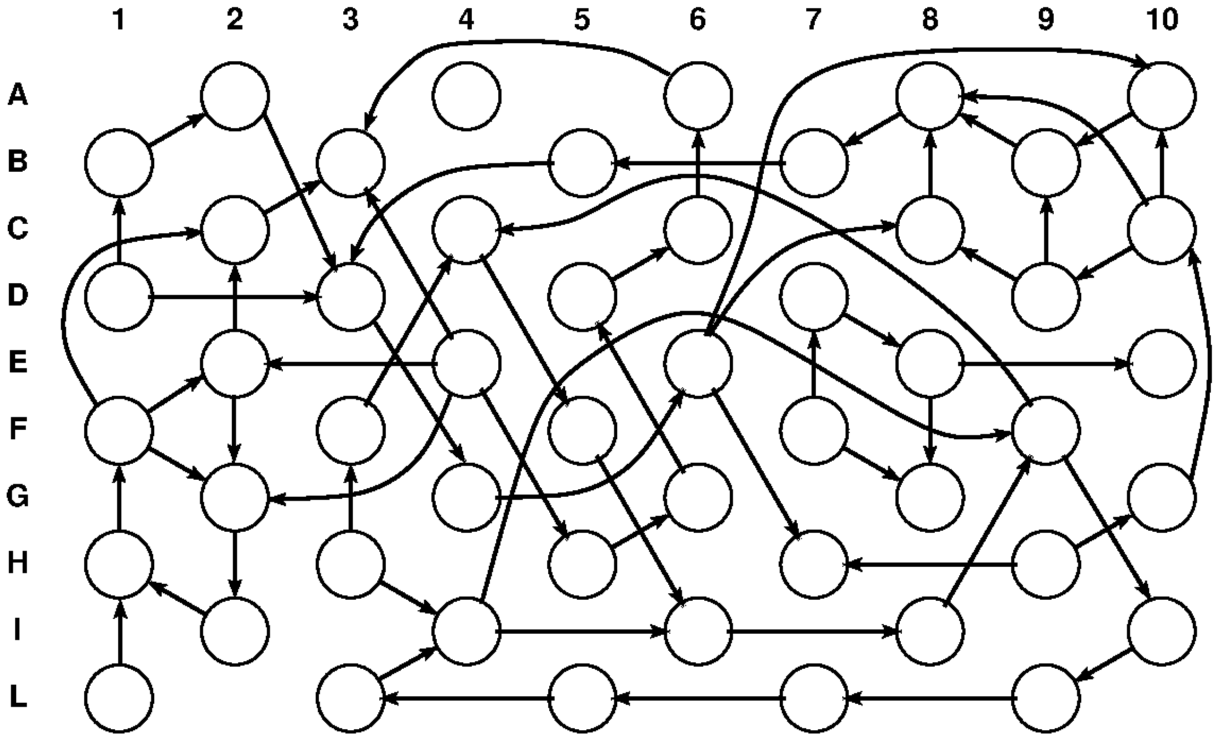
Esame di Ricerca Operativa - 25 settembre 2025

Facoltà di Scienze MM.FF.NN. - Verona

4 esercizi per 101 punti in palio voto $\geq \frac{5}{6}$(punti $-$ 6), 45$\longrightarrow$ 30 *e lode*)

- CORREZIONE -

Esercizio 1 (con 7 richieste: 1+1+3+1+1+1+5 = 13 punti [grafi visual]):



Tranne che per l'ultima richiesta, si ignorino le direzioni degli archi in figura. Eventuali nozioni mancanti possono essere pubblicamente richieste al docente.

Richieste dell'Esercizio 1

1 .1 (1 pt, componenti_connesse) Colora i nodi in modo da evidenziare le diverse componenti connesse

1 .2 (1 pt, distingui nodi e archi speciali)

nodi isolati	**foglie**	**cutnodes**	**bridges**

1 .3 (3 pt, make bipartite) rendi il grafo bipartito rimuovendo il minor numero di archi (1pt se suggerisci quali archi rimuovere ed evidenzi la bipartizione del grafo risultante, 1pt se esibisci una famiglia di cicli dispari che richiedano la rimozione di quel numero di archi per essere tutti eliminati, 1pt per il numero di soluzioni ottime). Addobba sempre la figura sopra per l'esibizione dei certificati

1 .4 (1 pt, planarità) Dire se l'intero grafo è planare oppure no, argomentandolo via certificati

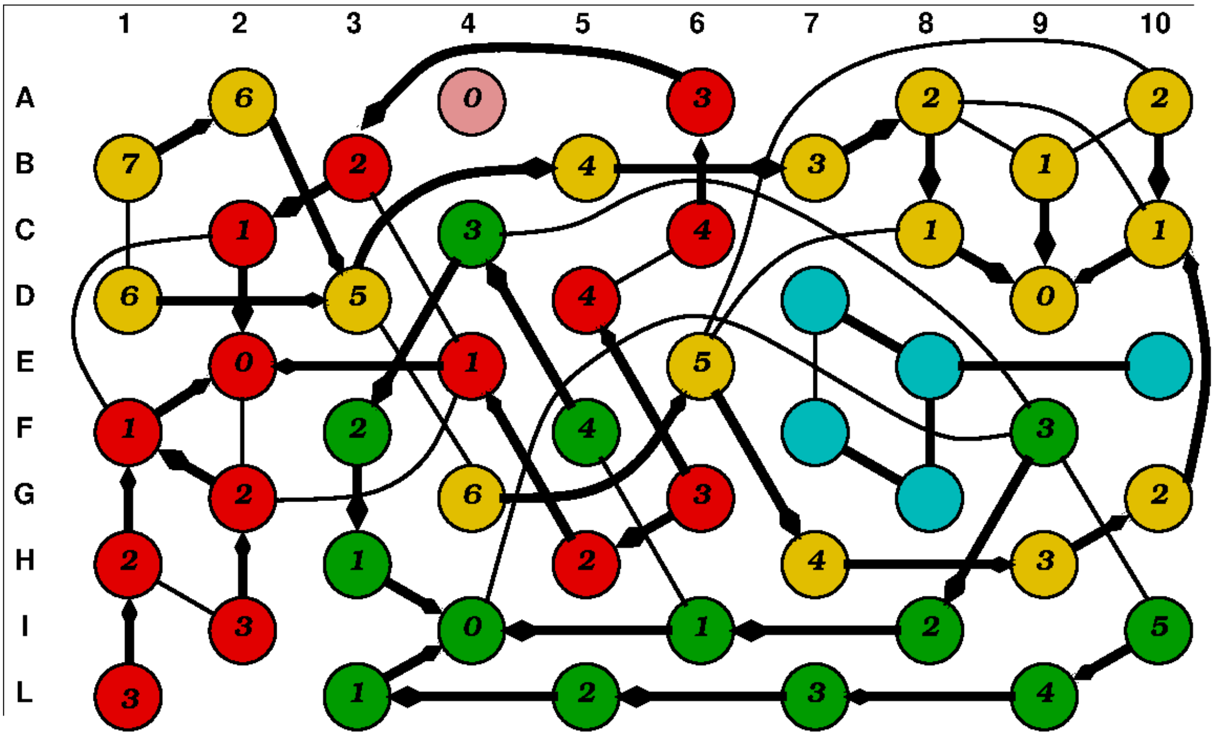
1 .5 (1 pt, Hamilton) Per ogni componente di più nodi, fornire un ciclo Hamiltoniano se presente, altrimenti un cammino Hamiltoniano se presente, altrimenti spiega perchè nisba

1 .6 (1 pt, Eulero) Per ogni componente di più nodi, fornire un ciclo Euleriano se presente, altrimenti un cammino Euleriano se presente, altrimenti spiega perchè nisba

1 .7 (5 pt, strong-connectivity) Si riguardino ora gli archi come diretti, ciascuno orientato come in figura. Per ciascuna componente tranne il nodo isolato si fornisca un ordinamento topologico oppure un ciclo diretto in essa contenuto (1pt). Per ogni componente non-aciclica si evidenzino: le componenti fortemente connesse (1pt), il DAG delle componenti fortemente connesse (1pt) e il relativo ordine topologico (1pt) e si certifichi la forte connessione di ciascuna componente offrendone costruzione per aggiunta di orecchie partendo da un ciclo diretto (1pt)

Svolgimento esercizio 1 .

Richiesta 1 (1 pt) (goal: componenti_connesse**).**
La colorazione fornita in figura evidenzia le componenti connesse del grafo. King Arthur può verificare che per ciascun arco i due estremi hanno lo stesso colore e concluderne che, se due nodi hanno colore diverso appartengono allora a componenti diverse.

