PyGFA

Progettazione e sviluppo di una libreria Python per la gestione di file GFA

Diego Lobba

matricola:795702

Università degli studi di Milano-Bicocca
Dipartimento di Informatica Sistemistica e Comunicazione

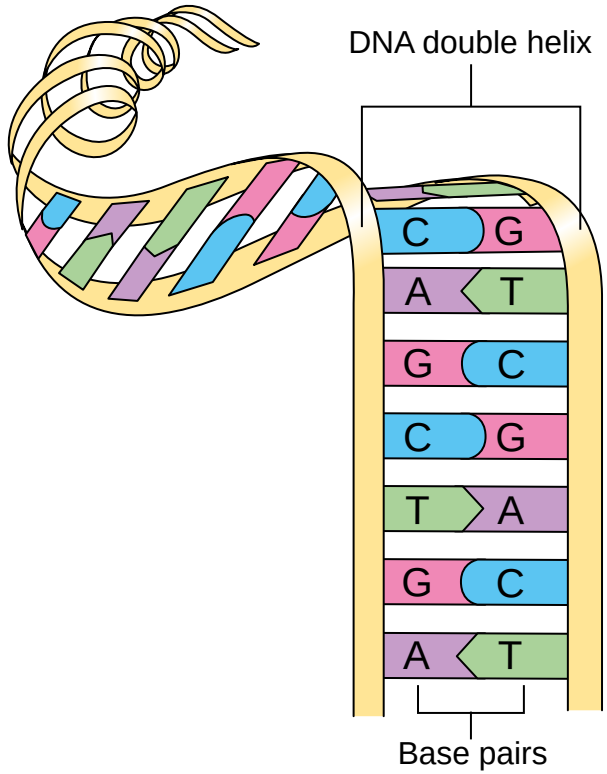
Relatore: Prof. Gianluca Della Vedova
Correlatore: Marco Previtali

Attività svolte

- Studio delle specifiche GFA
- Implementazione del sistema con NetworkX
- Costruzione dei casi di test e verifica della copertura del codice
- Benchmark per misurare le performance della libreria

Cos'è il DNA?

- DNA come stringa composta dalle lettere A, C, G, T
- Stringa ottenuta mediante riassemblaggio di sequenze più piccole ottenute da metodi NGS (Next Generation Sequencing)
- Rappresentare le informazioni di sequenziamento è un problema



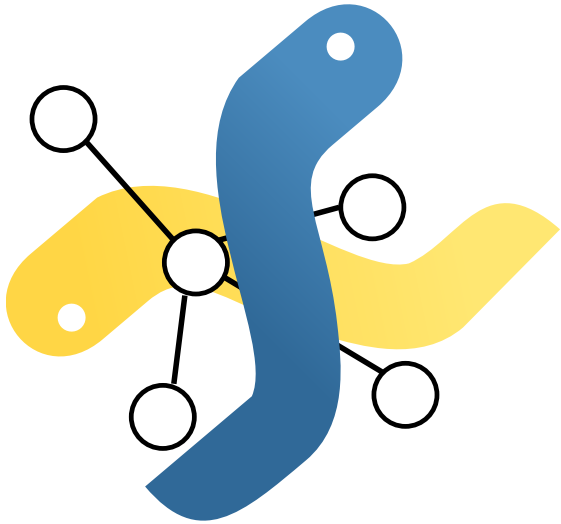
I file GFA

- Due specifiche
  - GFA1 pensata appositamente per grafi di assemblaggio
  - GFA2 più generica, superset di GFA1
- ogni linea rappresenta un concetto all'interno del grafo

```
S       s1      10      *
S       s2      10      *
S       s3      10      *
S       s4      10      *
E       ls1s2   s1+     s2+     7       9$      0       2       *
E       ls2s4   s2+     s4+     7       9$      0       2       *
E       ls1s3   s1+     s3+     7       9$      0       2       *
```

