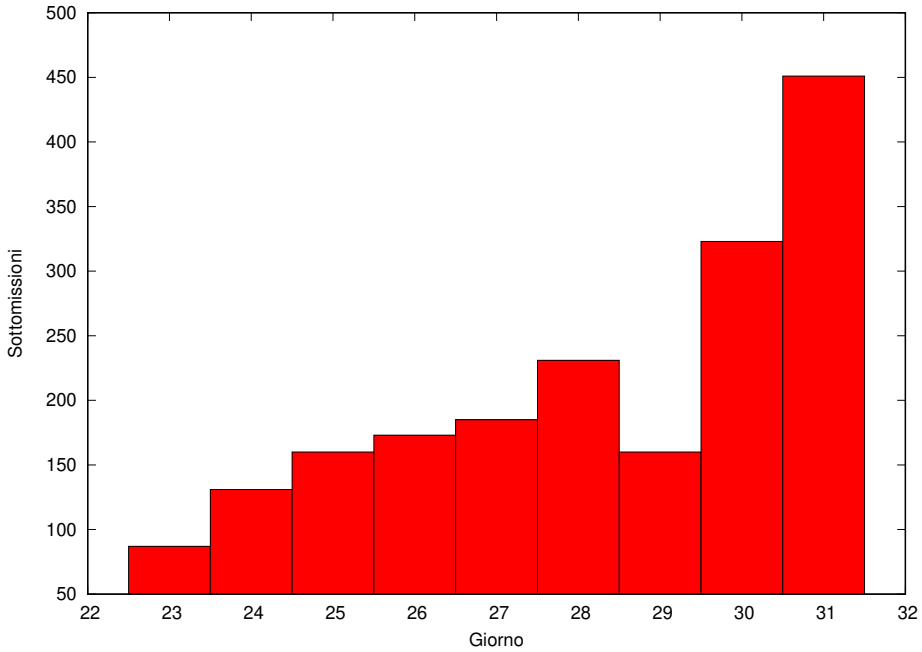
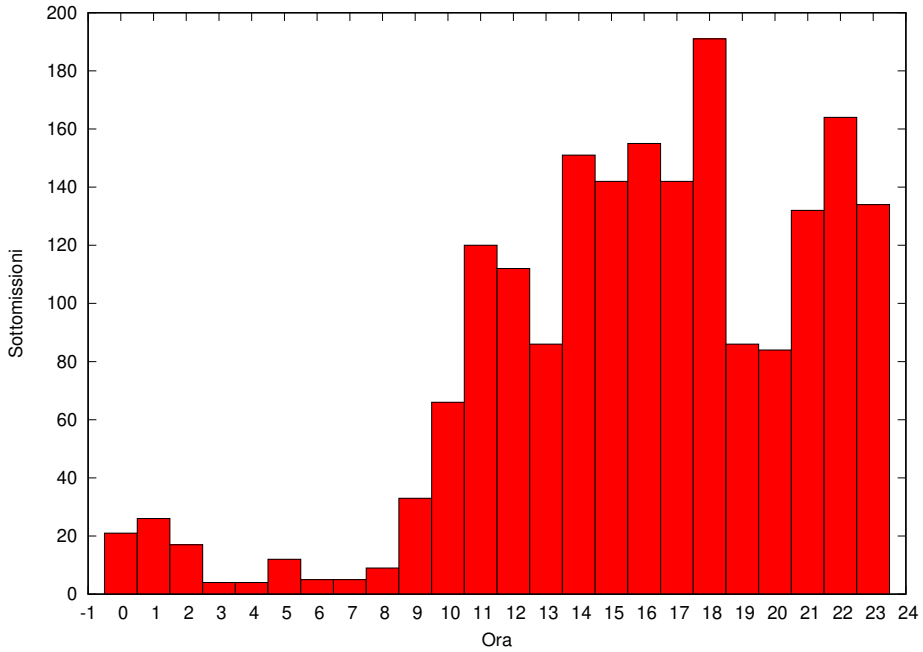
Secondo Progetto ASD 2023/2024

Numero sottoposizioni: `1901`





- 63 gruppi hanno fatto almeno una sottoposizione, di cui 59 hanno raggiunto la sufficienza;
- 191 studenti iscritti, di cui 160 appartenenti a gruppi che hanno fatto almeno una sottoposizione;

Risultati

Punteggi

- $P < 30$ $\longrightarrow$ progetto non passato.
- $30 \leq P < 70$ $\longrightarrow$ 1 punto bonus (21 gruppi).
- $70 \leq P < 98$ $\longrightarrow$ 2 punti bonus (14 gruppi).
- $98 \leq P < 99.2$ $\longrightarrow$ 3 punti bonus (19 gruppi).
- $P \geq 99.2$ $\longrightarrow$ 3.5 punti bonus (5 gruppi).

