 $r \leftarrow a_{\text{clique}} (\langle G, K \rangle)$
RETURN r

Inoltre:

Se a_{clique} lavora in tempo polinomiale
anche a_{Sodd} lavora in tempo polinomiale

Se $\text{CLIQUE} \in P$
allora $\text{SODD} \in P$

RIDUCIBILITA' TRA PROBLEMI

$$\Pi_1 : I_1 \rightarrow \{0, 1\}$$

$$\Pi_2 : I_2 \rightarrow \{0, 1\}$$

problemi di decisione

Π_1 è riducibile a Π_2 se $\exists f : I_1 \rightarrow I_2$ t.c.

$$\forall x \in I_1 \quad \Pi_1(x) = \Pi_2(f(x))$$

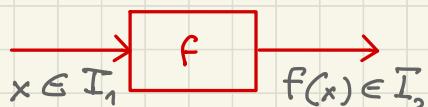
↑
riduzione



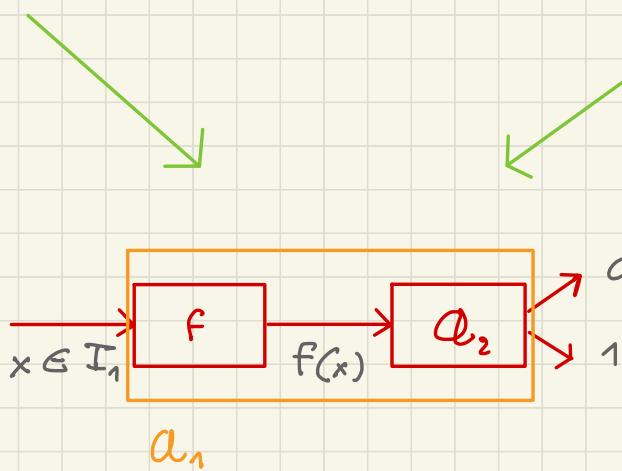
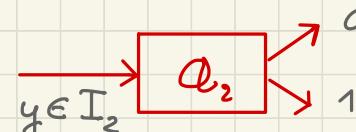
RIDUCIBILITA' TRA PROBLEMI

Supponiamo:

a) Π_1 riducibile a Π_2



b) L'algoritmo α_2 per Π_2



ALGORITMO $\alpha_1 (x \in I_1) \rightarrow \text{boolean}$
 $y \leftarrow f(x)$
 $r \leftarrow \alpha_2(y)$
RETURN r

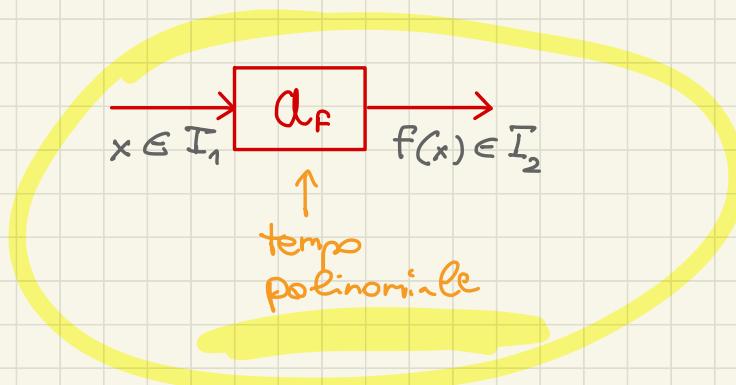
α_1 è un algoritmo per Π_1

RIDUCIBILITA' POLINOMIALE

Π_1 è riducibile POLINOMIALMENTE a Π_2 ($\Pi_1 \leq_p \Pi_2$)

se \exists riduzione $f: I_1 \rightarrow I_2$ calcolabile

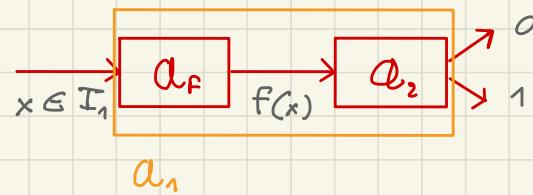
da un algoritmo A_f in Tempo polinomiale