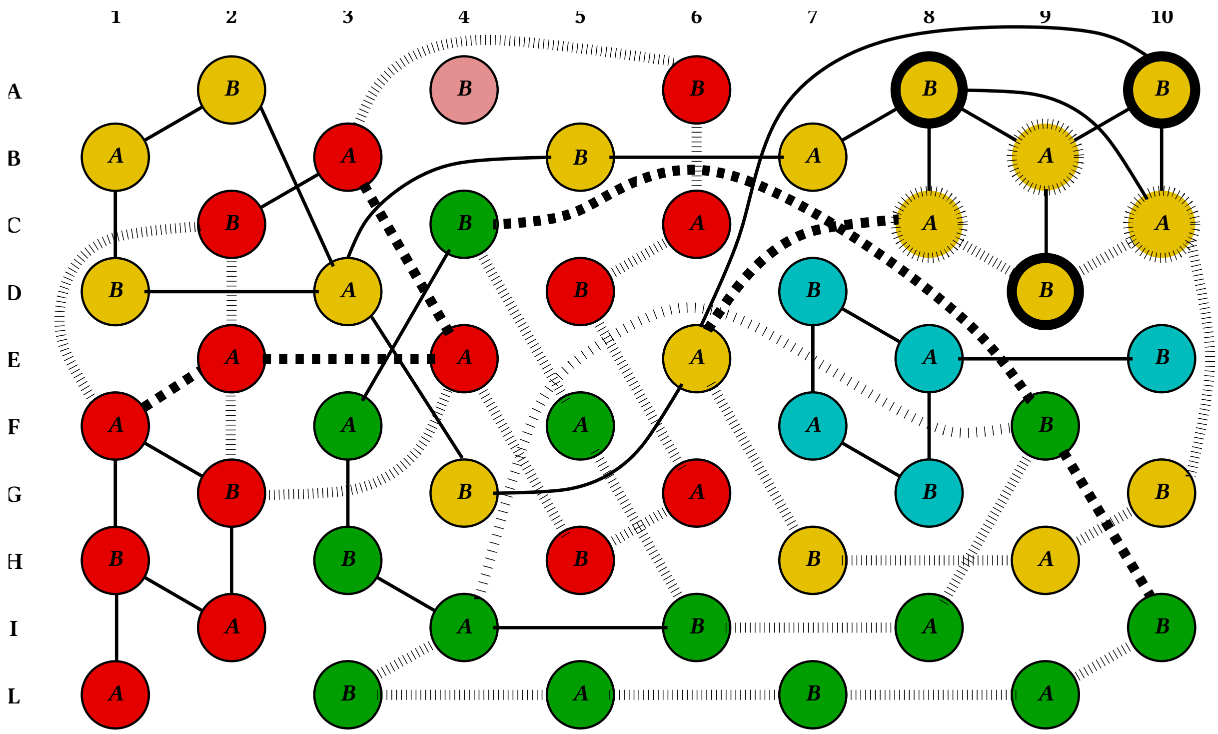


Per certificare l'implicazione inversa (nodi dello stesso colore appartengono ad una stessa componente) abbiamo inoltre fornito uno spanning tree per ogni componente. Per facilitare il lavoro di un King Arthur e consentirgli una verifica locale, negli alberi più complessi abbiamo eletto un rappresentante di classe da ciascuna componente e, fissatolo come radice, abbiamo orientato ogni arco da figlio a padre e riportata inoltre la distanza di ciascun nodo dalla rispettiva radice. Una equipe di King Arthurs che lavorano in parallelo, ciascuno situato su un diverso nodo, potrebbe condurre in $O(1)$ la seguente verifica: «se la distanza d riportata dal nodo è zero allora nessun arco esce da detto nodo, altrimenti ne esce precisamente un arco diretto verso un nodo dove è riportata la distanza $d - 1$.»

Richiesta 2 (1 pt) (goal: distingui nodi e archi speciali**).**

nodi isolati	foglie	cutnodes	bridges
A4	E10, L1	E8,H1,D3	E8-E10, H1-L1

Richiesta 3 (3 pt) (goal: make bipartite**).**
Il problema di produrre una 2-colorazione in cui il minor numero possibile di archi abbiano i due estremi dello stesso colore si decompone naturalmente sulle componenti connesse ed il numero delle soluzioni ottime per il grafo nel suo complesso sarà il prodotto dei numeri di soluzioni ottime per le singole componenti. Una 2-colorazione ottima è offerta in figura, dove ciascun nodo è etichettato A oppure B.



Le componenti rosa e blu sono già bipartite di loro, e la loro bipartizione è unica (a meno di scambiare A e B), ovvero l'unica soluzione ottima consiste nel non rimuovere alcun arco.

Nella componente verde dobbiamo rimuovere almeno due archi come dimostrato dai cicli dispari I4-F9-I10-L9-L7-L5-L3 e I4-I6-I8-F9-C4-F3-H3 evidenziati in figura: il lower-bound disegue poichè tali cicli sono edge-disjoint (ossia non hanno archi in comune) e da ciascuno di essi dobbiamo rimuovere almeno un arco; l'upper bound disegue dalla soluzione ammissibile che prevede di rimuovere i soli archi F9-I10 e F9-C4 rappresentata in figura. Per analizzare il numero di soluzioni ottime per la componente verde osserviamo come essa consti di quattro diversi cammini tesi tra i nodi I4 e F9. Due di questi cammini hanno lunghezze pari (4 e 6) e gli altri due hanno lunghezze dispari (1 e 3). Ogni soluzione ottima è ottenuta scegliendo due cammini della stessa parità e rimuovendo un qualsiasi arco da ciascuno di essi. Le soluzioni sono pertanto $4 \times 6 + 1 \times 3 = 27$.

La componente gialla non è bipartita (ciclo dispari evidenziato in figura) ma per renderla tale basta rimuovere l'arco E6-C8. Questa è l'unica soluzione ottima in quanto tra i nodi E6 e C8 sono tesi due cammini edge-disjoint di lunghezza pari.

La componente rossa richiede la rimozione di almeno tre archi come certificato da tre cicli edge-disjoint: il ciclo E4-B3-A6-C6-D5-G6-H5 e i due triangoli E2-C2-F1 e E2-G2-E4. Data la presenza dell'ulteriore triangolo E2-G2-F1, ogni soluzione ottima deve necessariamente rimuovere precisamente uno dei due archi E2-G2 ed E2-F1. La soluzione in figura rimuove E2-F1 e, del triangolo E2-G2-E4 rimuove l'arco E2-E4. A valle di questa seconda scelta deve necessariamente rimuovere l'arco E4-B3 per via del ciclo dispari B3-C2-F1-G2-E4 come terzo arco. Avesse rimosso G2-E4 come secondo arco avrebbe poi dovuto necessariamente rimuovere uno dei 6 archi del cammino B3-A6-C6-D5-G6-H5-E4 come terzo arco. Quindi 6+1=7 soluzioni ottime per la componente rossa rimuovono l'arco E2-F1. Le soluzioni ottime che rimuovono invece l'arco E2-G2 devono poi rimuovere l'arco E2-C2 oppure l'arco C2-F1. Nel primo caso è poi necessario rimuovere l'arco E4-B3, nel secondo caso resta necessariamente rimuovere uno dei 6 archi del cammino B3-A6-C6-D5-G6-H5-E4 come terzo arco. Quindi altre 7 soluzioni per un totale di 14 soluzioni per la componente rossa.

In conclusione, il numero totale di soluzioni ottime è $14 \times 1 \times 27 = 378$.

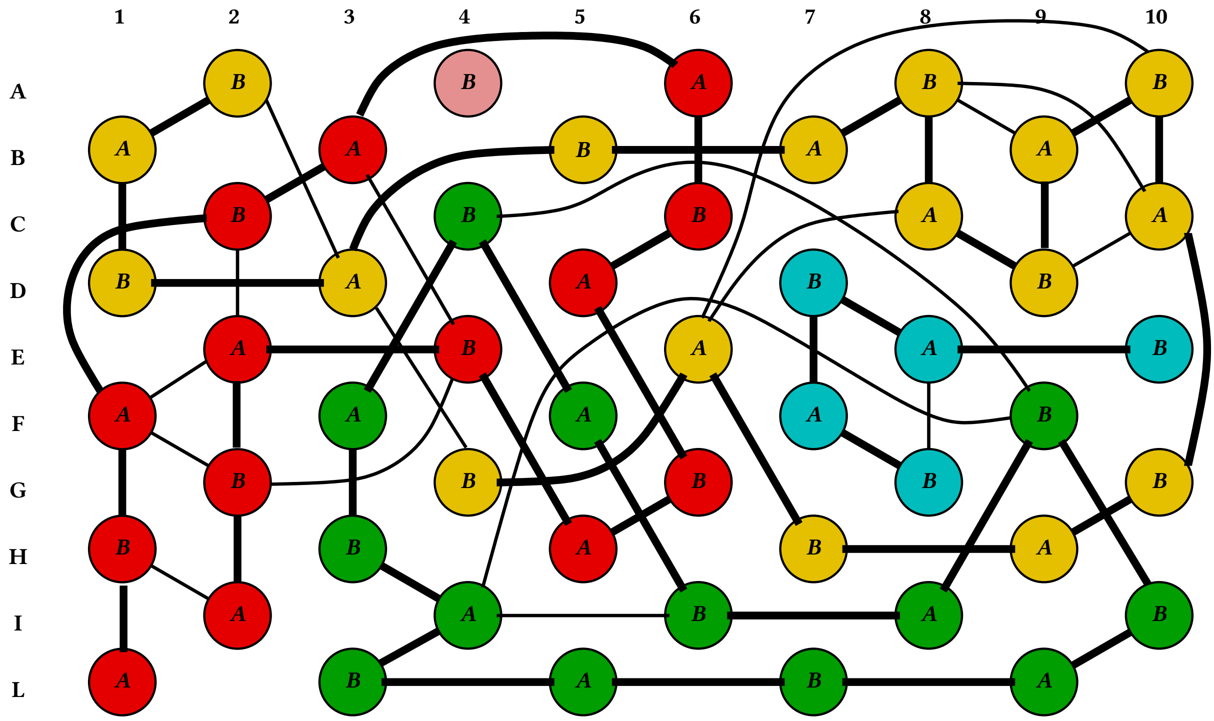
Numero di Soluzioni Ottime:

$\boxed{378}$

.

Richiesta 4 (1 pt) (goal: planarità**).**
per argomentare che il grafo non è planare basta argomentare che una qualsiasi delle sue componenti non lo è. Per argomentare che la componente gialla non lo è (invece è facile vedere che ogni altra lo è, solo della verde la figura fornita col testo non esibisce già un planar embedding ma basta ridisegnare l'arco I4-F9 immergendolo nella faccia esterna, ossia facendolo passare sopra tutti gli altri nodi) abbiamo esibito una $K_{3,3}$-subdivision nella figura dove i nodi sono stati colorati. Infatti nella stessa figura i sei nodi in alto a destra sono stati evidenziati da un bordo spesso che differenzia inoltre i tre nodi etichettati A (bordo spesso ma tratteggiato) e i tre nodi etichettati B (bordo spesso in linea continua). Questa distinzione corrisponde alla bipartizione del $K_{3,3}$ in 3 casette e 3 porcellini (acqua, luce e gas). Dobbiamo anche fornire, per ogni casetta, i tre percorsi verso acqua (L8), luce (L10) e gas (G9), avendo cura di non incrociare mai due percorsi su loro nodi interni. La casetta A8 è direttamente collegata (da arco singolo, nessun nodo intermedio) a ciascuna delle tre risorse. Lo stesso dicasi per la casetta D9. La casetta A10 è direttamente collegata ai porcellini C10 e B9 e non è difficile (in P, puoi usare la BFS) individuare un cammino da A10 a C8 nel grafo ottenuto dopo la rimozione degli altri 4 nodi importanti (A8,D9,C10,B9), ad esempio: A10-E6-C8.