PyGFA

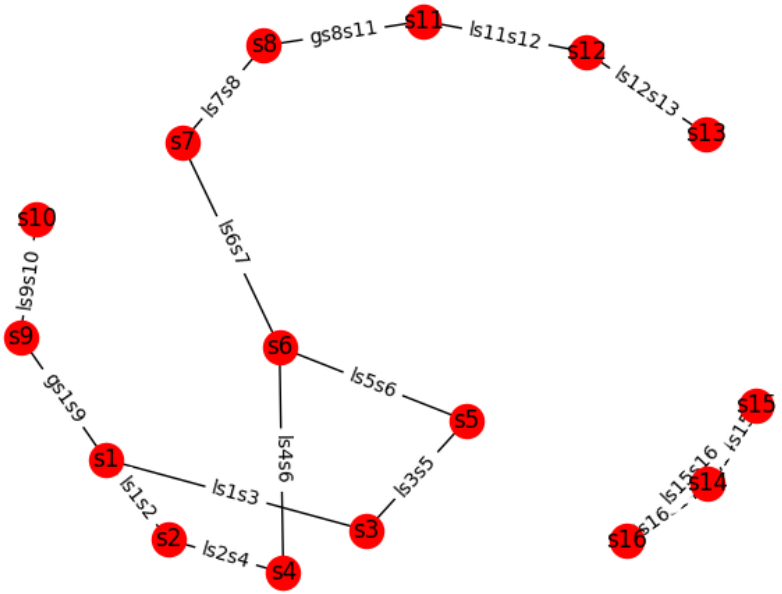
- É una libreria Python
- Gestisce le informazioni contenute nei file GFA
- Usa la classe **Multigrafo** offerta da NetworkX per contenere le informazioni ed eseguire operazioni sul grafo

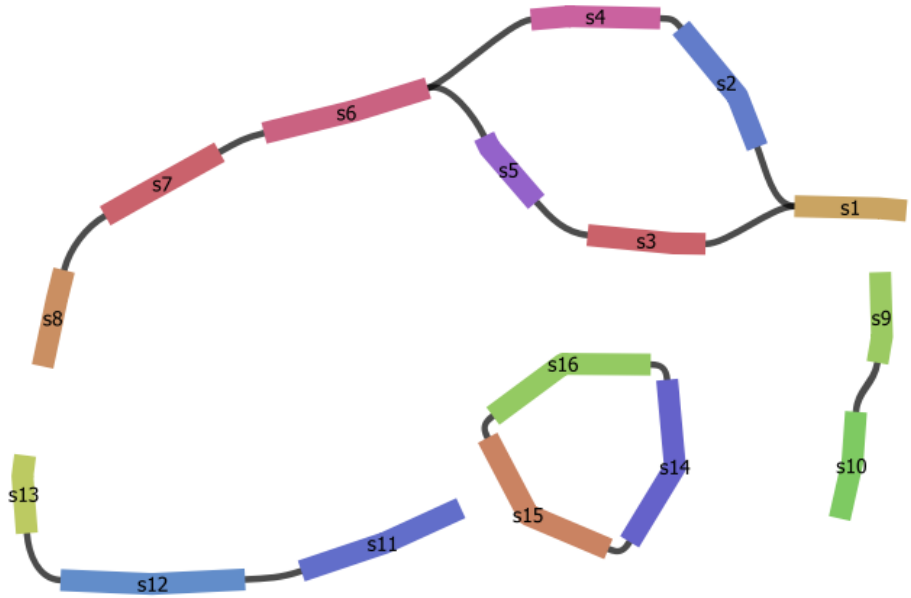


Logo di PyGFA.

Gestione delle informazioni

- da linea su file di testo a dizionario Python degli elementi del grafo
- gestione particolare degli archi di dovetail
- duplice visione del grafo