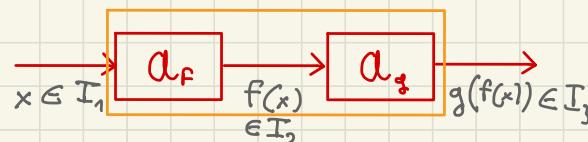
RIDUCIBILITÀ POLINOMIALE

Proprietà fondamentali

① Se $\Pi_1 \leq_p \Pi_2$ e $\Pi_2 \in P$ allora $\Pi_1 \in P$



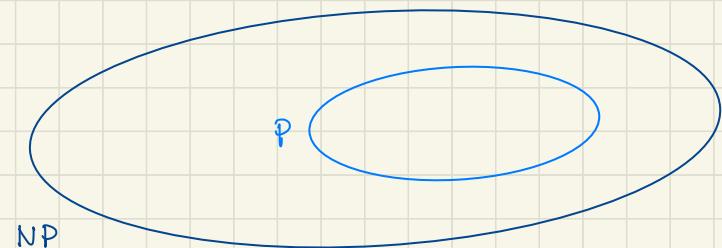
② Se $\Pi_1 \leq_p \Pi_2$ e $\Pi_2 \leq_p \Pi_3$ allora $\Pi_1 \leq_p \Pi_3$



PROBLEMI NP-HARD e NP-COMPLETI

- Π è NP-HARD (\circ NP-DIFFICILE) se ogni problema $\Pi' \in NP$ è riducibile polinomialmente a Π
i.e., $\forall \Pi' \in NP \quad \Pi' \leq_p \Pi$

- Π è NP-COMPLETO se:
 - Π è NP-HARD e
 - $\Pi \in NP$

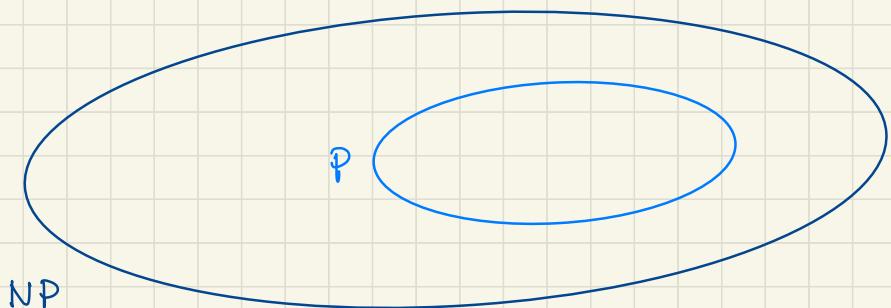
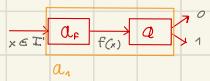


PROBLEMI NP-completi

Teorema Sia Π un problema NP-completo.

- Se $\Pi \notin P$ allora $P \neq NP$
- Se $\Pi \in P$ allora $P = NP$

① Se $\Pi' \leq_p \Pi$ e $\Pi \in P$ allora $\Pi' \in P$



TEOREMA DI COOK

SODD è NP-completo

Dim

[1] SODD ∈ NP

[2] $\forall \Pi' \in \text{NP} \quad \forall \Pi' \leq_p \text{SODD}$

CLIQUE è NP-completo

Dim

1 CLIQUE ∈ NP

2 Abbiamo visto che SODD \leq_p CLIQUE

Poiché SODD è NP-completo: $\forall \Pi' \in P \quad \Pi' \leq_p \text{SODD}$

Proprietà
 $\Pi_1 \leq_p \Pi_2$ e $\Pi_2 \leq_p \Pi_3$ allora $\Pi_1 \leq_p \Pi_3$