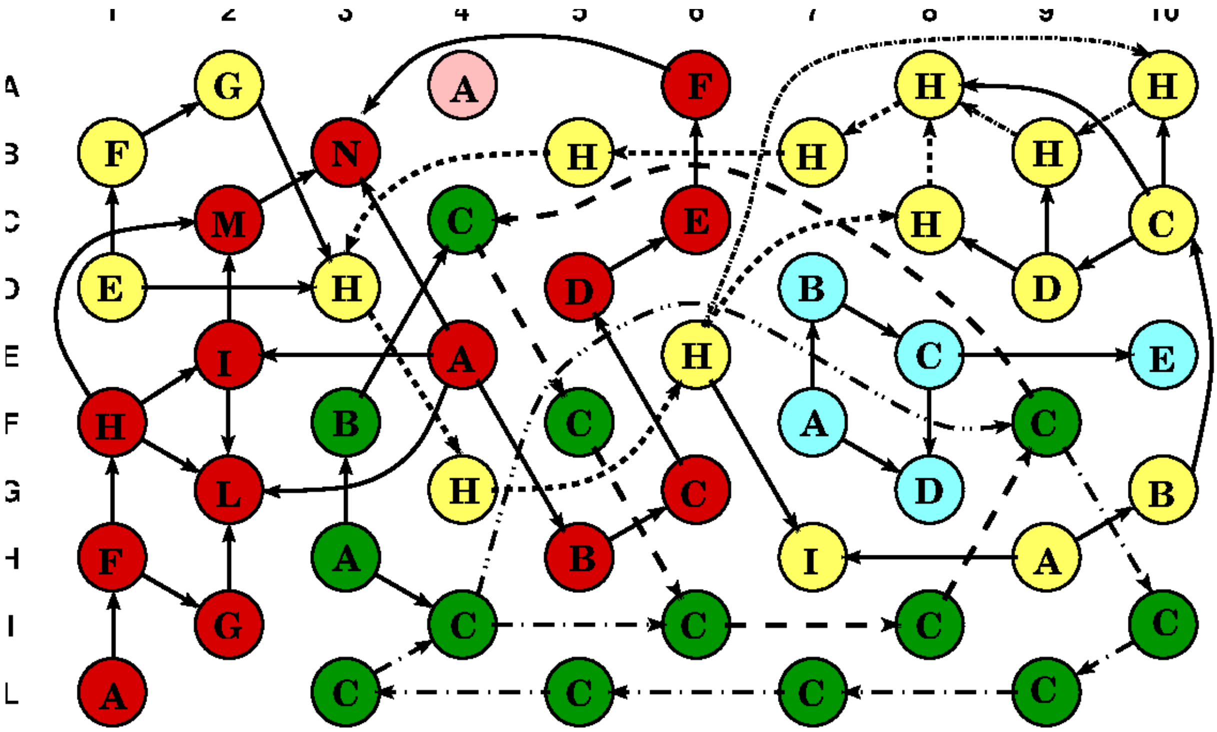
Richiesta 5 (1 pt) (goal: Hamilton**).**
la seguente figura esibisce un ciclo Hamiltoniano in ciascuna componente che ne possieda almeno uno (la sola componente verde, e quella rosa di un solo nodo), altrimenti esibisce un cammino Hamiltoniano chiudendo così la partita.



Infatti, la componente blu non può avere cicli Hamiltoniani dato che ha una foglia (E10). La componente gialla non può avere un ciclo Hamiltoniano dato che ha un cutnode (D3), così come la rossa (cutnode in H1).

Richiesta 6 (1 pt) (goal: Eulero)**.**
A10, D9 e D9 sono 3 nodi di grado dispari (3) della componente gialla, che pertanto non ha nè cicli nè cammini Euleriani. C4 e I6 sono gli unici due nodi di grado dispari della componente verde, che pertanto non ha cicli ma ha cammini Euleriani: percorri prima il cammino costituito dai 3 archi che non sono contenuti nel ciclo Hamiltoniano in figura, poi percorri tale ciclo. La componente rossa non può avere ciclo Euleriano in quanto ha almeno tre nodi di grado dispari (L1,H1,B3). La componente blu non può avere cicli Euleriani dato che ha una foglia (E10), ma un cammino Euleriano è ottenuto percorrendo prima l'arco E8-G8 e poi il cammino Hamiltoniano esibito in figura.

Richiesta 7 (5 pt) (goal: strong-connectivity)**.**



Le componenti azzurra, rossa e rosa sono dei DAG come evidenziato dall'ordinamento topologico $A, B, C, \ldots$ in cui i nodi possono essere eseguiti rispettando le precedenze dettate dagli archi.

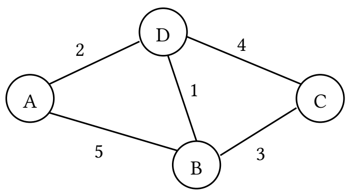
Per la componente gialla e verde la figura evidenzia la partizione dei nodi nelle componenti fortemente connesse (due nodi sono labellati con la stessa lettera quando appartengano alla stessa componente, ossia quando ciascuno possa essere raggiundo dall'altro). Il valore della lettera certifica che qualora i nodi di una stessa componete fossero collassati in un unico nodo ne risulterebbe un DAG, ossia che le classi della partizione fornita non vadano ulteriormente fuse tra loro. Cha la partizione corretta non sia invece più fine, ossia che ogni due nodi di una stessa classe fornita siano effettivamente mutualmente raggiungibili, è certificato dalla ear-decomposition che abbiamo dato per ogni classe non banale (ossia con almeno due nodi): si costruisca il sottografo indotto dai nodi della classe partendo dal ciclo in linea tratteggiata, poi si aggiungano le orecchie (cammini che partano e terminino in nodi già raccolti nella classe in formazione, ossia già dimostrati mutualmente raggiungibili a questo punto della costruzione; i nodi interni di questi cammini vengono ad aggiungersi alla classe in formazione)

Esercizio 2 (con 14 richieste: 1+1+1+1+1+1+6+2+1+2+1+6+5+10 = 39 punti [modellazione/ riduzioni]):

Il modello di PL qui a destra (in alto) computa un albero ricoprente di peso minimo (MST, minimum spanning tree) in un generico grafo $G = (V, E)$ con costi non-neativi sugli archi $c : E \to \mathbb{R}_+$. Il modello coinvolge un vettore x di variabili reali, indicizzato dal set E. Per ogni $F \subseteq E$, definiamo $x(F) := \sum_{e \in F} x_e$. Dove $\mathbb{S}$ è l'insieme dei sottoinsiemi di V non-banali (ossia nè vuoti nè uguali a V), per ogni $S \in \mathbb{S}$ si indica con $\delta(S)$ l'insieme degli archi con un estremo in S e l'altro in $V \setminus S$ (ossia gli archi del taglio di spiagge S e $V \setminus S$).

$$
\begin{aligned}
\min\ & x(E) \\
& x(\delta(S)) \geq 1 \text{ for all } S \in \mathbb{S} \\
& x \geq 0
\end{aligned}
$$

your instance:



Richieste dell'Esercizio 2

2 .1 (1 pt, one MST) dare un MST per il grafo in figura

2 .2 (1 pt, unicity) Certifica l'unicità dell'MST avvalendoti del lemma dei tagli. Quali tagli hai dovuto osservare/esibire?)

2 .3 (1 pt, explicit instance-specific) Scrivi esplicitamente il modello di PL che computa l'MST per l'istanza assegnata in figura. (Istanzia cioè il modello generale fornito per l'istanza specifica.)

2 .4 (1 pt, canonic form) In che forma particolare è il problema di PL che hai appena scritto? Perchè? Mettilo in forma canonica definendo le variabili di slack.

2 .5 (1 pt, auxiliary problem) Poichè il problema non è ad origine ammissibile, si consideri il problema ausiliario.

2 .6 (1 pt, first tableau) Scrivi il primo tableau del problema ausiliario.

2 .7 (6 pt, first phase) Risolvi all'ottimo il problema ausiliario col metodo del simplesso (1 punto per ogni pivot e 1 punto per ogni prova del nove esplicitamente spesa e chiaramente rappresentata sul foglio).

2 .8 (2 pt, change horse) Ottieni un tableau del problema originale che esprima una soluzione di base ammissibile. Un punto è per il recupero della funzione obiettivo originaria.

2 .9 (1 pt, second phase) Ottieni il tableau che esprime la soluzione ottima del problema originale.

2 .10 (2 pt, dual) Recupera dall'ultimo tableau le soluzioni primale e duale ottime (1pt) ed interpretale (1pt) entrambe nel contesto del problema di MST che intendevi risovere sull'istanza asseganata.

2 .11 (1 pt, dual opts) La soluzione duale è unica? Si/no e perchè.

2 .12 (6 pt, complementary slackness) Si indichi ora col paramentro K il costo dell'arco AD. La soluzione ottima ottenuta sopra rimarrà comunque ammissibile. Poniti la sfida di stabilire se sia ottima o meno (dipenderà dal valore di K) tramite le condizioni degli scarti compementari. Riscrivi il problema primale e duale introducendo il parametro K (1pt). Imposta le condizioni degli scarti complementari relative alla nostra soluzione primale (2pt). Ottieni soluzione duale complementare ad essa (1pt). Come puoi ora stabilire il range di valori di K per cui le due soluzioni sono ottime (2pt)?

2 .13 (5 pt, Steiner tree problem) il problema MST è un caso particolare dello Steiner tree problem dove oltre al grafo $G = (V, E)$ con costi sugli archi viene dato in input un sottoinsieme T di V e all'albero da ritornare non si richiede più di raggiungere tutti i nodi in V ma basta che raggiunga tutti i nodi di T (può raggiungere anche nodi in $V \setminus T$ ma solo nella misura in cui ciò aiuti nel contenere il costo, o quantomeno non lo aumenti). Fornire formulazione di Programmazione Lineare Intera (PLI) dello Steiner tree problem (2pt) prendendo ispirazione dalla formulazione di PL per l'MST fornita con l'esercizio (consentiamo quindi che il numero di vincoli sia esponenziale nelle dimensioni dell'istanza). Dare poi però un esempio dove la soluzione ottima del problema di PL rilassato (ottenuto ignorando i vincoli di interezza) è frazionaria (3pt).

