MAZZETTI MASSIMO
0253467
PROGETTO2

RELAZIONE FINALE

Il progetto che ho sviluppato viene diviso in due file :

- *creagrafi*;
- *hasCycleUF*.

Nel file creaGrafi abbiamo le funzioni:

- *grafoCiclo*;
- *grafoAciclo*;
- *hasCycleDFS*;
- *Decorator*.

Nella funzione *grafoCiclo* viene creato un grafo con cicli attraverso una lista di adiacenza, ovvero che ogni nodo "u" ha una lista contenente il suo nodo adiacente, cioè i nodi "v" tali che esiste un arco(u,v).

Come funzione che crea il grafo abbiamo presto come punto di riferimento quella scritta dai tutor (*Graph_AdjacencyList.py*) con la differenza che il numero di nodi viene scelto dall'utente e, dopo la stampa dei nodi e degli archi, la funzione restituisce il grafo.

Nella funzione *grafoAciclo* viene creato un grafo senza cicli anch'esso attraverso una lista di adiacenza. Il grafo crea gli archi tra nodi in questo modo: (1,2),(2,3),(3,4),(4,5),..., (n-1,n).

Nella funzione *hasCycleDFS* viene chiamata una funzione presente nel file *graph.py* che prende come parametri il grafo ed il numero dei nodi, che usa l'approccio della visita in profondità (*Depth-First-Search*), ovvero che parte dalla radice e procede visitando nodi, di figlio in figlio, fino a raggiungere una foglia. La Funzione retrocede poi al primo antenato che ha

ancora figli non visitati, se esistono, e ripete il procedimento a partire da uno di questi figli.

Nel *decorator* viene creato un file in cui sono scritti i dati degli esperimenti eseguiti con il formato "n,t" su ogni riga dove "n" è il numero di archi del grafo e "t" il tempo di esecuzione dell'algoritmo.

Nel file *hasCycleUF.py* abbiamo tre funzioni *hasCycleUF* ,*un'iterator* e un *decorator*. *HasCycleUF* usa la struttura UnionFind, ovvero una struttura dati per insiemi. Ciascun insieme è identificato da un rappresentante, che è un elemento dell'insieme ed ogni elemento è rappresentato da un'oggetto. *HasCycleUF u*tilizza tre operazioni: *MakeSet*, *Find* e *Union*.

Come struttura ho deciso di utilizzare la *QuickUnion*, dato che dopo vari esperimenti risultava essere quella più veloce rispetto alla *Union* e, rispetto alla *Find,* è poco più lenta rispetto alla *QuickFind*. Dopo aver creato la *Union, la funzione* vede se esiste un ciclo analizzando tutti gli archi fino a quando non torna ad un nodo già visitato.