Classifiche e sorgenti sul sito (controllate i numeri di matricola):

https://judge.science.unitn.it/slides/asd23/classifica_prog2.pdf

Problema

Vi è dato un grafo bipartito $G = (V_c \cup V_g, E)$, dove V_c sono i vertici rappresentanti i centrini, V_g sono i vertici rappresentanti i gomitoli ed E sono gli archi rappresentanti i fili. Vi è inoltre fissato un ordinamento verticale sui vertici, dato inizialmente dai loro ID.

Sapendo di poter permutare arbitrariamente i vertici V_c ma di dover mantenere fisso l'ordinamento dei vertici V_g:

- qual è il minimo numero di incroci K tra gli archi E ottenibili permutando l'ordine dei vertici V_c?
- qual è l'ordinamento dei vertici V_c che induce K incroci tra gli archi E?

In letteratura, questo problema è noto come **one-sided crossing minimization problem**.

Sottoproblema: numero di incroci

Data una permutazione arbitraria di V_c, dove $\texttt{ord}[c]$ rappresenta la posizione del centrino c nella permutazione, e l'ordinamento fisso iniziale di V_g, vogliamo calcolare il numero di incroci tra gli archi.

Osservazione

Dati due archi (c, g) e (c', g') con $c, c' \in V_c$ e $g, g' \in V_g$.

(c, g) e (c', g') si incrociano se e soltanto se:

- $(\texttt{ord}[c] < \texttt{ord}[c'] \wedge g > g') \vee (\texttt{ord}[c] > \texttt{ord}[c'] \wedge g < g')$