2 .14 (10 pt, NPC-proof) Dimostrare l'NP-hardness dello Steiner tree problem con una riduzione da Min Node Cover, il problema di rimuovere il minimo numero di nodi dal grafo per lasciarlo privo di archi (5pt). Altri 5pt sono in palio per una riduzione da 3-SAT.

Svolgimento esercizio 2 .

Richiesta 1 (1 pt) (goal: one MST**).**

L'istanza assegnata in figura ammette un'unico MST. Esso include i 3 archi: AD, BC e BD.

Richiesta 2 (1 pt) (goal: unicity**).**

L'arco AD è incluso in ogni MST in quanto arco di peso strettamente minimo della stella di A. L'arco BC è incluso in ogni MST in quanto arco di peso strettamente minimo della stella di C. L'arco BD è incluso in ogni MST in quanto arco di peso strettamente minimo del taglio $\delta(S)$, con $S = \{A, B\}$.

Richiesta 3 (1 pt) (goal: explicit instance-specific**).**

$$
\begin{array}{llllll}
\min 5x_{A,B} + 2x_{A,D} + 3x_{B,C} + x_{B,D} + 4x_{C,D} & & & & & \\
x_{A,B} + x_{A,D} & & & & \geq 1 & \text{(for S=\{A\})} \\
x_{A,B} + x_{B,C} + x_{B,D} & & & & \geq 1 & \text{(for S=\{B\})} \\
x_{B,C} + x_{C,D} & & & & \geq 1 & \text{(for S=\{C\})} \\
x_{A,D} + x_{B,D} + x_{C,D} & & & & \geq 1 & \text{(for S=\{D\})} \\
x_{A,D} + x_{B,C} + x_{B,D} & & & & \geq 1 & \text{(for S=\{A,B\})} \\
x_{A,B} + x_{A,D} + x_{B,C} + x_{C,D} & & & & \geq 1 & \text{(for S=\{A,C\})} \\
x_{A,B} + x_{B,D} + x_{C,D} & & & & \geq 1 & \text{(for S=\{A,D\})} \\
x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D} \geq 0 & & & & &
\end{array}
$$

Poichè $\delta(S) = \delta(V \setminus S)$ per ogni insieme di nodi $S \subset V$, i vincoli prescritti dal modello di PL proposto sono solo quelli sopra. In realtà, ha senso poi far cadere il vincolo generato per $S = \{A, C\}$ dacchè esso è dominato dal vincolo per $S = \{A\}$ in quanto le variabili sono tutte non-negative.

Richiesta 4 (1 pt) (goal: canonic form**).**

Il problema è in forma di minimizzazione standard. Pertanto, abbiamo due opzioni per la corma canonica.

Opzione 1: prima di portarlo in forma canonica, lo portiamo in forma di massimizzazione standard:

$$
\begin{array}{lll}
-\max -5x_{A,B} - 2x_{A,D} - 3x_{B,C} - x_{B,D} - 4x_{C,D} & & \\
-x_{A,B} - x_{A,D} & \leq -1 & \text{(for S=\{A\})} \\
-x_{A,B} - x_{B,C} - x_{B,D} & \leq -1 & \text{(for S=\{B\})} \\
-x_{B,C} - x_{C,D} & \leq -1 & \text{(for S=\{C\})} \\
-x_{A,D} - x_{B,D} - x_{C,D} & \leq -1 & \text{(for S=\{D\})} \\
-x_{A,D} - x_{B,C} - x_{B,D} & \leq -1 & \text{(for S=\{A,B\})} \\
-x_{A,B} - x_{B,D} - x_{C,D} & \leq -1 & \text{(for S=\{A,D\})} \\
x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D} \geq 0 & &
\end{array}
$$

e da qui produciamo la forma canonica introducendo, per ogni vincolo, una **variabile *di slack*** non-negativa:

$$
\begin{array}{llr}
-\max -5x_{A,B} - 2x_{A,D} - 3x_{B,C} - x_{B,D} - 4x_{C,D} & & \\
s_A = x_{A,B} + x_{A,D} - 1 & & \text{(for S=\{A\})} \\
s_B = x_{A,B} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{B\})} \\
s_C = +x_{B,C} + x_{C,D} - 1 & & \text{(for S=\{C\})} \\
s_D = +x_{A,D} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{D\})} \\
s_{AB} = +x_{A,D} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{A,B\})} \\
s_{AD} = x_{A,B} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{A,D\})} \\
x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D}, s_A, s_B, s_C, s_{AB}, s_{AD} \geq 0 & &
\end{array}
$$

Opzione 2: produrre direttamente la forma canonica introducendo, per ogni vincolo, una **variabile *di surplus*** non-negativa:

$$
\begin{array}{llr}
\min 5x_{A,B} + 2x_{A,D} + 3x_{B,C} + x_{B,D} + 4x_{C,D} & & \\
s_A = x_{A,B} + x_{A,D} - 1 & & \text{(for S=\{A\})} \\
s_B = x_{A,B} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{B\})} \\
s_C = x_{B,C} + x_{C,D} - 1 & & \text{(for S=\{C\})} \\
s_D = x_{A,D} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{D\})} \\
s_{AB} = x_{A,D} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{A,B\})} \\
s_{AD} = x_{A,B} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{A,D\})} \\
x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D}, s_A, s_B, s_C, s_{AB}, s_{AD} \geq 0 & &
\end{array}
$$

Richiesta 5 (1 pt) (goal: auxiliary problem**).**
Si noti che tutti i termini noti sono negativi. Fosse anche uno solo, dobbiamo rivolgerci ad un metodo del simplesso a due fasi. Il testo chiede di avvalersi del problema ausiliario. Per produrre la forma standard del problema ausiliario, introduciamo la variabile non-negativa «di colla aggiustatutto» x_0.

$$
\begin{array}{llr}
\min x_0 & & \\
s_A = x_0 + x_{A,B} + x_{A,D} - 1 & & \text{(for S=\{A\})} \\
s_B = x_0 + x_{A,B} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{B\})} \\
s_C = x_0 + x_{B,C} + x_{C,D} - 1 & & \text{(for S=\{C\})} \\
s_D = x_0 + x_{A,D} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{D\})} \\
s_{AB} = x_0 + x_{A,D} + x_{B,C} + x_{B,D} - 1 & & \text{(for S=\{A,B\})} \\
s_{AD} = x_0 + x_{A,B} + x_{B,D} + x_{C,D} - 1 & & \text{(for S=\{A,D\})} \\
x_0, x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D}, s_A, s_B, s_C, s_{AB}, s_{AD} \geq 0 & &
\end{array}
$$

Richiesta 6 (1 pt) (goal: first tableau**).**

Per ottenere il primo tableau del problema ausiliario ne riscriviamo la forma canonica in formato tabellare.

Spesa questa scelta, la forma canonica e primo dizionario sono:

Dizionario Prob Aux

$$
\begin{cases}
z = 0 + 1x_0 + 0x_{A,B} + 0x_{A,D} + 0x_{B,C} + 0x_{B,D} + 0x_{C,D} \\
s_A = -1 + 1x_0 + 1x_{A,B} + 1x_{A,D} + 0x_{B,C} + 0x_{B,D} + 0x_{C,D} \\
s_B = -1 + 1x_0 + 1x_{A,B} + 0x_{A,D} + 1x_{B,C} + 1x_{B,D} + 0x_{C,D} \\
s_C = -1 + 1x_0 + 0x_{A,B} + 0x_{A,D} + 1x_{B,C} + 0x_{B,D} + 1x_{C,D} \\
s_D = -1 + 1x_0 + 0x_{A,B} + 1x_{A,D} + 0x_{B,C} + 1x_{B,D} + 1x_{C,D} \\
s_{AB} = -1 + 1x_0 + 0x_{A,B} + 1x_{A,D} + 1x_{B,C} + 1x_{B,D} + 0x_{C,D} \\
s_{AD} = -1 + 1x_0 + 1x_{A,B} + 0x_{A,D} + 0x_{B,C} + 1x_{B,D} + 1x_{C,D}
\end{cases}
$$

$\min z :$

$x_0, x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D} \geq 0$

$s_A, s_B, s_C, s_D, s_{AB}, s_{AD} \geq 0$

Tableau Prob Aux

	–	x_0	$x_{A,B}$	$x_{A,D}$	$x_{B,C}$	$x_{B,D}$	$x_{C,D}$
z	0	1	0	0	0	0	0
s_A	-1	(1)	1	1	0	0	0
s_B	-1	1	1	0	1	1	0
s_C	-1	1	0	0	1	0	1
s_D	-1	1	0	1	0	1	1
s_{AB}	-1	1	0	1	1	1	0
s_{AD}	-1	1	1	0	0	1	1

Richiesta 7 (6 pt) (goal: first phase**).**

Come si vede, pivotando la variabile x_0 con una qualsiasi delle variabili in base otterremo un primo dizionario ammissibile. Scegliamo, di pivotare con la variabile x_A, ottenendo:

Dizionario Ammissibile Aux

$$
\begin{cases}
z = 1 - 1s_A - 1x_{A,B} - 1x_{A,D} + 0x_{B,C} + 0x_{B,D} + 0x_{C,D} \\
x_0 = 1 + 1s_A - 1x_{A,B} - 1x_{A,D} + 0x_{B,C} + 0x_{B,D} + 0x_{C,D} \\
s_B = 0 + 1s_A + 0x_{A,B} - 1x_{A,D} + 1x_{B,C} + 1x_{B,D} + 0x_{C,D} \\
s_C = 0 + 1s_A - 1x_{A,B} - 1x_{A,D} + 1x_{B,C} + 0x_{B,D} + 1x_{C,D} \\
s_D = 0 + 1s_A - 1x_{A,B} + 0x_{A,D} + 0x_{B,C} + 1x_{B,D} + 1x_{C,D} \\
s_{AB} = 0 + 1s_A - 1x_{A,B} + 0x_{A,D} + 1x_{B,C} + 1x_{B,D} + 0x_{C,D} \\
s_{AD} = 0 + 1s_A + 0x_{A,B} - 1x_{A,D} + 0x_{B,C} + 1x_{B,D} + 1x_{C,D}
\end{cases}
$$

$\min z :$

$s_A, x_{A,B}, x_{A,D}, x_{B,C}, x_{B,D}, x_{C,D} \geq 0$

$x_0, s_B, s_C, s_D, s_{AB}, s_{AD} \geq 0$

Dizionario Ammissibile Aux

	–	s_A	$x_{A,B}$	$x_{A,D}$	$x_{B,C}$	$x_{B,D}$	$x_{C,D}$
z	1	-1	-1	-1	0	0	0
x_0	1	1	-1	-1	0	0	0
s_B	0	1	0	-1	1	1	0
s_C	0	1	(-1)	-1	1	0	1
s_D	0	1	-1	0	0	1	1
s_{AB}	0	1	-1	0	1	1	0
s_{AD}	0	1	0	-1	0	1	1

Dal secondo pivot in poi la driving force è quella dell'ottimalità, come colonna di pivot scegliamo quindi una variabile a costo ridotto negativo (come la $x_{A,B}$) e per la riga di pivot facciamo attenzione a mantenere l'ammissibilità. Come variabile uscente scegliamo quindi la s_C che attualmente è a zero (e verrebbe condotta a diminuire all'aumentare della $x_{A,B}$).