$\Pi' \leq_p \text{CLIQUE}$

e'
NP-hard

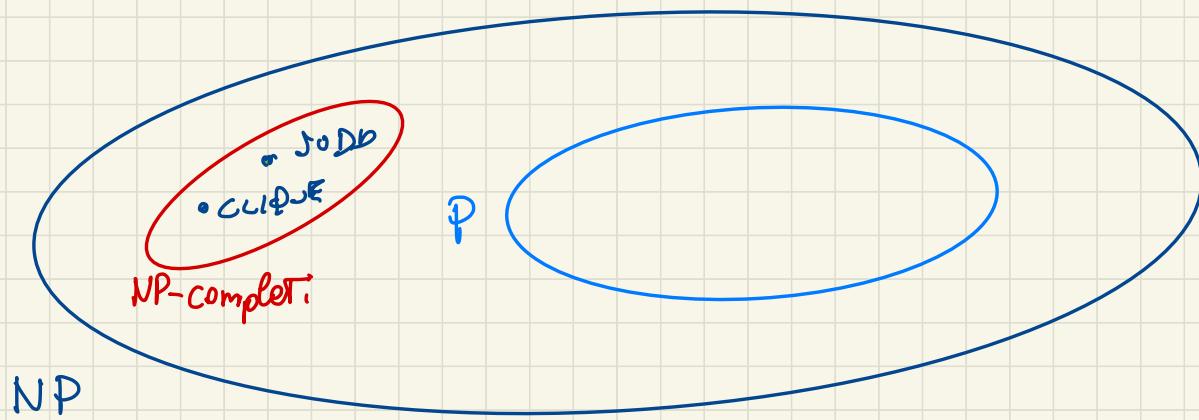
CLIQUE
NP-hard

In generale

Se Π è NP-completo e $\Pi \leq_p \tilde{\Pi}$ con $\tilde{\Pi} \in NP$

allora $\tilde{\Pi}$ è NP-completo

Se $P \neq NP$:



ALCUNI PROBLEMI NP-COMPLETI

CAMMINO HAMILTONIANO

Istanza

Grafo $G = (V, E)$

Quesione

\exists cammino che visita tutti i vertici
del grafo una e una sola volta ?

CAMMINO PIÙ LUNGO

Istanza $G = (V, E, \omega)$ grafo pesato

$K \in \mathbb{N}$

Quesione \exists cammino semplice in G di lunghezza $\geq K$?

CAMMINO HAMILTONIANO

$G = (V, E)$



CAMMINO PIÙ LUNGO

$G = (V, E, \omega)$

$\omega(x, y) = 1 \quad \forall (x, y) \in E$

$K = n - 1$

3 - SODD

Istanza Formula booleana ϕ in f.n. congiuntiva con clausole
di lunghezza 3

Queszione ϕ è soddisfacibile?

$$x \vee y \vee z \vee w \equiv (x \vee y \vee z) \wedge (\bar{z} \vee z \vee w)$$

$$x \vee y \vee z \vee w \vee k \equiv (x \vee y \vee z) \wedge (\bar{z} \vee z \vee b) \wedge (\bar{b} \vee w \vee k)$$

3 - SODD è NP-completo

2 \rightarrow add $(x \vee \bar{y}) \wedge (x \vee \bar{z}) \wedge \dots \in P$

f.n. disgiuntiva $(x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge \bar{y}) \vee (x \wedge \bar{z} \wedge y) \in P$
SODD \rightarrow DSODD costo?