Sottoproblema: numero di incroci

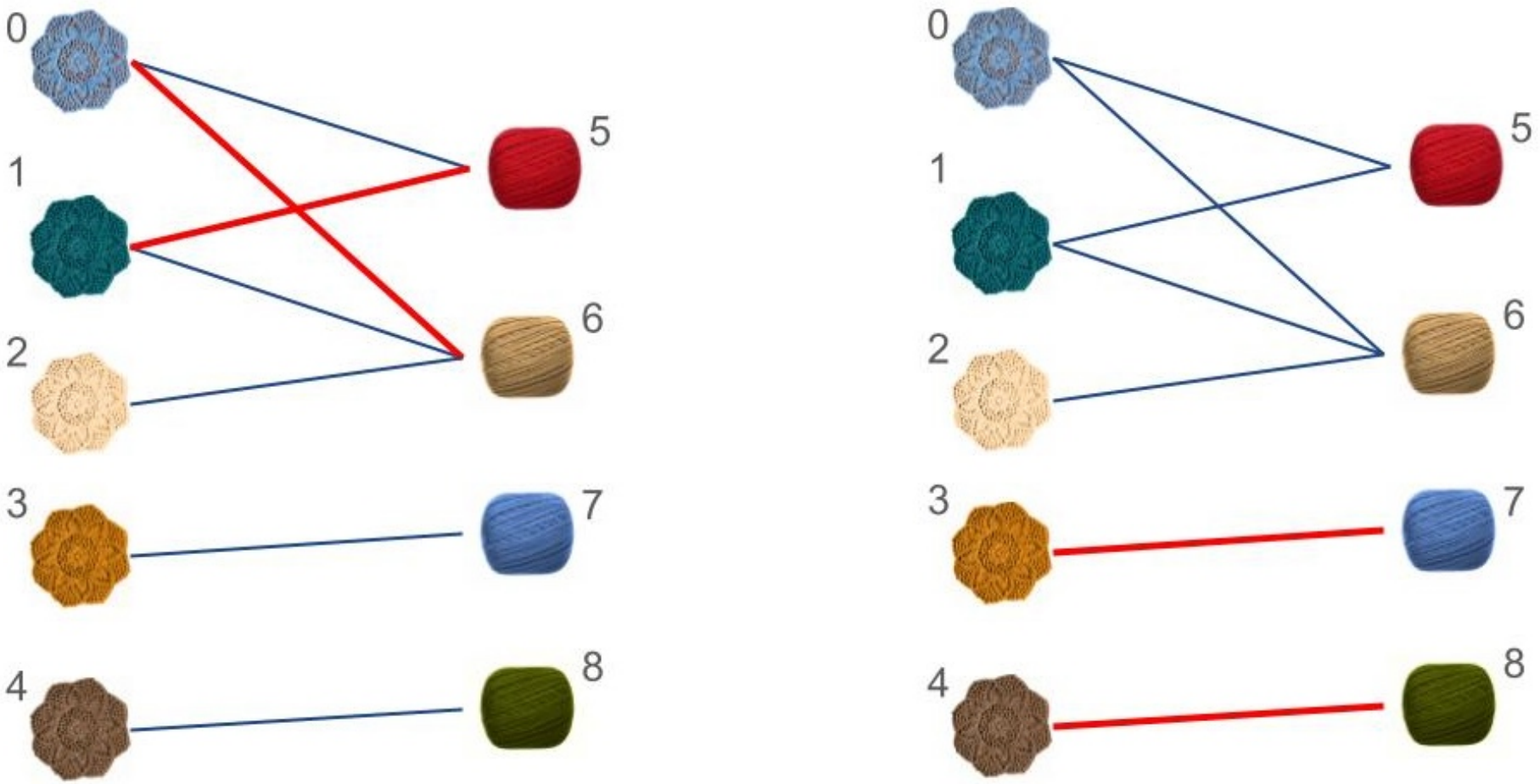


Figura: Caso archi con incrocio a sinistra e archi senza incrocio a destra.

Sottoproblema: numero di incroci

Ai fini del progetto, per risolvere questo sottoproblema basta semplicemente un algoritmo $O(|E|^2)$ che compara tutti gli archi e conta quelli che si incrociano.

Osservazione

Il sottoproblema può essere risolto con un algoritmo di complessità $O(|E| \log |V_g|)$ che sfrutta strutture dati particolari come Segment Tree o Fenwick Tree[a].
Non è necessario, ma velocizzare questa parte può permettervi di provare più soluzioni (e.g., local search).

[a] Wilhelm Barth, Michael Jünger e Petra Mutzel. "Simple and efficient bilayer cross counting". In: Graph Drawing. Springer. 2002, pp. 130–141.

Sottoproblema: numero di incroci

- Avevamo 6 casi di test in cui la permutazione in input dei vertici V_c induceva il minimo numero di incroci possibili

  $\Longrightarrow$ per risolvere questi 6 casi bastava contare il numero di incroci K nel grafo dato in input.

Complessità

- soluzione: `sol-30-count-n2.cpp`
- complessità:

