Relazione progetto 14/01/2019 (Traccia 2)

Scelte implementative e funzionamento dell'algoritmo:

Per la prima parte richiesta dalla traccia è stato implementato l'algoritmo **hasCycleUF,** il quale, basandosi su un iterator che scandisce tutti gli archi del grafo dato e sulla struttura dati UnionFind che utilizza un algoritmo di tipo quickFindBalanced, effettua un controllo sulla radice dei due nodi connessi dall'arco stesso, la quale, qualora risultasse identica per entrambi, verificherebbe la presenza di almeno un ciclo all'interno del grafo.

La seconda parte riguarda invece **hasCycleDFS,** la quale si basa su **hasCycleDFSsubGraph**implementato direttamente come metodo in Graph_AdjacencyList. Quest'ultima si avvale del sottografo composto da v (vertice di partenza della visita DFS) e tutti i vertici da esso raggiungibili.

Inizializzando per primo il vertice v come visitato, ricorre su tutti i vertici ad esso adiacenti: se uno di questi è visitato e non è genitore del vertice corrente, è presente un ciclo nel grafo. hasCycleDFS, tramite un'altra chiamata ricorsiva estende il controllo di hasCycleDFSsubGraph a tutti i possibili nodi dai quali far partire una visita DFS.

Risultati sperimentali:

I dati presenti in forma grafica e tabulare sono stati raccolti utilizzando un'apposita funzione indicata con time_it, che prende in input una funzione qualsiasi e ne ritorna il tempo di esecuzione.

Le caratteristiche della macchina utilizzata sono invece le seguenti:

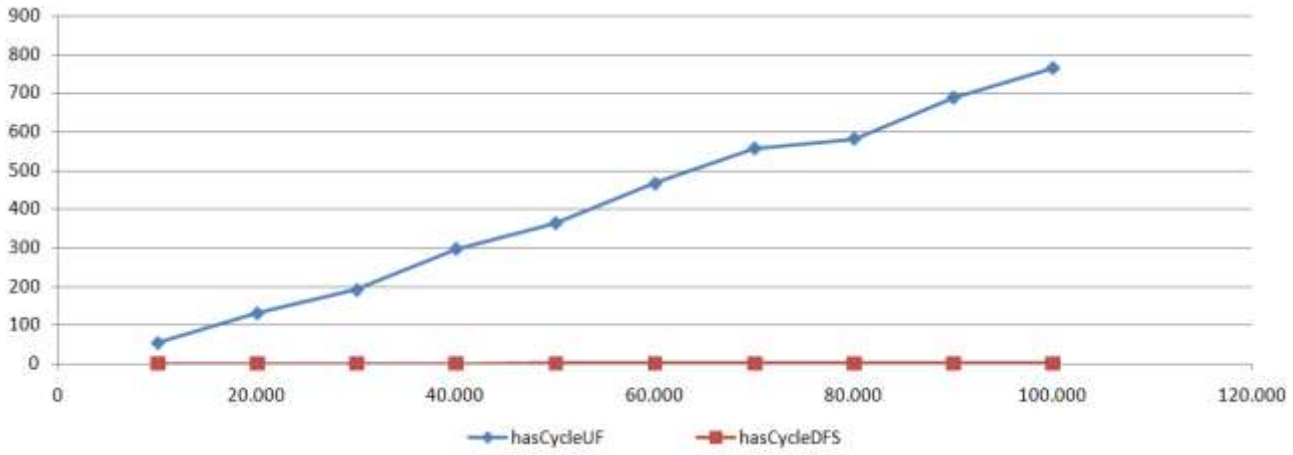
Processore:	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
Memoria installata (RAM):	16,0 GB (15,9 GB utilizzabile)

Segue quindi un confronto tra i tempi di esecuzione di hasCycleUF e hasCycleDFS in presenza di un ciclo e non.

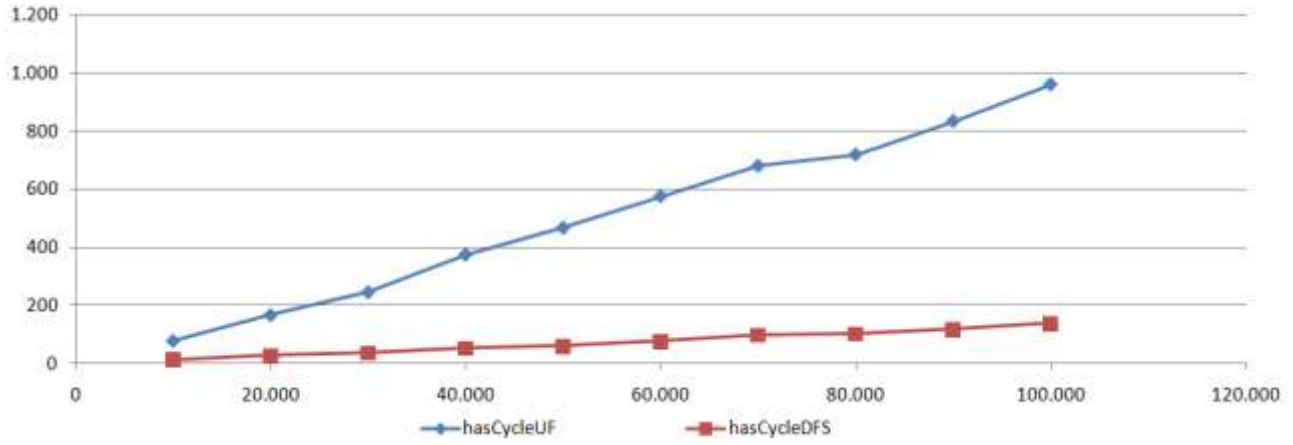
	Cycle = False	
Edges	hasCycleUF (milsec)	hasCycleDFS (mil sec)
10.000	79,3341097	10,91194153
20.000	167,6468849	25,3098011
30.000	246,4680672	34,22355652
40.000	376,0242462	50,63271523
50.000	468,7161446	58,03227425
60.000	577,3439407	74,89657402
70.000	681,9992065	97,2173214
80.000	720,6885815	101,183176
90.000	836,281538	115,5428886
100.000	962,2392654	137,8867626

	Cycle = True	
Edges	hasCycleUF (mil sec)	hasCycleDFS (mil sec)
10.000	53,56693268	0.0
20.000	130,4621696	0.0
30.000	190,9799576	0.0
40.000	297,1045971	0.0
50.000	364,0577793	0,481367111
60.000	466,2384987	0,495910645
70.000	557,4915409	0,495910645
80.000	582,2992325	0,495910645
90.000	688,9433861	0,495910645
100.000	765,3260231	0,495672226

Cycle = True



Cycle = False



Commento dei risultati sperimentali:

I risultati ottenuti mostrano come l'algoritmo che si basa su una visita DFS impieghi meno tempo di quello basato su UnionFind; ovviamente il tempo di esecuzione è proporzionale all'input dato, che rimane comunque sempre sotto al secondo anche per 100.000 archi. La presenza di un ciclo fa si che l'algoritmo DFS arresti la ricorsione subito dopo averlo trovato mentre quello UnionFind scandisce comunque tutti gli archi, risultando molto inefficiente anche in questo caso.