PARTIZIONE

Istanza Insieme finito A di oggetti

F₇ peso $s: A \rightarrow \mathbb{N}$

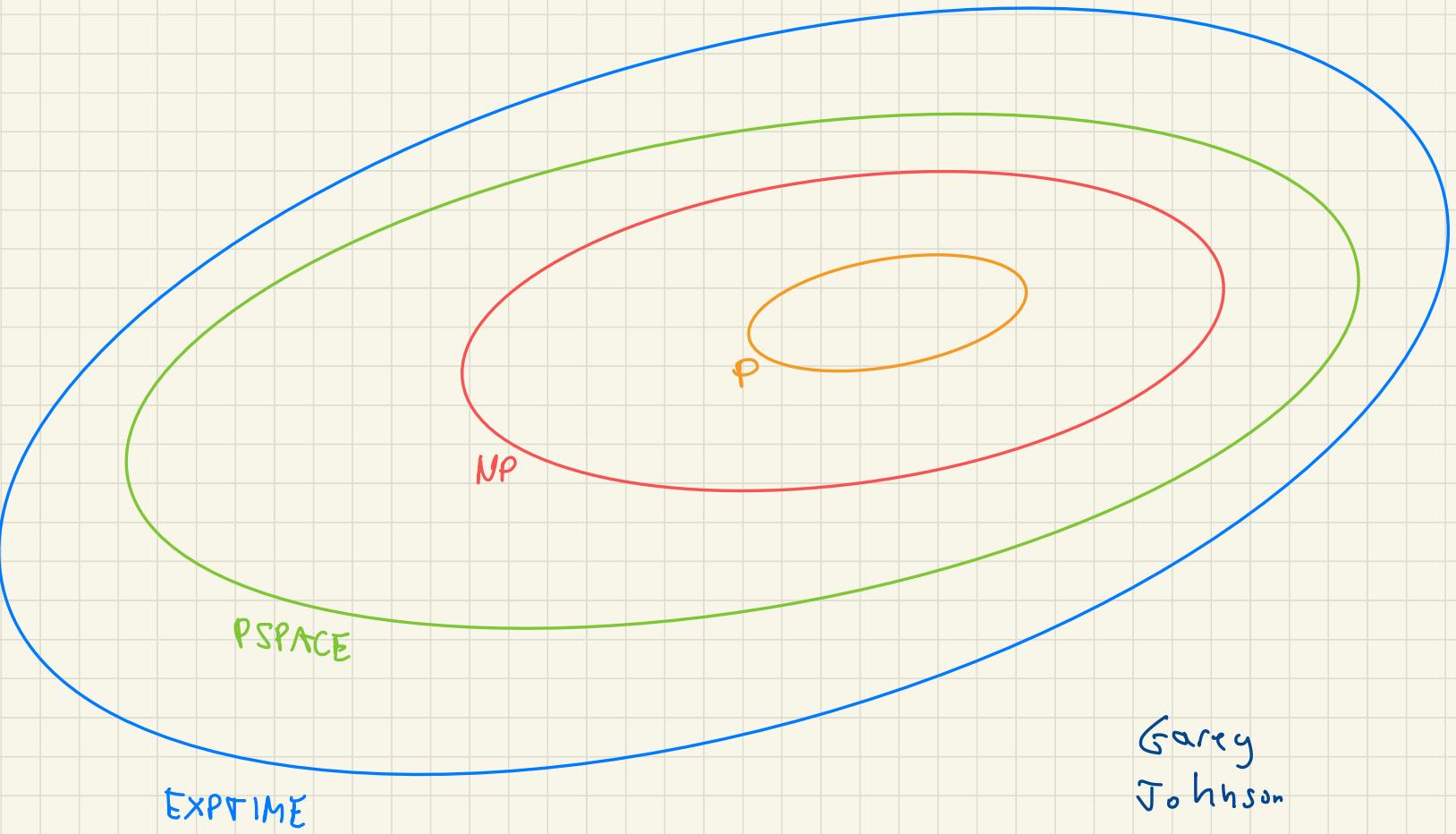
Quesione $\exists A' \subseteq A$ t.c. $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$?