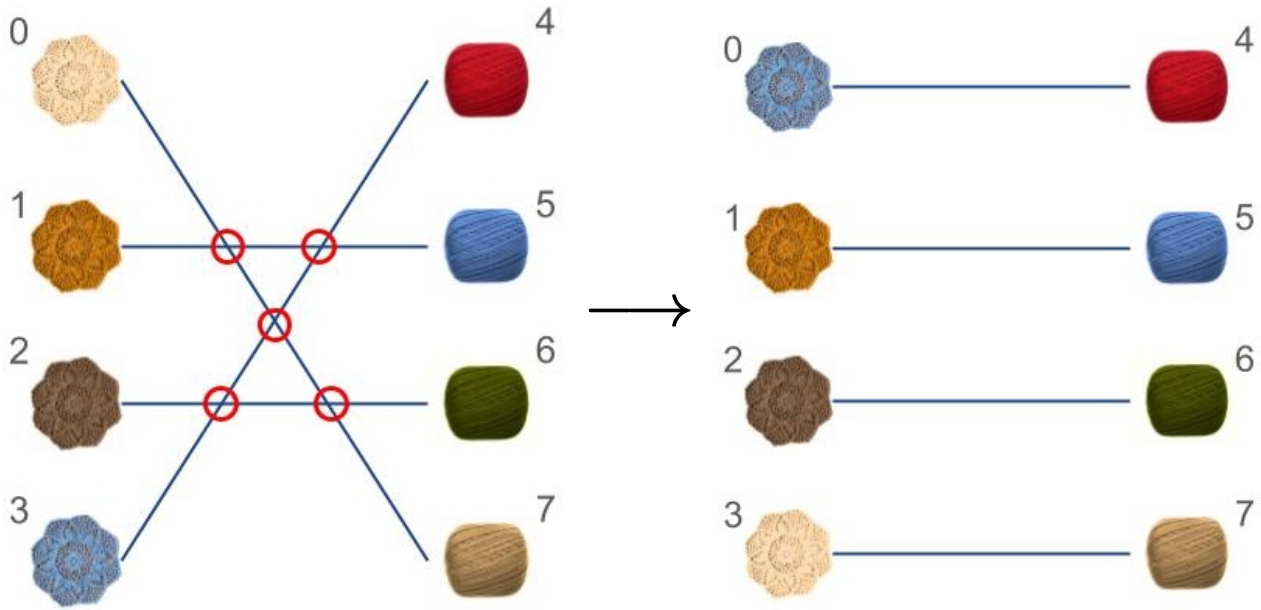
$$\mathcal{O}(|E|^2)$$

- 30 punti

Caso generale I: minimizzare gli incroci

Idea: decidiamo la posizione del centrino c_i in base ai gomitoli a cui è collegato: in un buon ordinamento, ogni centrino sarà "di fronte" ai gomitoli a cui è collegato.
Decidiamo un criterio di ordinamento dei centrini:

- Caso semplice: ogni centrino c_i è collegato a un solo gomitolo g_i
  $\implies$ se $g_i < g_j$, allora $\texttt{ord}[c_i] < \texttt{ord}[c_j]$



Caso generale I: minimizzare gli incroci

- Caso generale: ogni centrino c_i è collegato ai gomitoli $G_i = \{g_{i1}, \ldots, g_{ik}\}$
  $\implies$ definiamo una funzione $f : G_i \longmapsto \mathbb{R}$ per cui se $f(g_i) < f(g_j)$, allora $\texttt{ord}[c_i] < \texttt{ord}[c_j]$.

Buoni candidati per f sono:

- media
- mediana
- varianti più o meno fantasiose

Possiamo provare più strategie di ordinamento e prendere quella da cui risulta il numero minore di incroci.

Complessità

⇒ soluzione: `sol-95-media-mediana.cpp`

⇒ complessità:

$$\mathcal{O}(|V_c| \log |V_c| + |E|^2)$$

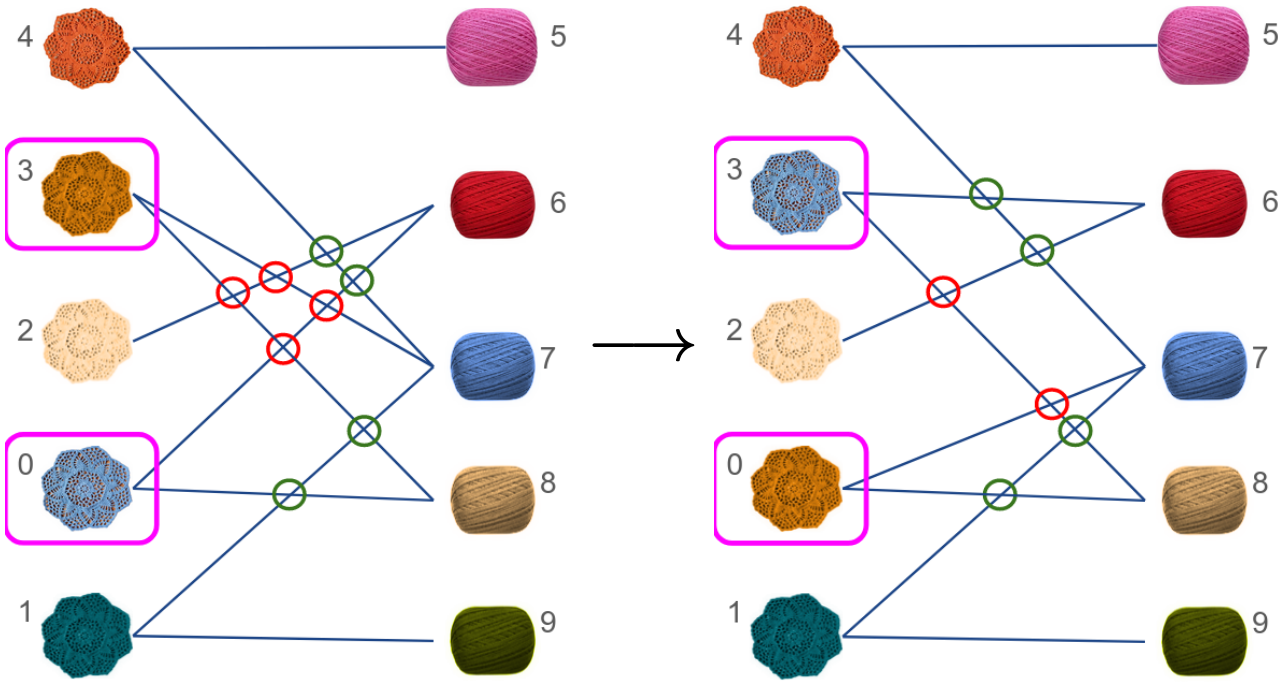
⇒ $\approx$ 95 punti

Migliorare una soluzione: ricerca locale

Dato un ordinamento dei centrini, possiamo provare a scambiare due centrini e vedere se il numero di incroci diminuisce.