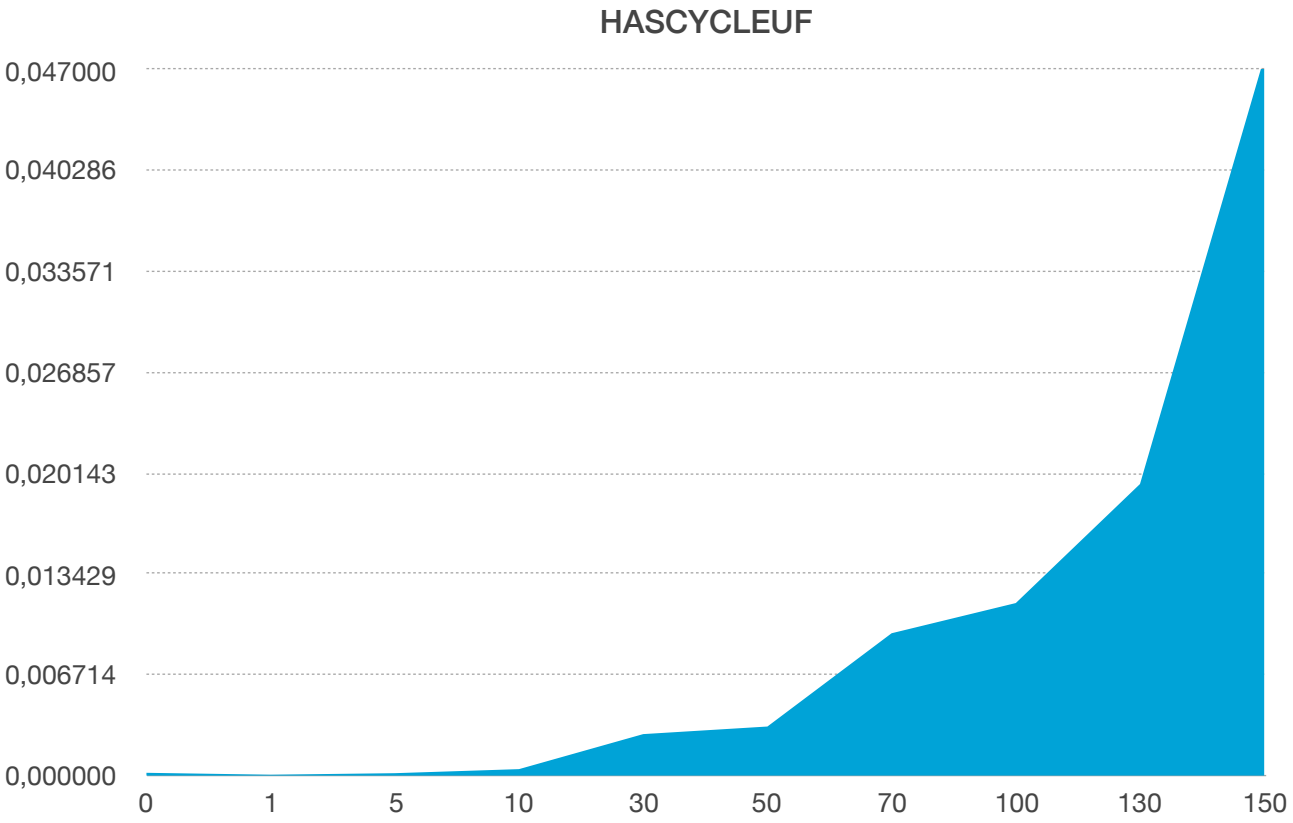
C'è *un'iterator* che scandisce tutti gli edge del grafo e il *decorator* uguale a quello di *creagrafi*.