Ma da qui, poichè un solo studente è proseguito oltre, interrompo la correzione e lascio l'esercizio come cantiere aperto. Credo sia sufficientemente impostato da giungere fino all'ottenimento di un MST come soluzione ottima del problema di PL proposto dall'esercizio.

Esercizio 3 (con 12 richieste: 1+1+1+1+1+1+1+1+2+1+1+2 = 14 punti [programmazione dinamica]):

La seguente tabella offre, nella sua terza riga, una sequenza S di numeri naturali (la prima riga, a caratteri in neretto, serve solo ad indicizzarla).

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
31	10	19	50	60	17	28	39	45	26	36	59	21	42	30	55	57	20	22	14	37	56	24	47	57	40	23

Richieste dell'Esercizio 3

3 .1 (1 pt, DP: last_in_pos) Alla tabella si aggiunga una riga che in ogni posizione i, con $1 \leq i \leq n$, riporti la massima lunghezza di una sottosequenza strettamente crescente di S che di S prenda l'elemento in posizione i come suo ultimo elemento.

3 .2 (1 pt, DP: first_in_pos) Si aggiunga una riga che in ogni posizione i, con $1 \leq i \leq n$, riporti la massima lunghezza di una sottosequenza strettamente crescente di S che di S prenda l'elemento in posizione i come suo primo elemento.

3 .3 (1 pt, opt_sol: libera) Trovare una sottosequenza strettamente crescente di S di massima lunghezza. Specificare quanto è lunga e fornirla.

3 .4 (1 pt, certify opt) Fornire un minimo numero di sottosequenze mai crescenti tali che ogni elemento della sequenza originale in input ricada in almeno una di esse. Specificare quante sono e fornirle.

3 .5 (1 pt, opt_sol: last) Trovare una sottosequenza strettamente crescente di S di massima lunghezza tra quelle che terminano con l'elemento in posizione 19. Specificare quanto è lunga e fornirla.

3 .6 (1 pt, opt_sol: left) Trovare una sottosequenza strettamente crescente di S di massima lunghezza tra quelle che di S non prendono alcun elemento di indice inferiore a 8. Specificare quanto è lunga e fornirla.

3 .7 (1 pt, opt_sol: prende) Trovare una sottosequenza strettamente crescente di S di massima lunghezza tra quelle che di S prendono l'elemento in posizione 13. Specificare quanto è lunga e fornirla.

3 .8 (1 pt, N-sequenza) Una sequenza è detta una N-sequenza, o sequenza strettamente crescente con al più un ripensamento, se esiste un indice i tale che ciascuno degli elementi della sequenza, esclusi al più il primo e l'i-esimo, è strettamente maggiore dell'elemento che lo precede. Trovare la più lunga N-sequenza che sia una sottosequenza della sequenza data. Specificare quanto è lunga e fornirla.

3 .9 (2 pt, quante opt sol: libere) Le sottosequenze di S sono 2^n, in corrispondenza biunivoca coi sottoinsiemi degli indici degli elementi di S che includono. Stabilire quante siano le sottosequenze strettamente crescenti di S di massima lunghezza.

3 .10 (1 pt, opt_sol: last) Quante sono le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che prendono l'elemento in posizione 19 come loro ultimo elemento?

3 .11 (1 pt, opt_sol: first) Quante sono le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che prendono l'elemento in posizione 8 come loro primo elemento?

3 .12 (2 pt, quante opt sol: con) Quante sono le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che di S includono l'elemento in posizione 13?

Svolgimento esercizio 3 .

Richiesta 1

Per rispondere alle prime richieste conviene compilare preventivamente un paio di tabelle di programmazione dinamica, di fatto quelle oggetto delle prime due richieste.

Ecco la prima tabella esplicitamente richiesta (`max_len_last_at`, compilata da sinistra):

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
1	1	2	3	4	2	3	4	5	3	4	6	3	5	4	6	7	3	4	2	5	7	5	6	8	6	5
31	10	19	50	60	17	28	39	45	26	36	59	21	42	30	55	57	20	22	14	37	56	24	47	57	40	23

Richiesta 2

Ed ecco la seconda (`max_len_first_at`), compilata da destra (la nuova riga subito sotto a quella coi valori della sequenza in input):

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**

1	1	2	3	4	2	3	4	5	3	4	6	3	5	4	6	7	3	4	2	5	7	5	6	8	6	5
31	10	19	50	60	17	28	39	45	26	36	59	21	42	30	55	57	20	22	14	37	56	24	47	57	40	23
6	8	7	4	1	7	6	5	4	6	5	1	5	4	4	3	1	5	4	4	3	2	3	2	1	1	1

Il fatto che il massimo valore sia 8 sia nella riga prodotta per evadere la Richiesta 1 che nella riga ora inserita per evadere la Richiesta 2 viene a verifica parziale della correttezza dei conteggi.

Richiesta 3

La massima lunghezza di una sottosequenza strettamente crescente di S è pertanto appunto **8**. Volendo ricostruire una sottosequenza strettamente crescente di tale lunghezza si procede a ritroso rispetto all'ordine in cui i numeri nella riga (possiamo lavorare su quella prodotto nella Richiesta 1, oppure su quella prodotto nella Richiesta 2, è in differente) sono stati calcolati, ed utilizzarli come angioletti compagnetti (oracoli che ci anticipano le conseguenze di ogni nostra scelta, se includere o meno l'elemento nella posizione corrente del nostro esodo). Affidandoci a tali grilli parlanti diviene facile affrontare a colpo sicuro ogni scelta guidati dalla forza della promessa.

Le soluzioni ottime di un problema combinatorico come questo possono essere anche in numero esponenziale, ma eccone una:

pos	**2**	**3**	**7**	**8**	**9**	**16**	**22**	**25**
$S[i]$	10	19	28	39	45	55	56	57

E' facile verificare l'ammissibilità di tale soluzione, ma come verificarne l'ottimalità?

Richiesta 4

Poichè una sottosequenza strettamente crescente ed una sottosequenza mai crescente non possono condividere più di un singolo elemente della sequenza di riferimento, allora un set di k sottosequenze mai crescenti che coprano ogni elemente della sequenza di riferimento dimostrano che nessuna sottosequenza strettamente crescente può contenere più di k elementi (contenendone al più uno per ogni sottosequenza mai crescente).

Ecco quindi il nostro certificato di ottimalità della soluzione fornita al punto precedente, nella forma di $k = 8$ sottosequenze mai crescenti che coprano ogni elemente della sequenza S di riferimento per questo esercizio:

pos	**1**	**2**
$S[i]$	31	10

pos	**3**	**6**	**20**
$S[i]$	19	17	14

pos	**4**	**7**	**10**	**13**	**18**
$S[i]$	50	28	26	21	20

pos	**5**	**8**	**11**	**15**	**19**
$S[i]$	60	39	36	30	22

pos	**9**	**14**	**21**	**23**	**27**
$S[i]$	45	42	37	24	23

pos	**12**	**16**	**24**	**26**
$S[i]$	59	55	47	40

pos	**17**	**22**
$S[i]$	57	56

pos	**25**
$S[i]$	57

Richiesta 5

La massima lunghezza di una sottosequenza strettamente crescente di S tra quelle che terminano con l'elemento in posizione 19 è il **4** che si trova nella colonna 19 della riga di programmazione dinamica `last_in_pos` introdotta con la Richiesta 1. La tecnica/mantra per ricostruire una soluzione ottima è quella di consultare ad ogni scelta la tabella di programmazione dinamica, come spiegato sopra. La parola di Javè non è univoca ma potente in ricchezza generativa: le soluzioni ottime di un problema combinatorico come questo possono essere anche in numero esponenziale. Per soddisfare la richiesta ci basta esibirne una:

pos	**2**	**6**	**18**	**19**
$S[i]$	10	17	20	22

Richiesta 6

La massima lunghezza di una sottosequenza strettamente crescente di S tra quelle che di S non prendono alcun elemento di indice inferiore a 8 è **6** come si può evincere della riga di programmazione dinamica `first_in_pos` introdotta con la Richiesta 2. (Volendo rendere questa riga più leggibile, si potrebbe aggiungere come ulteriore riga di programmazione dinamica il calcolo del massimo valore che lei offre a destra di ciascuna posizione, poi consultare tale riga nella colonna 8.) La tecnica/mantra per ricostruire una soluzione ottima è sempre quella (anche per altri problemi di ottimizzazione risolti tramite programmazione dinamica). Per soddisfare la richiesta ci basta esibire una soluzione ottima:

pos	**10**	**11**	**14**	**16**	**22**	**25**
$S[i]$	26	36	42	55	56	57

Richiesta 7

Combinando il dato in colonna 13 dalle due tabelle compilate per le Richieste 1 e 2 scopriamo che la massima lunghezza di una sottosequenza strettamente crescente di S che prenda l'elemento in posizione 13 è $\mathbf{3+5-1=7}$. Ricostruendo al rispettivo ritroso entrambe (verso destra oppure verso sinistra) le ali di una soluzione ottima si ottiene:

pos	**2**	**6**	**13**	**14**	**16**	**22**	**25**
$S[i]$	10	17	21	42	55	56	57

Richiesta 8

Per trovare un punto di ripensamento favorevole per la N-sottosequenza di S ho confrontato i valori nelle tabelle compilate per rispondere alle Richieste 1 e 2. Per semplificare tale confronto (da quadratico a lineare) ho preferito introdurre due ulteriori righe di PD (una, compilata da sinistra, riporta la massima lunghezza di una sottosequenza strettamente crescente per ogni prefisso; l'altra, compilata da destra, riporta la massima lunghezza di una sottosequenza strettamente crescente per ogni suffisso; sono ovvie le ricorrenze con cui calcolarle avvalendosi dei valori già prodotti per rispondere alle Richieste 1 e 2).

