Seconda prova pratica in itinere

Ingegneria degli Algoritmi 2018/2019

Ovidiu Daniel Barba
ovidiudaniel.barba@alumni.uniroma2.eu

Luca Pepè Sciarria
luca.pepesciarria@alumni.uniroma2.eu

12 Dicembre 2018

1 Modalità di svolgimento

Il progetto può essere svolto singolarmento oppure in gruppi da massimo 3 persone (consigliato gruppo di 2 persone).

È altamente **sconsigliato** di copiare codice reperito on-line o da altri elaborati e di usare librerie esterne, poichè provoca l'annullamento della consegna con conseguente esclusione dalle prove in itinere.
È invece **consigliato** di usare e manipolare il codice visto a lezione e disponibile sul sito della parte pratica del corso.

2 Scelta del progetto

Una volta individuato il gruppo, definiamo s come la somma delle matricole dei componenti del gruppo. Il progetto assegnato è dato dal valore di $s \, mod \, 2 + 1$. Per esempio, se il gruppo è composta da 2 persone con matricola 0244962 e 0246425, $s = 244962 + 246425 = 491387$ e il progetto assegnato è il secondo poichè $s \, mod \, 2 + 1 = 491387 \, mod \, 2 + 1 = 1 + 1 = 2$.

Una volta individuato il progetto da svolgere, compilare il form (una sola volta per gruppo) **solo se** il gruppo del primo esonero è cambiato, altrimenti rimane la composizione precedente del gruppo.
Il form va compilato entro il **31 Dicembre 2018**.

3 Materiale da consegnare

- Breve relazione in formato pdf contenente:
  - scelte implementative
  - risultati sperimentali sia in forma tabulare che grafica con unità di misura appropriate
  - commento dei risultati ottenuti
- Codice sorgente:
  - implementazione richiesta nella traccia
  - esempio di uso dell'algoritmo o struttura dati implementata
  - esperimenti effettuati

4 Consegna

Tutto il materiale richiesto dovrà essere caricato su un repository online (ad esempio GitHub, Dropbox, Google Drive, ecc), il cui link dovrà essere inviato al tutor di riferimento del progetto (Progetto 1 - Pepè Sciarria, Progetto 2 - Barba) entro il **14 Gennaio 2019**.

5 Progetti

5.1 Progetto 1 [*Pepè Sciarria*]

Dato un grafo G non orientato, connesso e pesato sui vertici con pesi strettamente positivi, implementare l'algoritmo `visitaInPriorità` basato su *visitaGenerica* in cui l'insieme F è una coda con priorità. La priorità di un vertice è il suo peso e al valore più alto corrisponde priorità più alta. La coda F viene inizializzata con il vertice di peso maggiore e ne verrà estratto sempre il vertice con priorità maggiore. Modificare la struttura dati Grafo in modo da gestire il peso dei vertici e studiare in modo sperimentale `visitaInPriorità` confrontando l'andamento dell'algoritmo utilizzando diverse code con priorità al variare della grandezza dell'input (numero di vertici e/o archi maggiori). Effettuare dei grafici al variare del numero di archi e al variare del numero di vertici. Implementare inoltre un algoritmo per la generazione di un grafo G come descritto precedentemente.
`N.B.`: non è richiesta l'implementazione di una classe Albero in cui memorizzare il risultato della visita ma può essere usato una lista in cui inserire in ordine i nodi visitati.

5.2 Progetto 2 [*Barba*]

Algorithm 1 Checks whether a Graph has a cycle

```
procedure hasCycleUF(Graph G) → bool
    uf ← UnionFind
    for all unvisited (u, v) ∈ E(G) do
        if uf.find(u) == uf.find(v) then cycle detected
        else uf.union(u, v)
    return no cycle present
```

Dato un grafo G connesso, non orientato e non pesato, implementare due algoritmi per determinare se G ha almeno un ciclo:

- `hasCycleUF(G)` che usa la struttura dati UnionFind come descritto in **Algorithm 1** e un *iterator* per scandire tutti gli edge del grafo
- `hasCycleDFS(G)` che usa l'approccio della visita in profondità (*Depth-First Search*)

Dopo aver scelto l'implementazione di *UnionFind* (QuickFind, QuickUnion, ecc) che minimizza il tempo di esecuzione di *hasCycleUF(G)*, confrontare il tempo di esecuzione dei due algoritmi al variare della dimensione del grafo di input. I dati degli esperimenti devono essere scritti su un file da un *decorator* di Python (applicato alle funzioni dei due algoritmi) 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. Inoltre implementare un algoritmo che genera grafi con o senza cicli, da usare nella fase di testing.