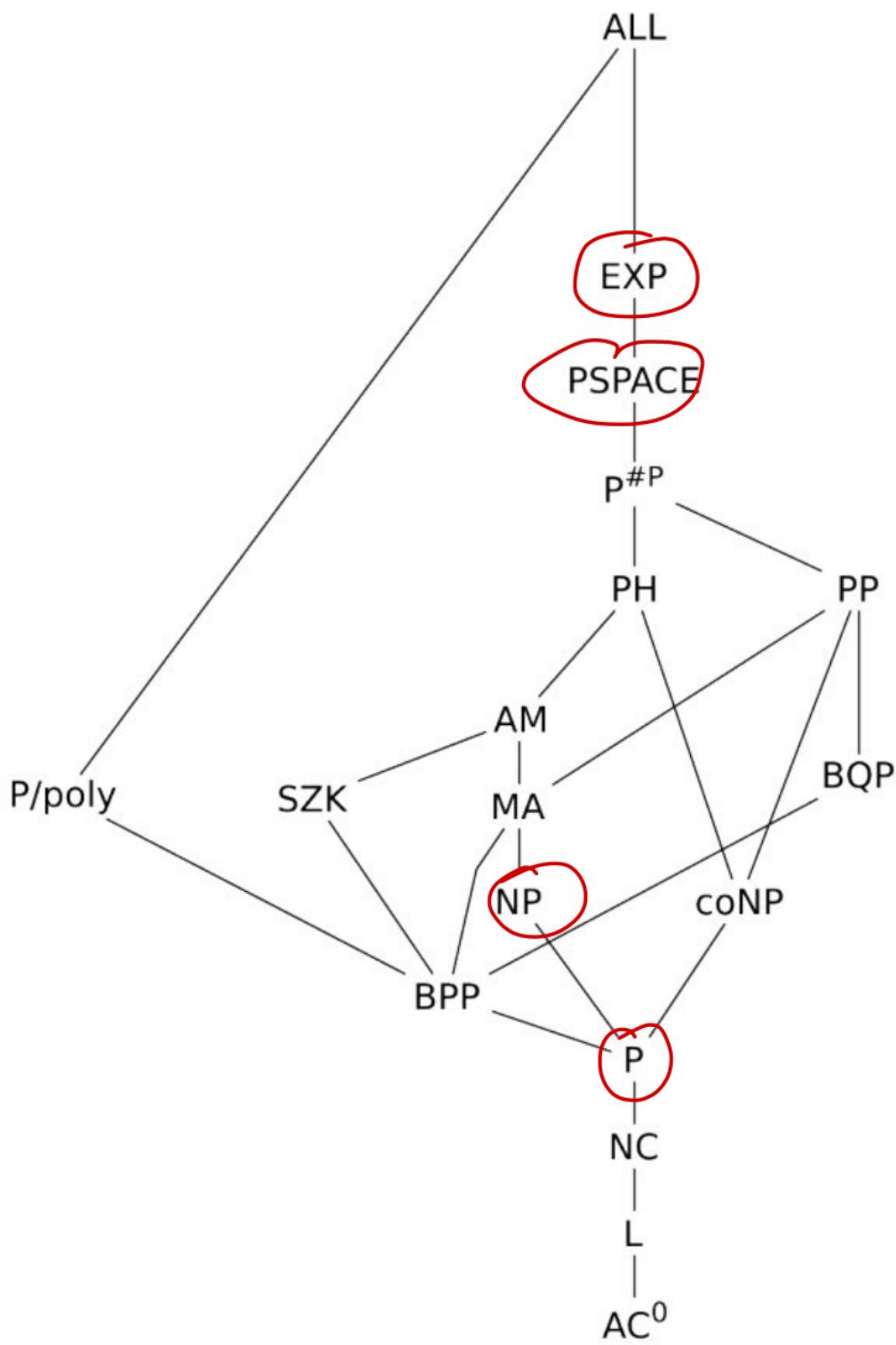
COMMESSO VIAGGIATORE

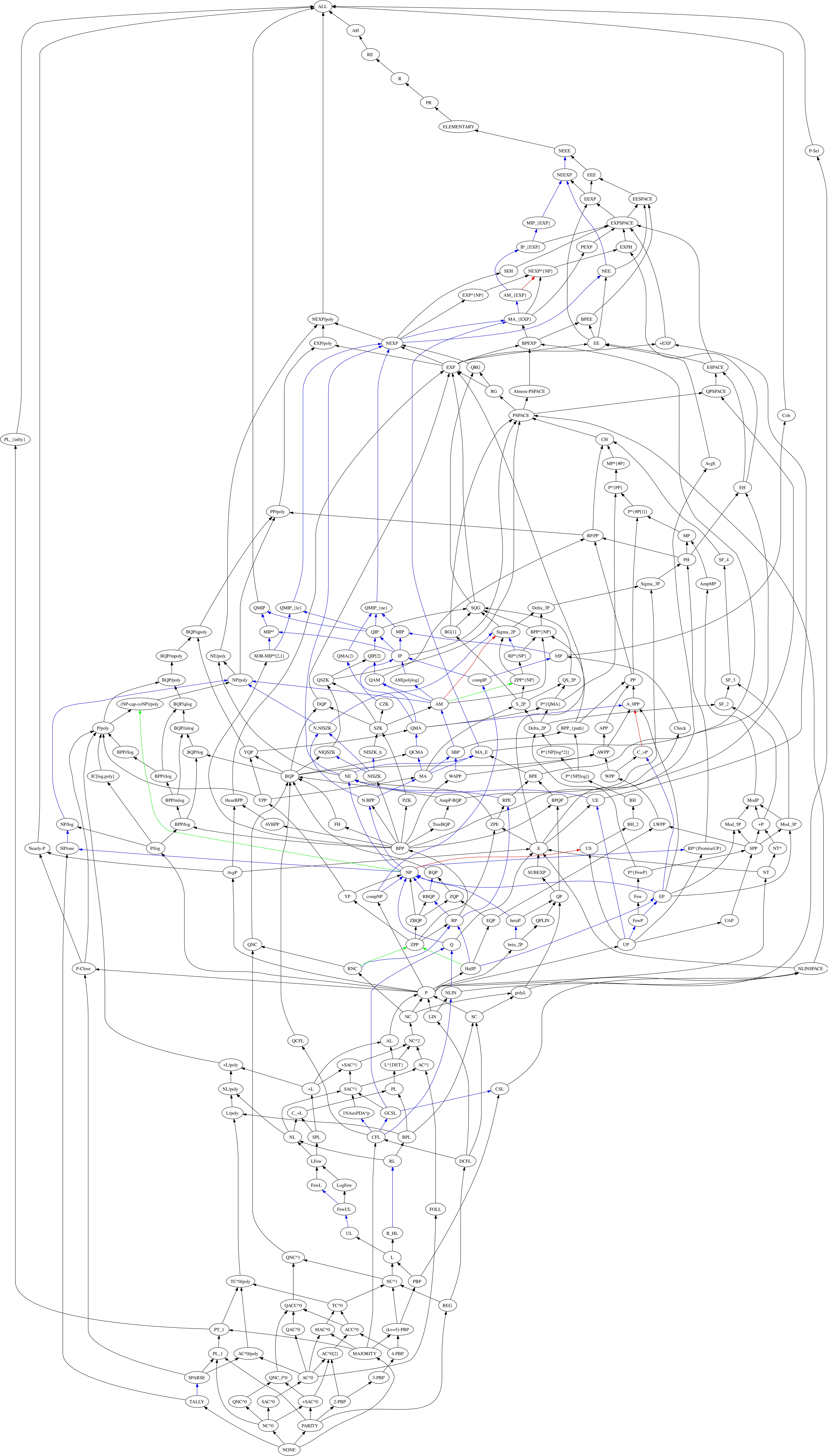
Istanza $C = \{c_1, c_2, \dots, c_m\}$ insieme finito di città
 $d(c_i, c_j) \in \mathbb{Z}^+$ distanze tra coppie di città $i, j = 1, \dots, m$

$B \in \mathbb{Z}^+$ numero

Quesione \exists "tour" che visita tutte le città di lunghezza
al più B ?