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
1	1	2	3	4	4	4	4	5	5	5	6	6	6	6	6	7	7	7	7	7	7	7	7	8	8	8
1	1	2	3	4	2	3	4	5	3	4	6	3	5	4	6	7	3	4	2	5	7	5	6	8	6	5
31	10	19	50	60	17	28	39	45	26	36	59	21	42	30	55	57	20	22	14	37	56	24	47	57	40	23
6	8	7	4	1	7	6	5	4	6	5	1	5	4	4	3	1	5	4	4	3	2	3	2	1	1	1
8	8	7	7	7	7	6	6	6	6	5	5	5	5	5	5	5	5	4	4	3	3	3	2	1	1	1

In fondo ogni N-sottosequenza può essere spezzata in due sottosequenze strettamente crescenti, una sinistra ed una destra. La tabella quì sopra può essere compilata in tempo lineare e consente di stabilire in tempo lineare che possiamo far terminare in posizione 17 la sottosequenza sinistra e far partire in posizione 18 la sottosequenza destra.

Procedendo a ritroso sulle due ali come da competenza già esibita otteniamo la seguente N-sequenza di massima lunghezza:

pos	**2**	**6**	**10**	**11**	**14**	**16**	**17**	**18**	**19**	**21**	**22**	**25**
$S[i]$	10	17	26	36	42	55	57	20	22	37	56	57

Essa è lunga **12**.

Richiesta 9

Dalla seguente tabella evinco che sono **8** le sottosequenze strettamente crescenti di S di massima lunghezza. La riga centrale della tabella riporta la sequenza in input, mentre le due righe collocate sopra (sotto) di essa sono compilate da sinistra (da destra). Le righe limitrofe alla sequenza in input (`max_len_last_at` e `max_len_first_at`) le abbiamo gia incontrate in precedenti tabelle ma vengono quì riprese in quanto ancillari al computo delle due ulteriori righe che ci servono ora (`num_opts_last_at` e `num_opts_first_at`). L'idea è che in ogni posizione i, con $1 \leq i \leq n$, la riga `num_opts_last_at` (`num_opts_first_at`) riporti il numero delle più lunghe sottosequenze strettamente crescenti di S che di S prendano l'elemento in posizione i come loro ultimo (primo) elemento.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
1	1	1	1	1	1	2	2	2	2	4	2	2	6	6	8	8	2	4	1	14	8	4	26	8	18	4
1	1	2	3	4	2	3	4	5	3	4	6	3	5	4	6	7	3	4	2	5	7	5	6	8	6	5
31	10	19	50	60	17	28	39	45	26	36	59	21	42	30	55	57	20	22	14	37	56	24	47	57	40	23
6	8	7	4	1	7	6	5	4	6	5	1	5	4	4	3	1	5	4	4	3	2	3	2	1	1	1
3	8	4	1	1	4	3	2	1	1	1	1	7	1	3	1	1	3	3	3	2	1	1	1	1	1	1

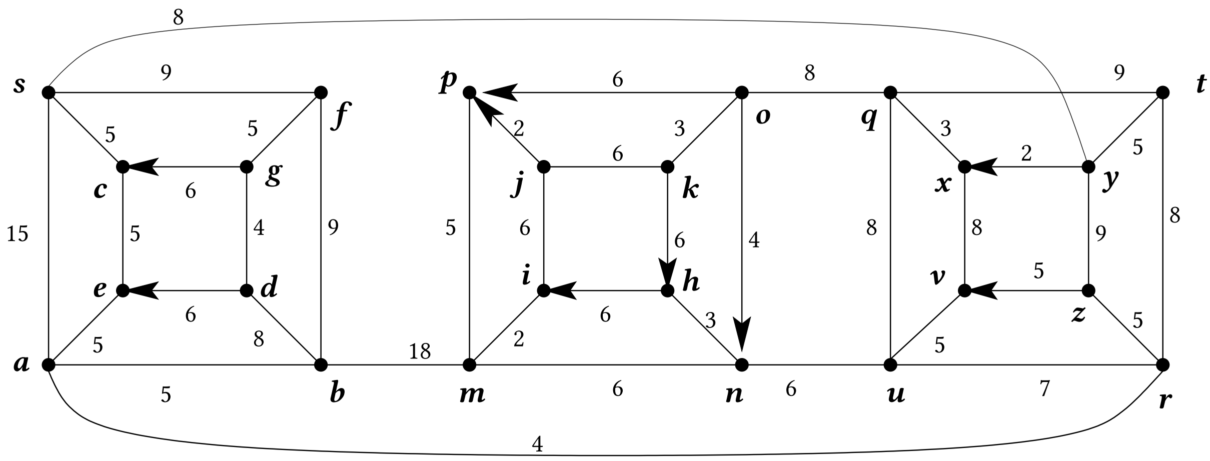
Anche quì ho preferito compilare sia delle righe di PD (programmazione dinamica) da sinistra (che rispondono a domande per ogni prefisso di S) che delle righe di PD da destra (che rispondono alle domande speculari per ogni suffisso di S) in modo da trovare verifica almeno parziale nel confronto dei valori ottenuti per i due (quello sinistro e quello destro) sotto-problemi più grossi, ultimi a cadere come birilli sotto i colpi della ovvia ricorrenza che definisce la rispettiva famiglia. Per altro questo doppio conteggio mi serve se poi voglio rispondere in economia a tutte le richieste che seguono.

Richiesta 10
Sono $\mathbf{4}$ le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che prendono l'elemento in posizione 19 come loro ultimo elemento.

Richiesta 11
Sono $\mathbf{2}$ le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che prendono l'elemento in posizione 8 come loro primo elemento.

Richiesta 12
Sono $\mathbf{14 = 2 \times 7}$ le sottosequenze strettamente crescenti di S di massima lunghezza tra quelle che di S includono l'elemento in posizione 13.

Esercizio 4 (con 11 richieste: 8+2+3+4+2+2+1+2+5+4+2 = 35 punti [grafi]):



Richieste dell'Esercizio 4

4 .1 (8 pt, recognize planarity) Dire, certificandolo, se siano planari o meno il grafo G, il grafo G_u ottenuto da G sostituendo l'arco sy con un arco su, e il grafo G_q ottenuto da G sostituendo l'arco sy con un arco sq (3pt). Per quelli di questi tre grafi che non siano planari, identifica gli archi la cui rimozione renderebbe il grafo planare (1pt) corredando ciascuno arco col relativo planar ebbeding (2pt) e fornendo certificati di non planarità in quantità sufficiente da coprire ogni altro arco (2pt)

4 .2 (2 pt, recognize 2-colorability) Dire, certificandolo, quale sia il minimo numero di archi da rimuovere per rendere bipartiti i grafi G, G_u e G_q (1pt se corretti tutti i certificati di bicolorazione, 1 pt se ok ogni certificato di ottimalità).

4 .3 (3 pt, shortest paths) Rispettando i sensi unici, si riporti la distanza di ciascun nodo dal nodo s (1pt). Dare un albero dei cammini minimi (1pt) e si descriva lo spazio di tali alberi precisando quanti sono (1pt).

4 .4 (4 pt, max-flow/min-cut) In G, trovare un massimo s,t-flusso (2pt) e un minimo s,t-taglio (2pt).

4 .5 (2 pt, flow sensitivity) Specifica l'incremento nel massimo valore di flusso ottenibile rimuovendo il vincolo di capacità del singolo arco e, per ogi arco e per il quale detto incremento è non-nullo (1pt).

4 .6 (2 pt, certify flow sensitivity) Per uno qualsiasi degli archi identificati al punto precedente esibisci prova che aumentandone la capacità si possa ottenere il valore di flusso dichiarato(1pt). Certifica anche che l'aumento non è superiore a quanto dichiarato (1pt).

4 .7 (1 pt, one MST) In G, fornire un albero ricoprente di peso minimo.

4 .8 (2 pt, MST categorize edges) Etichetta ciascun arco con la lettera A se appartiene a ogni MST, B se a nessuno, C altrimenti. (Se li hai ti conviene usare 3 colori.)

4 .9 (5 pt, MST certificates) Per ciascuno dei quattro archi incidenti nel nodo m certificare l'etichetta assegnatagli al punto precedente.

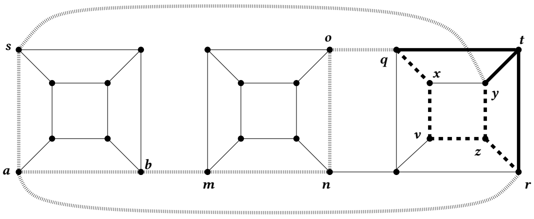
4 .10 (4 pt, count MSTs) Quanti sono gli MST in G?

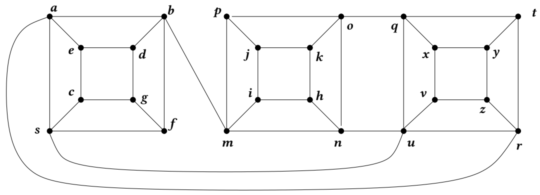
4 .11 (2 pt, max match) Fornire un matching di massima cardinalità in $G_{p,n}$, il grafo ottenuto da G rimuovendo i nodi p ed n (1pt). Certifica la non esistenza di un matching con un numero maggiore di archi? (1pt)

Svolgimento esercizio 4 .