MAZZETTI MASSIMO
0253467
PROGETTO2

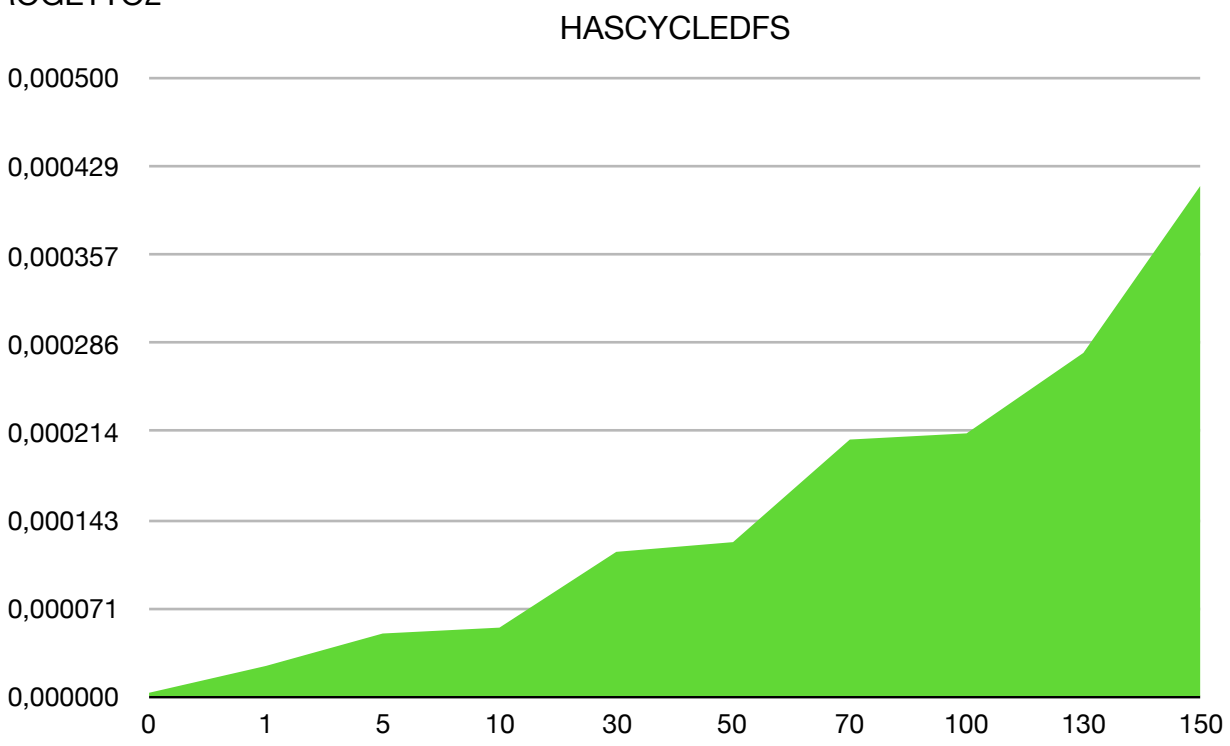
GRAFI CON CICLO

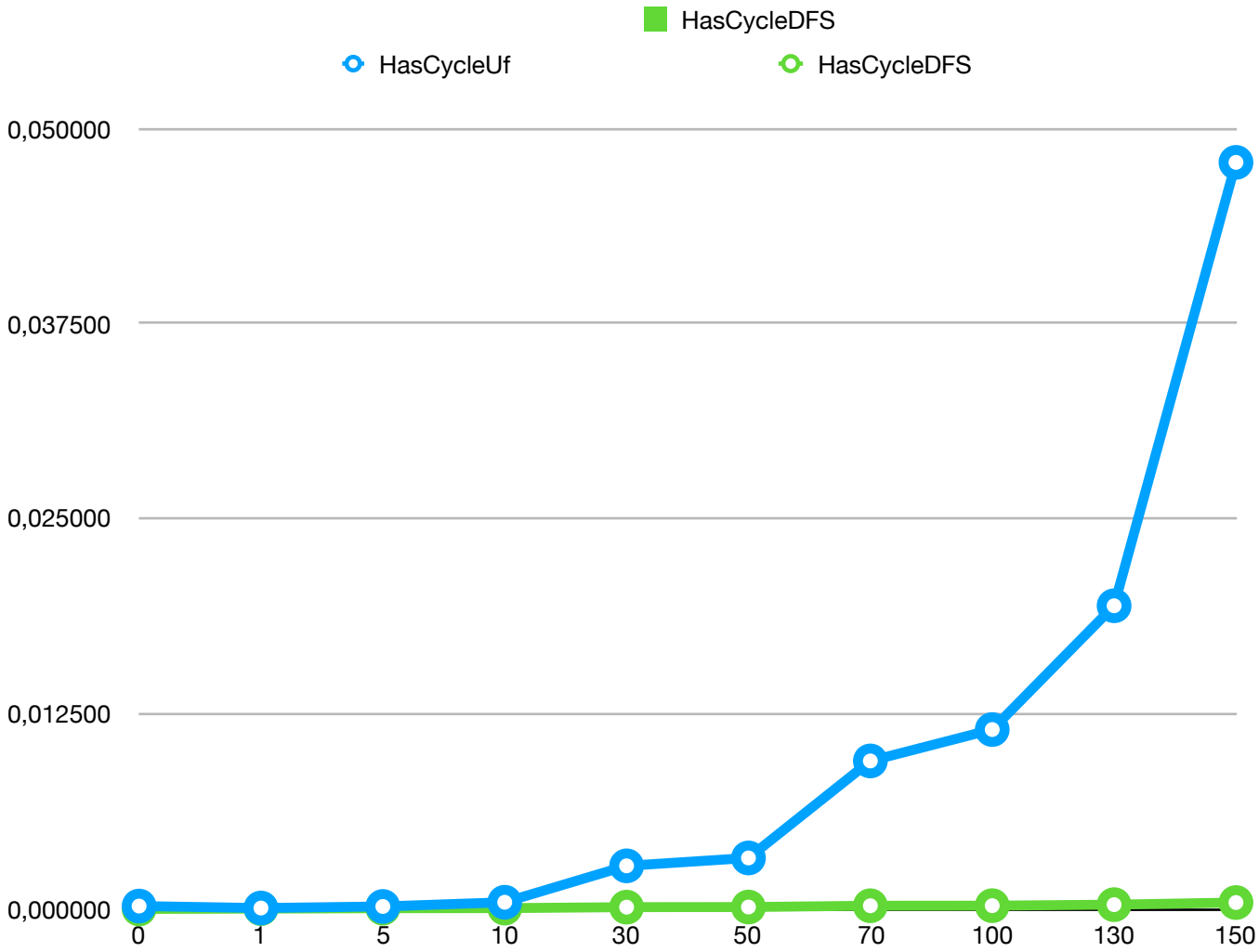
Descrizione	0	1	5	10	30
HasCycleUf	0,000187158584594727	0,00006103515625	0,000161647796630859	0,000431776046752293	0,002763986587524441
HasCycleDFS	0,000003099441528320 31	0,000024795532226562 5	0,000051021575927734 4	0,000055789947509765 6	0,000117063522338867