Osservazione

Se scambiamo due centrini c_i e c_j tali che $\texttt{ord}[c_i] < \texttt{ord}[c_j]$, gli unici incroci che variano sono quelli tra i fili dei centrini c per cui $\texttt{ord}[c_i] \leq \texttt{ord}[c] \leq \texttt{ord}[c_j]$.



Migliorare una soluzione: ricerca locale

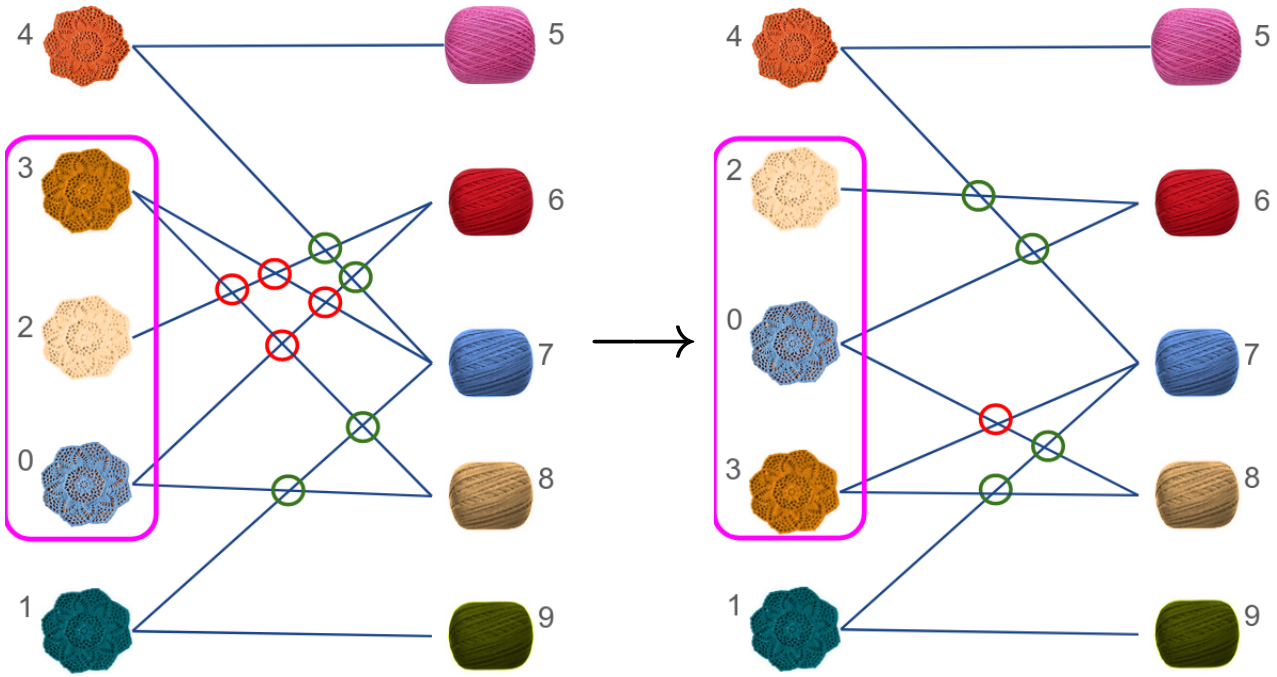
- Se scambiamo due centrini c_i, c_j consecutivi nell'ordinamento, ossia tali che $\texttt{ord}[g_i] + 1 = \texttt{ord}[g_j]$, allora

$$\texttt{nuovo\_costo} = \texttt{costo\_corrente} - c_{c_i,c_j} + -c_{c_j,c_i}$$

  dove $c_{u,v}$ è il numero di incroci tra i fili che toccano u e quelli che toccano v assumendo $\texttt{ord}[u] < \texttt{ord}[v]$.
- Oppure possiamo scegliere una coppia di centrini non consecutivi e ricalcolare il costo del sotto-array. Le coppie da scambiare possono essere scelte in modo random, oppure secondo qualche euristica (es. numero di gomitoli collegati, varianza nella posizione dei gomitoli collegati, ecc.)