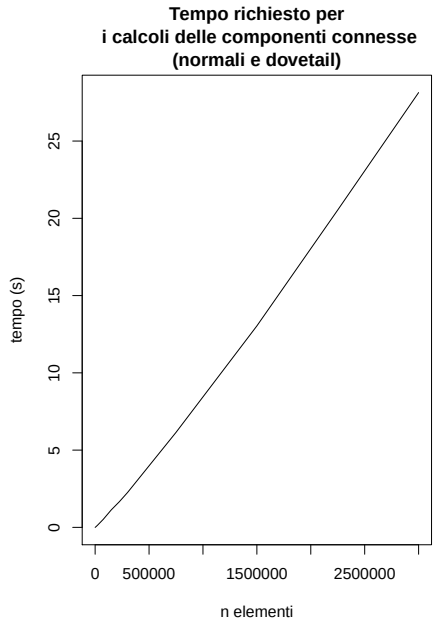


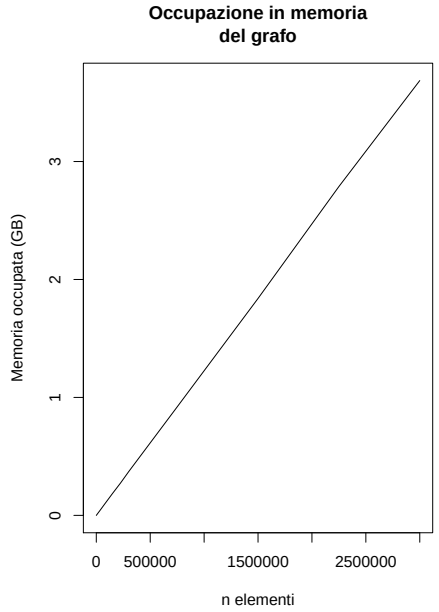
Funzionalità

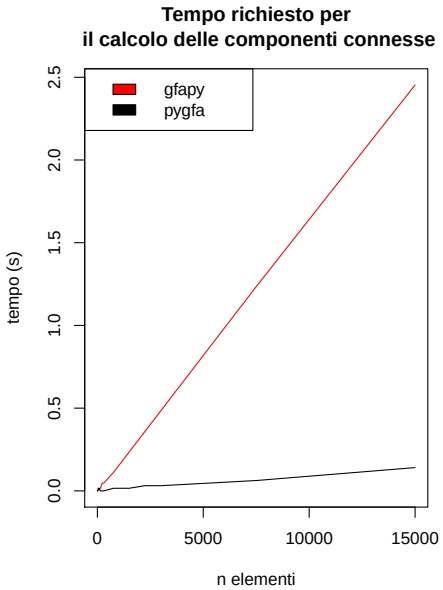
- calcolo delle componenti connesse
- calcolo dei percorsi racchiusi tra due nodi
- salvataggio del grafo in una delle due specifiche
- ricerca degli elementi del grafo utilizzando un comparatore definito dall'utente

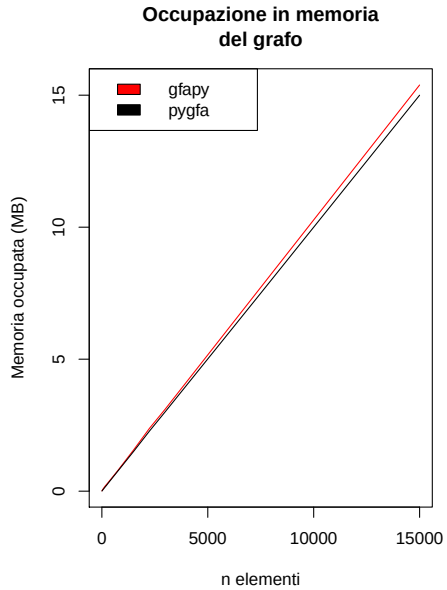
Benchmark

- Due serie di test dove:
  - si è analizzata la scalabilità di PyGFA
  - si è confrontata PyGFA con Gfapy
- PyGFA ha una grossa occupazione in memoria
- A parità di memoria occupata PyGFA è più prestante in termini di tempo









Metodo di sviluppo

- Extreme programming

  repeat

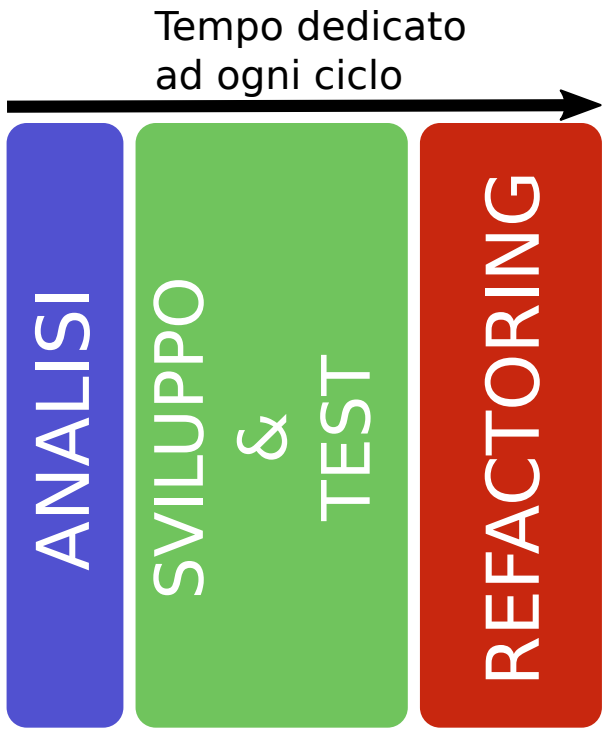
  Analizza

  Implementa $\wedge$ *Testa*

  Refactoring

  until *Fine sviluppo*

- Sviluppo basato sulle priorità
  - si implementa subito
  - si implementa affiancati dai casi di test



Strumenti di aiuto nello sviluppo

- Coverage.py
  - usato con unittest per verificare la copertura dei casi di test
  - 96% di copertura totale
- Pylint
- Sphinx e Read the Docs
  - estrazione della documentazione direttamente da codice
  - output in html con supporto mobile
  - hosting su piattaforma specifica

Conclusioni

- Stato di sviluppo:
  - l'attuale versione è stabile
  - necessità di un refactoring più accurato
  - piattaforma estendibile

FINE