Descrizione	50	70	100	130	150
HasCycleUf	0,003268957138061 52	0,009480237960815 43	0,011491060256958	0,019418001174926 8	0,047809839248657 2
HasCycleDFS	0,000124931335449219	0,000207901000976562	0,000212907791137695	0,000277996063232422	0,000412940979003906



MAZZETTI MASSIMO
0253467
PROGETTO2





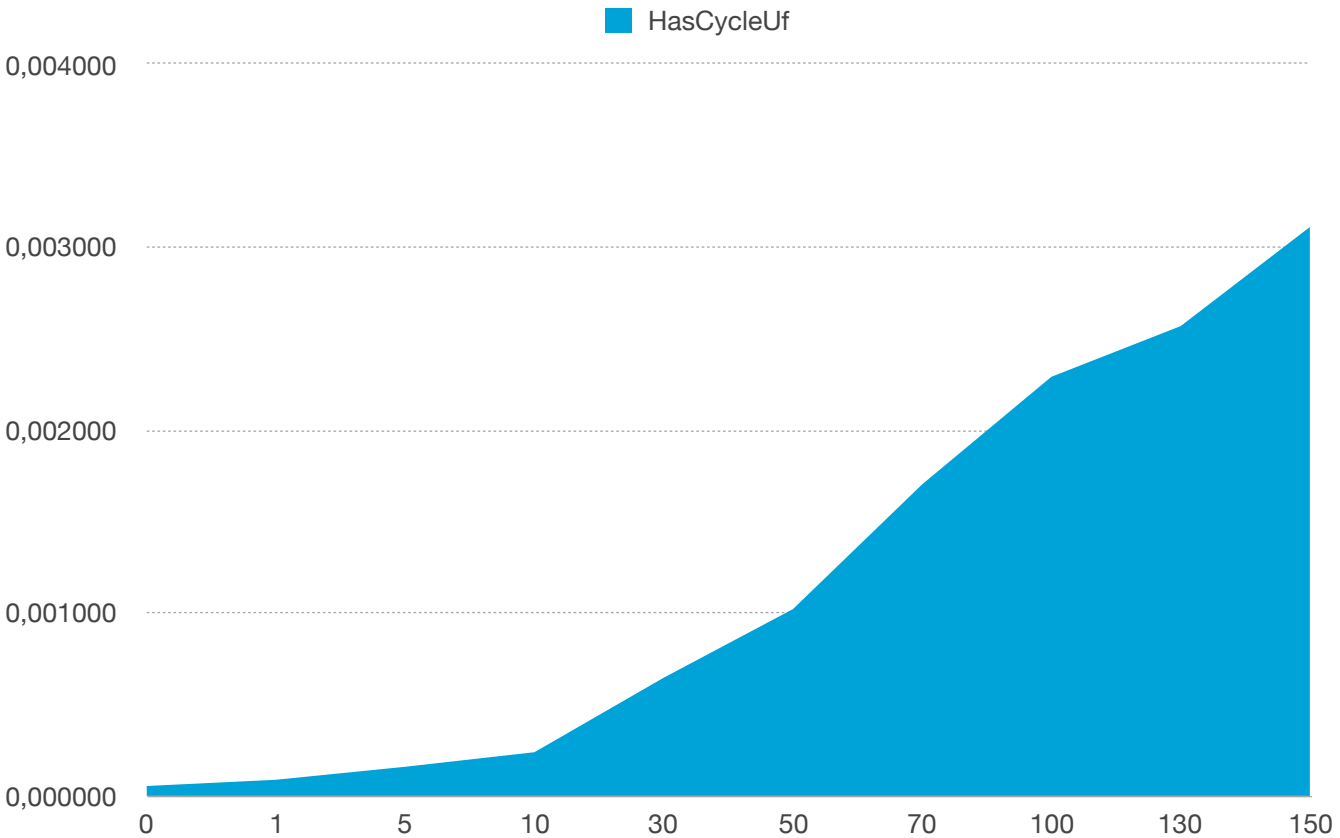
MAZZETTI MASSIMO
0253467
PROGETTO2

Osservando i grafici vediamo che l'algoritmo *HasCycleUF* cresce in modo stabile fino a 50 elementi, dopodiché inizia a cresce più rapidamente.
L'algoritmo *HasCycleDFS* cresce in modo stabile.
Confrontando entrambi gli algoritmi riusciamo a vedere che *HasCycleDfs* è nettamente più veloce rispetto a *HasCycleUF.*