Richiesta 1

La non-planarità di G è certificata dalla $K_{3,3}$ subdivision in figura, sulla sinistra. Per rendere G planare basta rimuovere uno qualsiasi di questi 4 archi: qt, sy, tr o bm. E' facile modificare il $K_{3,3}$ fornito in figura in modo da fargli evitare ogni altro arco (preso singolarmente), ma ovviamente tu per acquisire anche questi punti dovevi fornire un paio di altri $K_{3,3}$ nel grafo G. Sulla destra, si offre un planar embedding di G_u, che ne certifica invece la planarità.

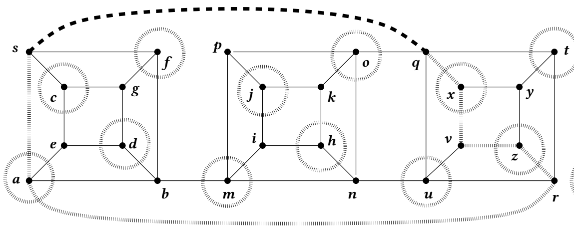


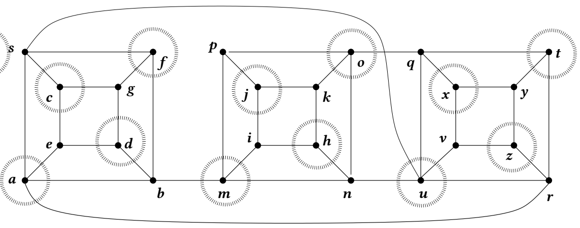


Un planar embedding di G_q è invece fornito nella figura a seguire, sulla sinistra.

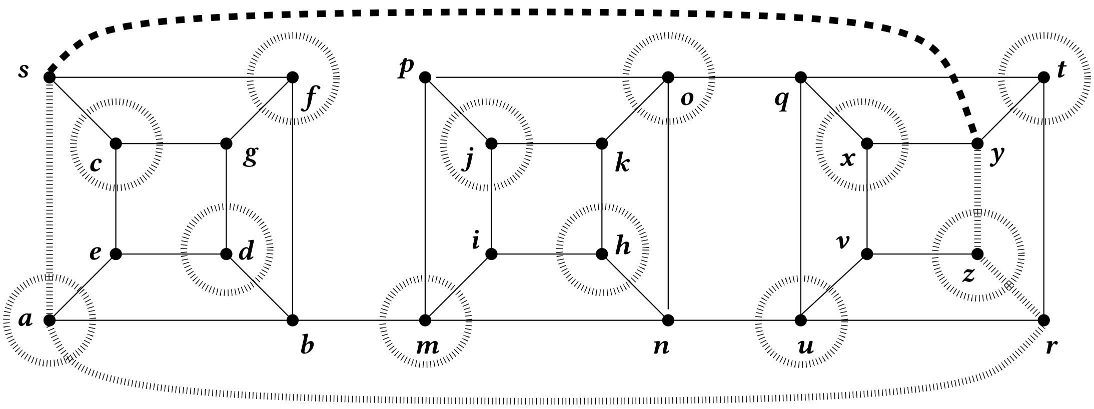
Richiesta 2

Nella figura a seguire, sulla sinistra, si mostra anche come il grafo G_q possa essere reso bipartito rimuovendo un solo arco (oltre alla bipartizione è evidenziato un ciclo dispari che certifica che almeno un arco deve essere rimosso). Sulla destra possiamo verificare che G_u era invece bipartito già di suo.



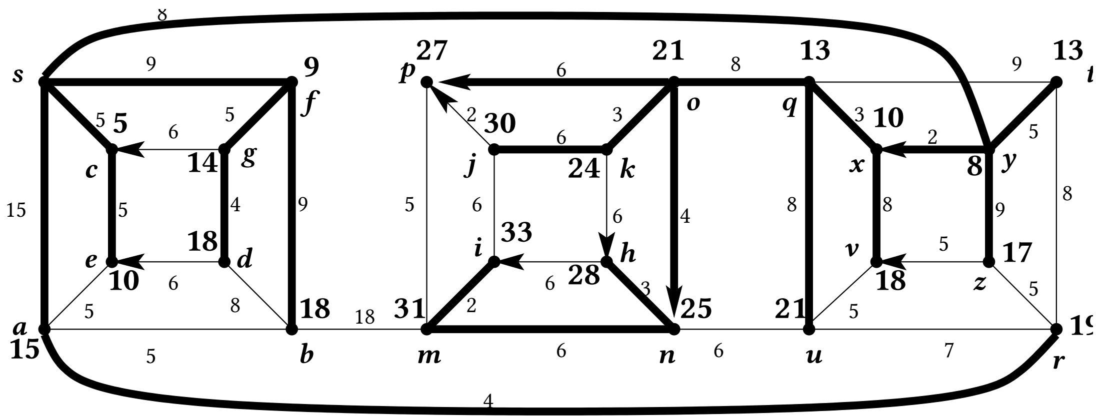


Su G serve rimuovere almeno un arco, e un solo arco di nuovo basta, ogni certificato richiesto è evidenziato in figura.



Richiesta 3

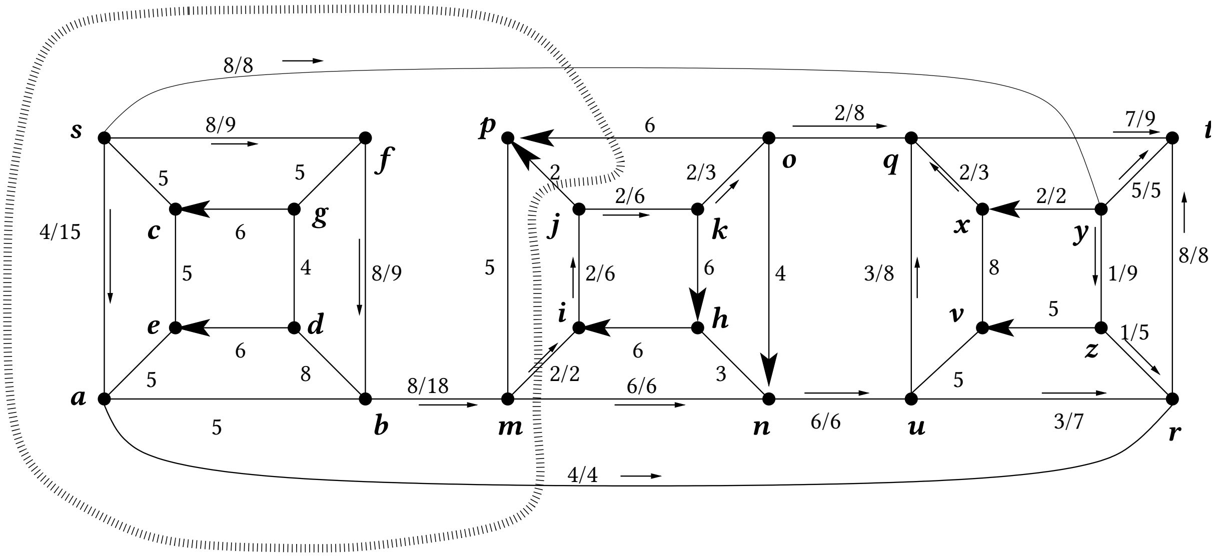
Ecco le distanze e un possibile albero dei cammini minimi dal nodo s (essi si certificano vicendevolmente.)



I possibili alberi dei cammini minimi sono due in quanto l’unico nodo che può scegliere il padre (tra due possibilità: il nodo s oppure il nodo e) è il nodo a.

Richiesta 4

Un flusso ottimo (valore 20) è visualizzato in figura, con l’un ico s, t-taglio minimo e dello stesso valore. L’unico altro s, t-taglio minimo si ottiene da quello riportato in figura facendo migrare il nodo i dal lato di s. Questo taglio da solo certifica l’ottimalità del flusso in figura, come di ogni altro possibile flusso di valore 20 (ce ne sono diversi, la formulazione di PL del max-flow tende ad essere altamente degenere).



Richieste 5 e 6

Tutti gli archi per i quali un aumento della capacità dell’arco potrebbe tradursi in un aumento del valore del flusso massimo dovranno appartenere a questo s, t-taglio (può essere vista come una manifestazione degli scarti complementari). Questa affermazione può anche essere spinta oltre, rendendola caratterizzante: gli archi un cui aumento di capacità comporterebbe un aumento nel massimo valore di flusso sono precisamente gli archi che appartengolo ad ogni s, t-taglio minimo. In questo caso abbiamo un solo s, t-taglio minimo e quindi gli archi in questione sono precisamente quelli dell’s, t-taglio minimo, ossia sono questi quattro: ar, sy, mi e mn. Per ogni unità di aumento della capacità su uno di questi archi ci si può attendere un’unità di aumento del flusso massimo, ma solo fino ad un valore soglia oltre il quale subentra un nuovo collo di bottiglia. Per problemi di max-flow min-cut con capacità degli archi tutte intere questi valori di soglia sono sempre tutti interi.

Nel caso dell'arco ar possiamo sfruttare il cammino $saruqt$ che nella rete ausiliaria a capacità minima 2 in corrispondenza dell'arco qt per produrre un nuovo flusso di valore $20 + 2 = 21$. Non ha senso aumentare la capacità dell'arco ar di oltre 2 unità perchè a quel punto viene a saturarsi l'arco qt, ultimo degli archi con testa in t, e la stella di t è un s, t-taglio (collo di bottiglia) di valore 22, di più non passa se non cominciamo ad aumentare anche la capacità di uno di questi 3 archi entranti in t.

Nel caso dell'arco mn possiamo sfruttare il cammino $sabmnuqt$ che nella rete ausiliaria a capacità residua minima 2 in corrispondenza dell'arco q, t per produrre un nuovo flusso di valore $20 + 2 = 22$. Di nuovo la stella entrante in t è s, t-taglio che dimostra che il valore dell's, t-flusso si ferma comunque a 22.

Se potessimo invece violare il vincolo di capacità sul solo arco sy, alzaremmo comunque il valore del flusso a 22 reinstradando lungo il cammino aumentante $syzruqt$, e di nuovo il prossimo arco a saturarsi sarebbe necessariamente l'arco qt e la stella entrante in t offrirebbe certificato di ottimalità del flusso così ottenuto.

