

Prova d'esame del 27/06/2019 – Turno A

Si consideri il data-set "denver_crime" che riporta una serie di oltre 300k eventi criminosi tra il 2014 ed il 2017 nella città di Denver (Colorado, USA). Il data-set contiene un'unica tabella 'events' (vedi la figura nella pagina successiva), ed è tratto dell'originale db costantemente aggiornato e disponibile al sito web <https://www.denvergov.org/opendata/dataset/city-and-county-of-denver-crime>.

Si ricorda che ogni evento è caratterizzato da una *categoria* di reato (*offense_category_id*), più generale, e da un *tipo* di reato (*offense_type_id*), più specifico. Ciascuna categoria di reato è quindi divisa in uno o più tipi di reato, specifici di quella categoria.

Si intende costruire un'applicazione JavaFX che permetta di interrogare tale base dati e ricavare informazioni utili alla distribuzione delle forze di polizia. L'applicazione dovrà svolgere le seguenti funzioni:

PUNTO 1

- Si permetta all'utente di selezionare dal menu a tendina una categoria di reato, tra quelle presenti nella colonna 'offense_category_id' ed un anno (estratto da 'reported_date') tra quelli presenti nel database.
- Alla pressione del bottone 'Analisi Distretti' si costruisca un grafo semplice pesato non orientato, i cui vertici corrispondano ai tipi di reato (colonna 'offense_type_id') corrispondenti ai criteri selezionati (si ignorino quindi tutti gli eventi che non appartengono alla categoria né all'anno specificati).
- Gli archi che collegano ciascuna coppia vertici (tipi di reato, diversi tra loro) hanno un peso pari al numero di distretti distinti (*district_id*) in cui si siano verificati entrambi i tipi di reato. Qualora tale numero sia pari a zero, l'arco non deve essere creato.
- Si visualizzi l'elenco di tutti gli archi il cui peso sia pari al peso massimo presente nel grafo. Per ogni arco si visualizzino i due tipi di reato (i due vertici) ed il peso stesso.

PUNTO 2

- A partire dal grafo calcolato nel punto precedente, si permetta all'utente di selezionare dalla tendina uno degli archi sopra mostrati, identificando così i due vertici adiacenti riferiti a tale arco.
- Alla pressione del bottone 'calcola percorso' si calcoli e visualizzi un cammino aciclico semplice, che inizi e termini nei due vertici selezionati, e che tocchi tutti i vertici ed abbia peso minimo (tra i percorsi che toccano tutti i vertici).

Nella realizzazione del codice, si lavori a partire dalle classi (Bean e DAO, FXML) e dal database contenuti nel progetto di base. È ovviamente permesso aggiungere o modificare classi e metodi.

Tutti i possibili errori di immissione, validazione dati, accesso al database, ed algoritmici devono essere gestiti, non sono ammesse eccezioni generate dal programma.

events	
🔑	incident_id: BIGINT
📄	offense_code: INTEGER
📄	offense_code_extension: INTEGER
📄	offense_type_id: VARCHAR(30)
📄	offense_category_id: VARCHAR(28)
📄	reported_date: DATETIME
📄	incident_address: VARCHAR(97)
📄	geo_lon: DOUBLE
📄	geo_lat: DOUBLE
📄	district_id: INTEGER
📄	precinct_id: INTEGER
📄	neighborhood_id: VARCHAR(26)
📄	is_crime: INTEGER
📄	is_traffic: INTEGER