Esercizi

Cognome.....

Algoritmi e Strutture Dati

Nome

Prova scritta del 19 gennaio 2024

Matricola

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. Ricordatevi di scrivere cognome e nome su TUTTI i fogli (inclusi quelli di brutta).

1. Considerate la sequenza di numeri 28 42 17 15 36 25 32 20.

(a) Disegnate l'albero AVL che si ottiene inserendo i numeri della sequenza, uno dopo l'altro, nell'ordine indicato, a partire da un albero inizialmente vuoto, ribilanciando quando necessario, in modo da ottenere un albero AVL dopo ogni inserimento.	(b) Disegnate l'albero 2-3 che si ottiene inserendo i numeri della sequenza, uno dopo l'altro, nell'ordine indicato, a partire da un albero 2-3 inizialmente vuoto.
---	---

2. Considerate la seguente sequenza di numeri, memorizzata in un array: 28 42 17 15 36 25 32 20 44.

(a) Supponete di ordinare la sequenza <i>in modo crescente</i> mediante l'algoritmo quickSort, scegliendo come perno il primo elemento ed utilizzando, per effettuare la partizione, la strategia presentata a lezione. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di quickSort.
(b) Supponete di ordinare la sequenza <i>in modo decrescente</i> mediante l'algoritmo quickSort, scegliendo come perno il primo elemento ed utilizzando, per effettuare la partizione, la strategia presentata a lezione. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di quickSort.
(c) Supponete di ordinare la sequenza <i>in modo crescente</i> mediante l'algoritmo heapSort. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.
(d) Supponete di ordinare la sequenza <i>in modo decrescente</i> mediante l'algoritmo heapSort. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

3. In una città molto fredda dell'Europa settentrionale è stato deciso di costruire un sistema di gallerie sotterranee che durante la stagione invernale permettano di spostarsi a piedi senza mai uscire all'aperto. Le gallerie saranno collocate esattamente secondo la mappa stradale della città: sotto ogni strada vi sarà una galleria collegata direttamente agli edifici di quella strada. Per facilitare gli spostamenti, inoltre, alcune gallerie saranno percorribili utilizzando dei *tapis rulants*, come spesso avviene nei lunghi corridoi degli aeroporti.

Si deve decidere in quali gallerie collocare i tapis rulants in modo che:

- (i) sia sempre possibile spostarsi da un punto all'altro della città utilizzando i tapis rulants;
- (ii) la lunghezza complessiva delle gallerie in cui sono collocati i tapis rulants sia minima.

Più in dettaglio:

- una strada è delimitata da due incroci e non ha incroci all'interno (la fine di una strada a fondo chiuso è anch'essa un incrocio);
- si conosce la lunghezza di ogni strada;
- tutte le gallerie possono essere percorse a piedi in entrambe le direzioni;
- le gallerie in cui sono collocati i tapis rulants sono chiamate *gallerie principali*, le altre sono chiamate *gallerie secondarie*;
- in ogni galleria principale vi sono due tapis rulants, uno per ogni direzione;
- in base alla condizione (i) da ogni incrocio deve essere possibile raggiungere ogni altro incrocio utilizzando esclusivamente gallerie principali; in base alla condizione (ii) la lunghezza totale delle gallerie principali deve essere minima.



Risolvete i seguenti punti nell'ordine indicato:

(a) Spiegate come il problema possa essere descritto e formalizzato in termini di grafi.

(b) Progettate un algoritmo che determini:

- l'insieme delle strade sotto le quali devono essere collocate le gallerie principali;
- l'insieme delle strade sotto le quali devono essere collocate le gallerie secondarie.