Infine, nel caso dell'arco mi possiamo sfruttare il cammino $sabmijkoqt$ che nella rete ausiliaria a capacità residua minima 1 in corrispondenza dell'arco k, o per produrre un nuovo flusso di valore $20 + 1 = 21$. Per dimosrare l'ottimalità di questa soluzione dobbiamo esibire un s, t-taglio di valore 21. Esso si ottiene modificando l's, t-taglio in figura facendo migrare i nodi i, j, k, h e n dal lato di t al lato di s.

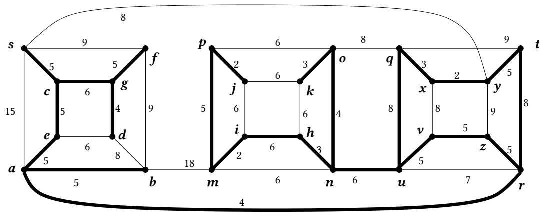
Richieste 7 e 8

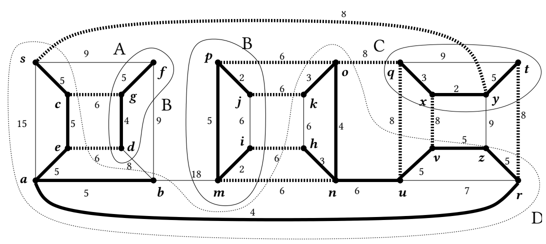
La figura quì sotto a sinistra visualizza un MST in linea spessa. Alla sua immediata destra classifichiamo gli archi di G in tre categorie:

linea spessa continua quelli che appartengono ad ogni MST

linea spessa tratteggiata quelli che appartengono a qualche MST ma non a tutti

linea sottile quelli che non appartengono ad alcun MST





Richiesta 9

Gli archi in linea spessa raccolgono i nodi in 4 isole A, B, C e D.

Ogni MST dovrà prendere almeno un arco del taglio che separa la componente A dal resto del grafo. Ogni tale arco ha un estremo in A e l'altro in D, pertanto la questione si riduce a scegliere esattamente uno di questi archi sapendo che sono intercambiabili. Converrà prendere un arco di peso minimo (6). Abbiamo due opzioni: arco cg oppure arco ed. In modo del tutto analogo, per collegare la componente B al resto del grafo andrà preso uno ed un solo arco di peso minimo con un estremo nella componente B e l'altro fuori, prendere più di un arco con un estremo in B non serve a nulla. Quindi questa scelta è tra 4 opzioni, e del tutto indipendente dalla scelta precedente. Infine, per saldare la componente C al blocco $D + A + B$ dobbiamo scegliere un arco di peso 8 (ci sono 5 opzioni).

Il numero totale di alberi ricoprenti di peso minimo è pertanto $2 \times 4 \times 5 = 40$.

Richiesta 10

Forniamo ora dei certificati specifici per la classificazione dei 4 archi per cui richiesto:

arco mb in nessun MST in quanto arco di peso strettamente massimo nel ciclo $mnurab$.

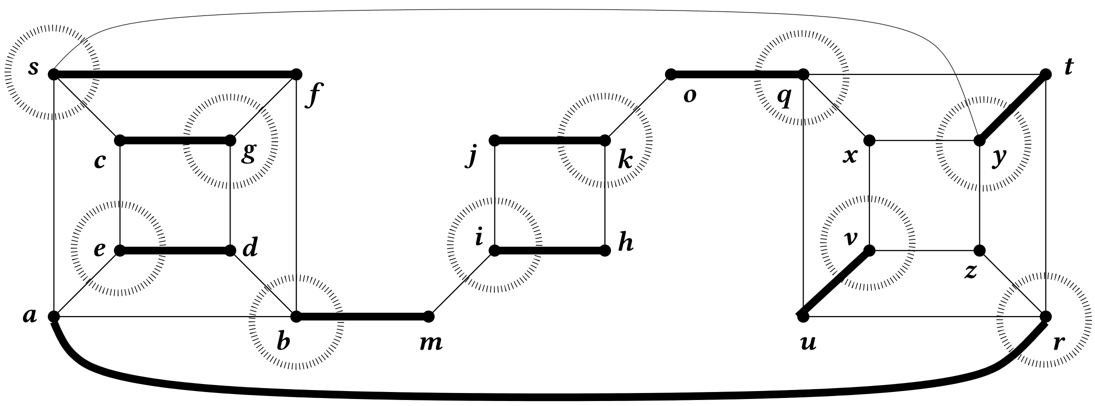
arco mp in tutti gli MST in quanto arco di peso strettamente minimo del taglio di spiaggia $\{m, i\}$ (ossia del taglio che separa i nodi m ed i dal resto del grafo).

arco mi in tutti gli MST in quanto arco di peso strettamente minimo del taglio di spiaggia m (ossia del taglio che separa m dal resto del grafo, detto anche la stella di m).

arco mn in qualche MST in quanto arco di peso minimo del taglio che separa la componente D dal resto del grafo (o anche il lato sinistro dal lato destro del grafo per come in figura). Non in tutti in quanto arco di peso massimo nel ciclo $mnop$.

Richiesta 11

Nel grafo G_{rs} evidenziamo un matching M ad un node cover X con $|X| = |M|$, che mutualmente certificano la propria ottimalità dato che in nessun grafo possono aversi un matching M ed un node cover X con $|X| < |M|$ (di ogni arco di un matching ogni node cover deve prendere almeno uno dei due estremi). Siccome il grafo era bipartito era di fatto garantito che una tale coppia di oggetti dovesse esistere.



CONSIGLI SU COME PREPARARSI ALL'ESAME

Per conseguire un voto per l'insegnamento di Ricerca Operativa devi partecipare ad un appello di esame. Il primo appello d'esame di ogni anno accademico ha luogo a giugno, dopo la conclusione del corso. Per gli appelli estivi in aula delta, non abbiamo controllo dell'aria condizionata e l'ambiente potrà risultarvi troppo freddo. Data la durata dell'appello consiglio di portarsi golfini, snack, acqua e matite o pennarelli colorati. Potete portarvi materiali cartacei ma non è consentita alcuna strumentazione elettronica. Dovete portare il tesserino col vostro numero di matricola.

Durante l'esame, dovrete lavore per almeno 4 ore a quella che definisco «una prova di cromatografia su carta». Serve per riconoscervi con ragionevole confidenza quanto avete lavorato, appreso, sedimentato. E trasformare questo in una proposta di voto la più congrua possibile. La logica dello svolgimento dell'esame deve essere quella di dimostrare al meglio le competenze acquisite andando con efficienza a raccogliere, dei molti punti messi in palio a vario titolo: cercate e concretizzate quelli che più vi convengono, non impegolatevi a dimostrare quello che non sapete o dove incontrate incertezze. Contano le risposte corrette, fornite in chiarezza, ed i certificati (in questo la struttura dell'esame ribadisce il ruolo metodologico ubiquito dei concetti di complessità computazionale propagandati nel corso). Tutto il resto (incluse le castronerie colossali ma anche le doppie risposte discordanti) non verrà conteggiato.

Ricordate che in buona sostanza il voto corrisponderà al punteggio positivamente raccolto. I punti messi in palio ad ogni tema eccedono significativamente quanto necessario al raggiungimento dei pieni voti, gestitevi quindi per dimostrare le competenze che avete, senza impelagarvi dove avete invece delle lacune. Non ci interessano le vostre mancanze o lacune quanto piuttosto quello che dimostrate di saper fare.

L'esame presenta diverse tipologie di esercizi e domande su vari aspetti di quanto esposto a lezione. Nel prepararti all'esame, prendi a riferimento i testi e le correzioni dei temi precedenti che trovi al sito del corso:

`http://profs.sci.univr.it/~rrizzi/classes/RO/index.html`

Ogni esercizio è anche un'opportunità di apprendimento e di allenamento, sfruttalo al meglio senza sprecarlo. Una prima utilità è quella di testare la tua preparazione all'esame. Dopo aver letto il testo, consigliamo pertanto di svolgere l'esercizio quantomeno nella propria mente. Ma, in sufficiente numero di esemplari, poi anche materialmente, prestando attenzione ai tempi impiegati ed ai punti conseguiti. Solo a valle di un'esperienza almeno parziale con l'esercizio, passa alla lettura del documento di correzione. Se non sai come affrontare l'esercizio, sbircia sì la correzione, ma cercando di utilizzarla solo come suggerimento, cercando di riacquisire quanto prima autonomia nella conduzione dell'esercizio.

E se invece ti sembra di saper risolvere del tutto l'esercizio? Beh, a questo punto vale il converso: controlla che quanto hai in mente come soluzione corrisponda a quanto considerato e proposto come svolgimento opportuno. Nel confronto con la correzione proposta, presta attenzione non solo alle risposte in sè, ma anche a come esse vadano efficacemente offerte all'esaminatore/verificatore, ossia alla qualità dei tuoi certificati, alla precisione della tua dialettica, a come ottemperi il contratto implicito nella soluzione di un problema ben caratterizzato. In un certo senso, questo ti consentirà di raggiungere pragmaticamente quella qualità che in molti chiamano impropriamente ordine'", che ha valore e giustamente finisce, volenti o nolenti, per essere riconosciuta in ogni esame della vita. Ordine, ma noi preferiamo chiamarlo saper rispondere in chiarezza alla consegna"" non significa bella calligrafia o descrizioni prolisse, ma cogliere tempestivamente gli elementi salienti, quelli richiesti da contratto più o meno implicito. In questo le competenze che abbiamo messo al centro di questo insegnamento di ricerca operativa potranno renderti più consapevolmente ordinato. Lo scopo del documento di correzione non è tanto quello di spiegare come l"esercizio vada risolto ma piuttosto come le risposte vadano adeguatamente esibite pena il mancato conseguimento dei punti ad esse associati. Aggiungo che per le tipologie di esercizio classiche, descrizioni curate dei più noti algoritmi risolutori possono essere facilmente reperite altrove (perchè non collaborare a raccogliere una ricca collezione di link a tali sorgenti?).