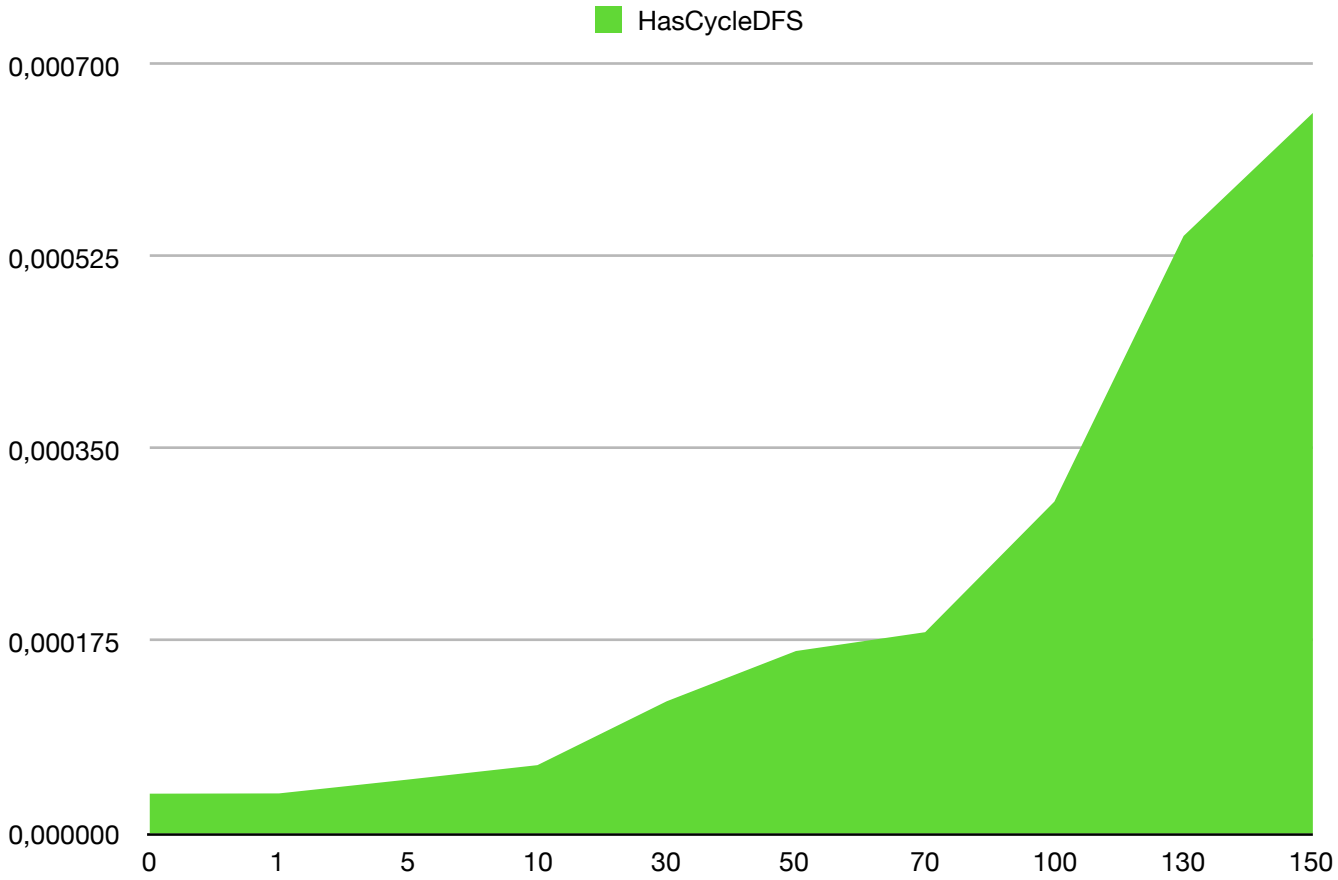
GRAFI SENZA CICLO

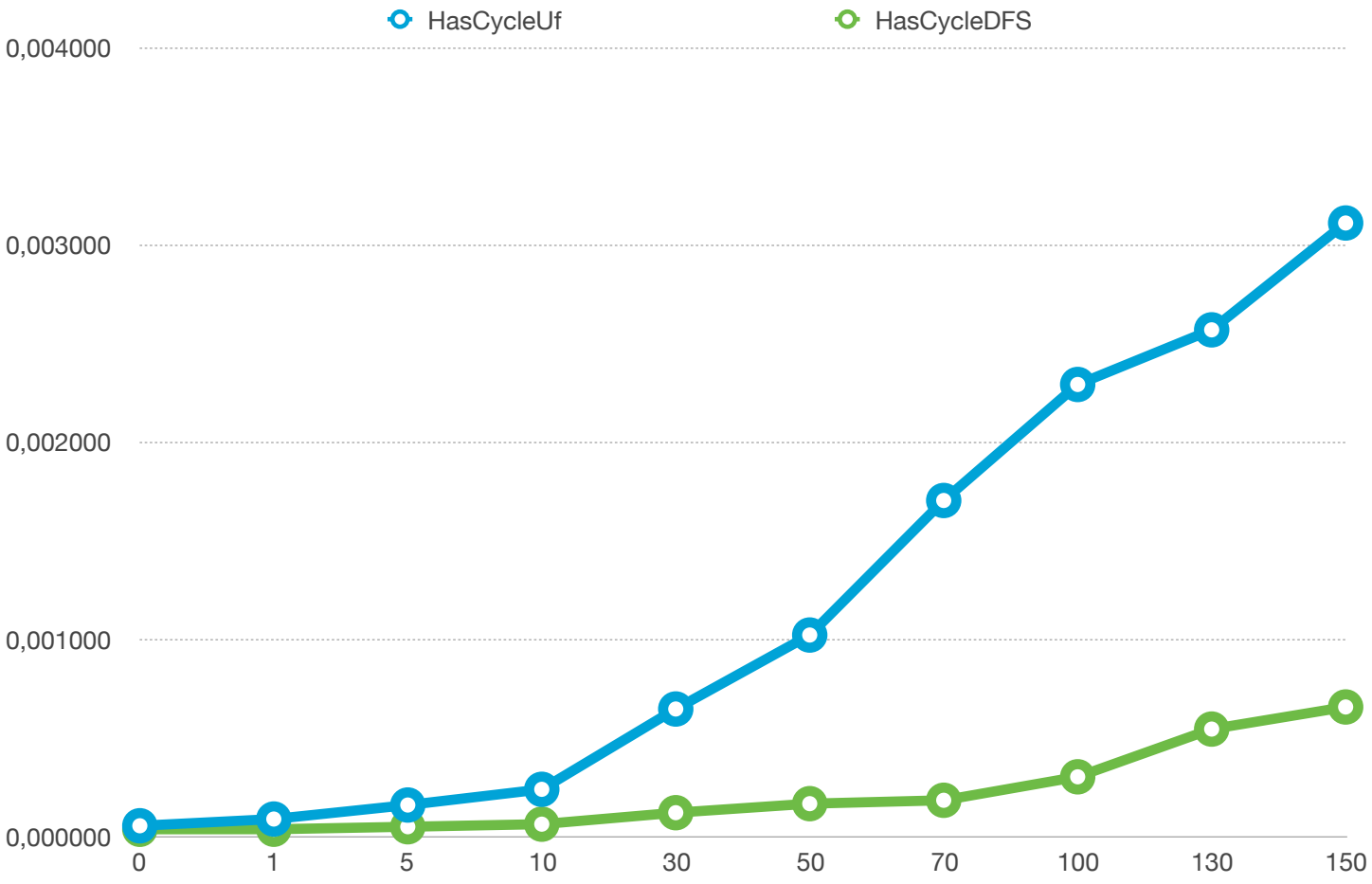
Descrizione	0	1	5	10	30
HasCycleUf	0,0000538825988769531	0,0000879764556884766	0,0001590251922 60742	0,0002386569976 80664	0,0006458759307 86133
HasCycleDFS	0,0000360012054443359	0,0000362396240234375	0,0000488758087158203	0,0000619888305664062	0,0001201629638 67188

Descrizione	50	70	100	130	150
HasCycleUf	0,0010209083557 1289	0,0017030239105 2246	0,0022912025451 6602	0,0025682449340 8203	0,0031099319458 0078
HasCycleDFS	0,0001659393310 54688	0,0001831054687 5	0,0003020763397 2168	0,0005440711975 09766	0,0006561279296 875



MAZZETTI MASSIMO
0253467
PROGETTO2





MAZZETTI MASSIMO
0253467
PROGETTO2

L’algoritmo *HasCycleUF cresce poco fino a 10 elementi dopodiché inizia a crescere più rapidamente.*
L’algoritmo HasCycleDFS cresce stabilmente fino a 70 elementi, poi cresce molto più velocemente.
Confrontandoli vediamo che HasCycleDFS è più veloce dell’altro.