Miglioramento ulteriore: blocchi ottimi

- Sappiamo che calcolare l'ordinamento ottimo di tutti i centrini non è fattibile in tempi ragionevoli.
- Sappiamo però che ordinando un blocco di centrini consecutivi, gli unici incroci che variano sono quelli tra i centrini nel blocco.
- Possiamo cercare un ordinamento ottimo all'interno di blocchi abbastanza piccoli da essere risolti velocemente.



Miglioramento ulteriore: blocchi ottimi

Come risolvere un blocco B in modo ottimo?

- Soluzione naive: provo ogni permutazione dei nodi, scelgo quella con il numero minore di incroci.
  $\implies$ Complessità: $O(|V_c^B|! \cdot |E^B|^2)$,
  dove V_c^B e E^B sono rispettivamente i nodi in V_c che fanno parte del blocco B e i fili che li toccano

  Quanto sono i grandi i blocchi che riusciamo a risolvere?
  Meno di 10 nodi: $10! \approx 3\,500\,000$

  Possiamo fare di meglio?

Miglioramento ulteriore: blocchi ottimi

- Soluzione basata su programmazione dinamica:
  Sia U l'insieme dei nodi di B non ancora ordinati.
  Sia DP[U] il numero di incroci in un ordinamento ottimo di U.

$$\text{DP}[U] = \begin{cases} 0 & \text{se } U = \emptyset \\ \min\limits_{u \in U} c_{u,U} + \text{DP}[U \setminus \{u\}] & \text{altrimenti} \end{cases}$$

  dove $c_{u,U}$ è il numero di incroci tra i fili di u e i fili degli altri centrini in U, assumendo un ordinamento di U in cui u è primo.

  $\implies$ Complessità: $O(2^{|V_c^B|} \cdot |E^B|^2)$. Riusciamo a risolvere blocchi fino a 20 nodi: $2^{20} \approx 1\,000\,000$.

Nota implementativa

In C++, un BitSet di dimensione fissa può essere rappresentato con `std::bitset`. Il metodo `to_ulong()` converte il bitset in un `unsigned long`, che possiamo usare come indice dell'array DP.