Descrivete l'algoritmo a parole e poi, *ad alto livello*, in pseudocodice.

Ispiratevi a uno degli algoritmi presentati a lezione, indicate quale.

(c) Discutete una possibile rappresentazione del grafo utilizzato e una corrispondente implementazione dell'algoritmo, ricorrendo anche a eventuali strutture di supporto, fornendo una stima dei tempi di calcolo.
Giustificate dettagliatamente la stima dei tempi di calcolo.

(d) Supponete ora che sia fissato un incrocio c corrispondente alla piazza centrale della città e che, al posto della condizione (ii), sia richiesto di minimizzare, per ogni incrocio x , la lunghezza totale delle gallerie principali che da c permettono di raggiungere x . Indicate se la strategia che avete descritto al punto (b) risolve anche questo problema o se occorre utilizzare una strategia differente e perché. In tal caso indicate quale tra gli algoritmi visti a lezione può essere utile (non è richiesta la descrizione dell'algoritmo).

Tutte le risposte a questo esercizio devono essere giustificate adeguatamente.

(a) Spiegate come il problema possa essere descritto e formalizzato in termini di grafi.

Graph → vertici rappresentano gli alberi
↑ archi strade
pesi lunghezze
non ordinato

Problema si allarga ricoprendo minimo

gallorice principale \leftrightarrow albero ricoprendo minimo
" secondario \leftrightarrow la copertura

(b) Progettate un algoritmo che determini:

- l'insieme delle strade sotto le quali devono essere collocate le gallerie principali;
- l'insieme delle strade sotto le quali devono essere collocate le gallerie secondarie.

Descrivete l'algoritmo a parole e poi, *ad alto livello*, in pseudocodice.

Ispiratevi a uno degli algoritmi presentati a lezione, *indicate quale*.

Kruskal

ALGORITMO Gallerie (Grafo $G = (V, E, w)$) \rightarrow Insieme, Turin, non

ordine E in modo Y diverse si è P

$P \leftarrow \emptyset$ // gallerie principali \rightarrow mappa
 $S \leftarrow \emptyset$ // " " Secondarie per ogni vertice

FOR EACH $(x, y) \in E$ nell'ordine croc DO

| IF x e y non sono connessi i. (V, P) $\leftarrow T_x \leftarrow \text{Find}(x)$
| THEN $P \leftarrow P \cup \{(x, y)\}$ $T_y \leftarrow \text{Find}(y)$
| ELSE $S \leftarrow S \cup \{(x, y)\}$ $\leftarrow \text{UNION}(T_x, T_y)$

RETURN P, S

- (c) Discutete una possibile rappresentazione del grafo utilizzato e una corrispondente implementazione dell'algoritmo, ricorrendo anche a eventuali strutture di supporto, fornendo una stima dei tempi di calcolo.
Giustificate dettagliatamente la stima dei tempi di calcolo.

Rappresentazione grafo \leftrightarrow lista di archi
perché.. -

Struttura di supporto \leftrightarrow perfezione (Union/Find)
 \hookrightarrow per cui: for each

Medi R-24

Tempo di calcolo

ordinamento HeapSort $O(m \lg n)$

rappresentazione UnionFind: QuickFind Diflessione
Union costa $O(1)$
Find costa $O(\alpha_m n)$

- (d) Supponete ora che sia fissato un incrocio c corrispondente alla piazza centrale della città e che, al posto della condizione (ii), sia richiesto di minimizzare, per ogni incrocio x , la lunghezza totale delle gallerie principali che da c permettono di raggiungere x . Indicate se la strategia che avete descritto al punto (b) risolve anche questo problema o se occorre utilizzare una strategia differente e perché. In tal caso indicate quale tra gli algoritmi visti a lezione può essere utile (non è richiesta la descrizione dell'algoritmo).