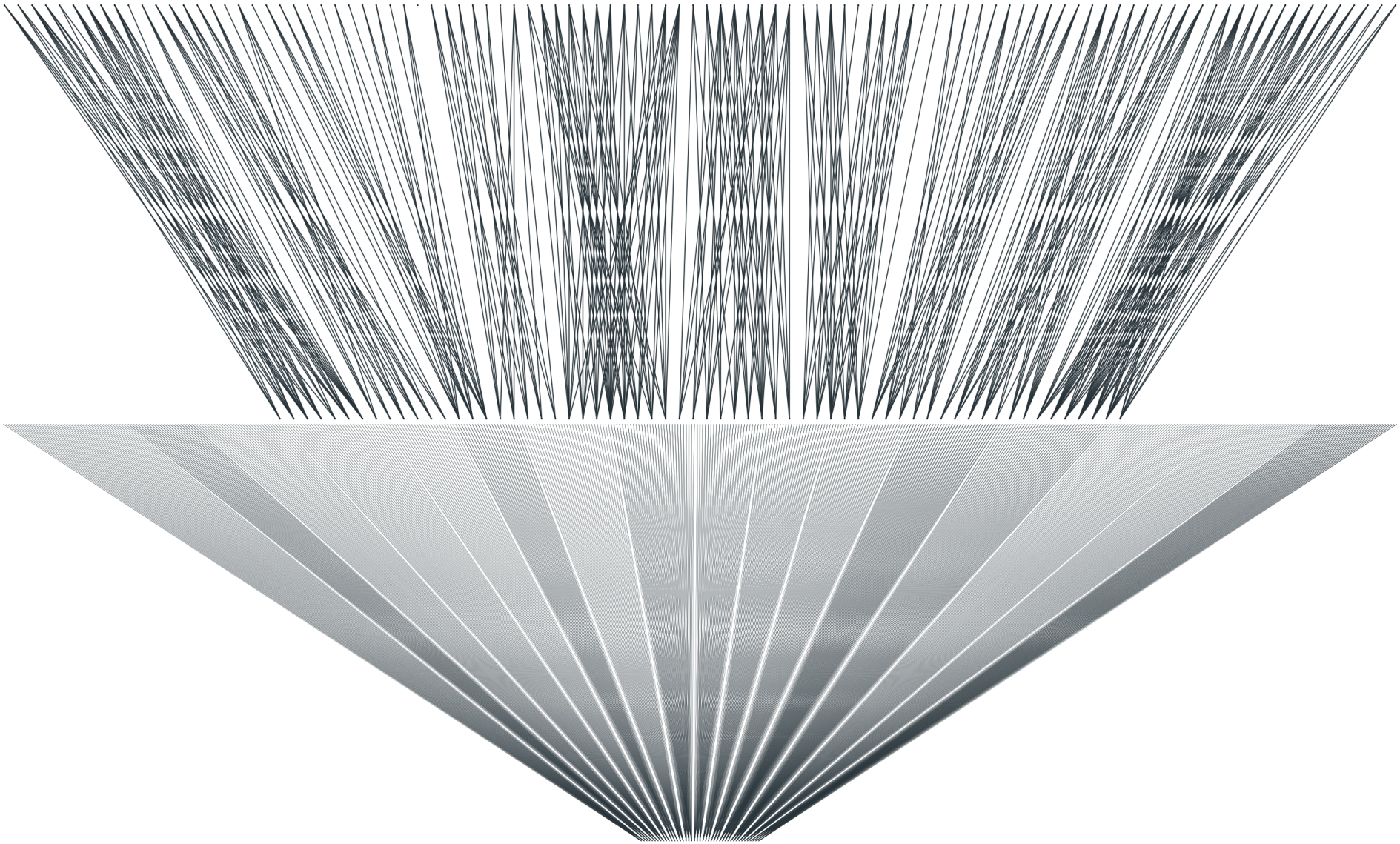
Nostra soluzione

⇒ Nella pratica:
- Troviamo un ordinamento buono usando l'euristica della media o della mediana
- Scegliamo in modo random blocchi di 10-15 nodi e li ordiniamo in modo ottimo.

⇒ soluzione: `sol-99-blocchi-ottimi.cpp`

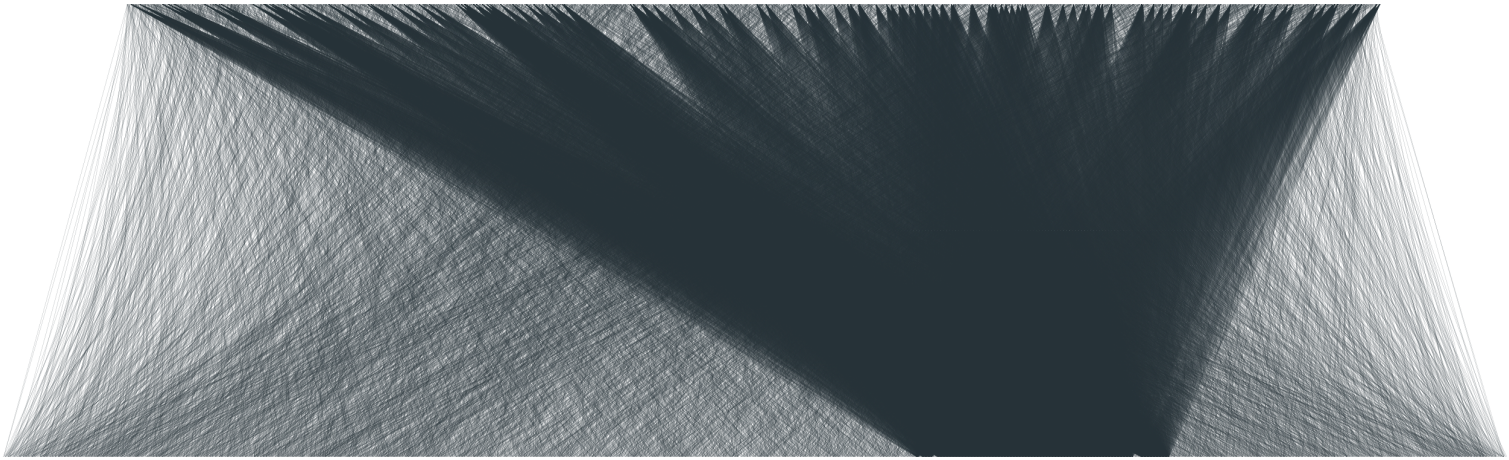
⇒ $\approx$ 99 punti

Test Case già ottimi: Sottografi indipendenti



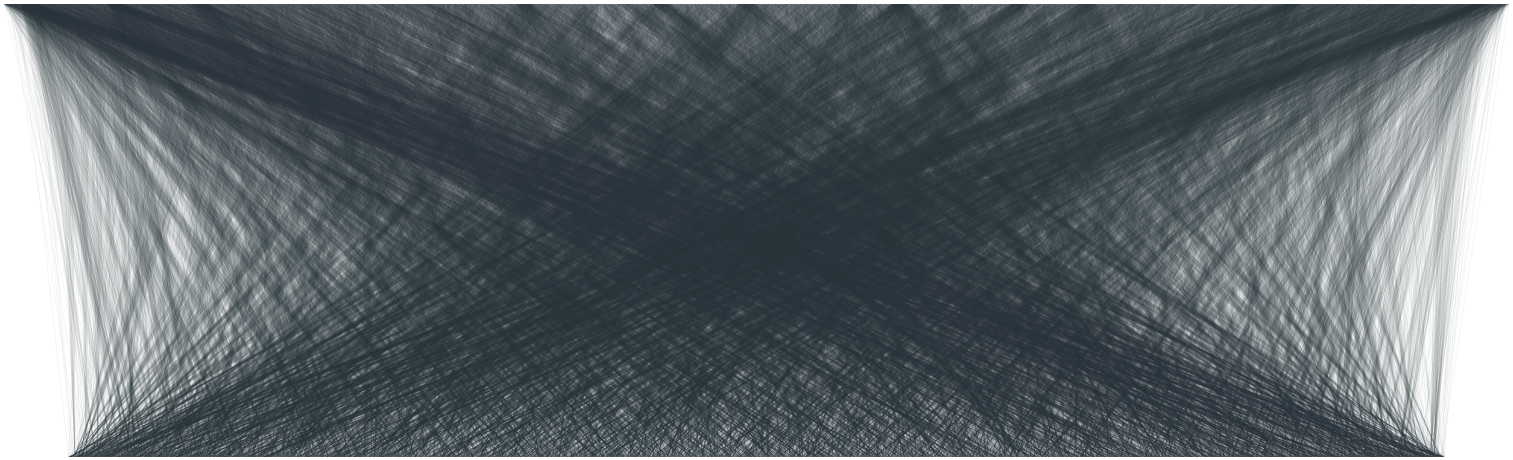
Test Case 16

(A) Input disordinato (centrini in basso, gomitoli in alto)

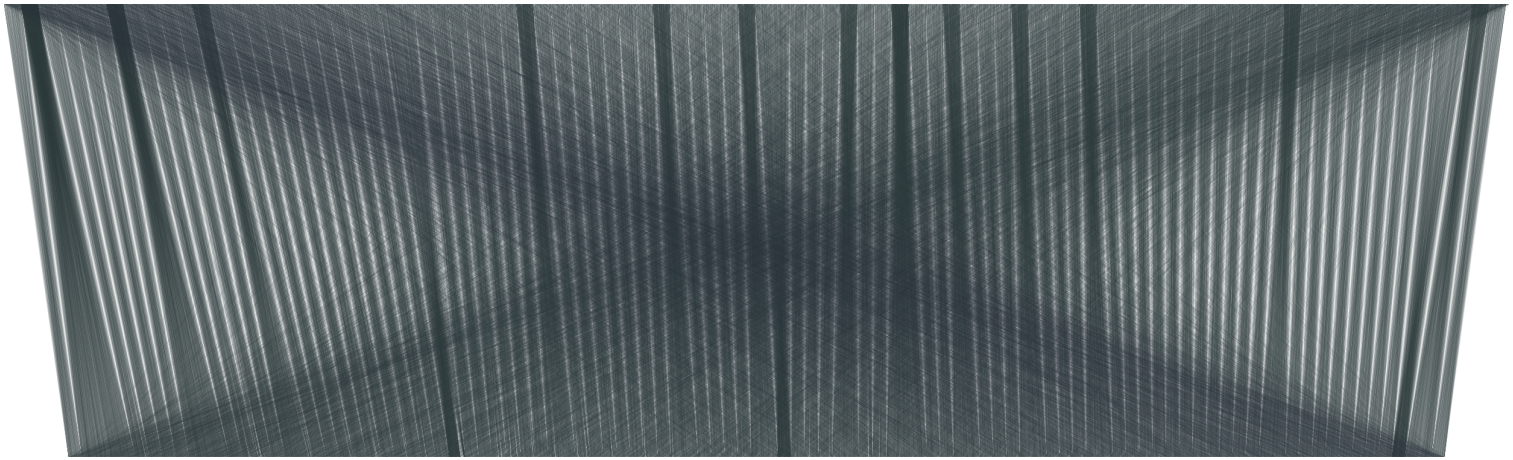


(B) Ordine della nostra soluzione

Test Case 17



(A) Input disordinato (centrini in basso, gomitoli in alto)



(B) Ordine della nostra soluzione