Università degli studi di Bologna a.a. 2019/20

Informatica - Algoritmi e strutture dati

Filippo Peterlini, filippo.peterlini@studio.unibo.it

AlgaT - GreedyIsTheWay

Struttura

```
├── README.md
├── build.sh
├── run.sh
├── pack.sh
├── build
├── javafx-sdk-12.0.1
└── src
    ├── Launcher.java
    ├── AlgatApp.java
    ├── RealBackpackController.java
    ├── KruskalController.java
    ├── JFXGraphLink.java
    ├── MFSet.java
    ├── QuestionsController.java
    ├── HomeView.fxml
    ├── RealBackpackView.fxml
    ├── KruskalView.fxml
    ├── img
    └── questions
        ├── kruskal
        └── realbackpack
```

Dipendenze

JDK 12.0.2 at jdk.java.net/12/
Java-FX 12.0.1 at openjfx.io

Introduzione

Il progetto è incentrato sulla tecnica di programmazione “greedy”, mostra l’ esecuzione di due algoritmi relativi alle lezioni sull’argomento presentandoli prima con una breve descrizione. Il primo algoritmo risolve il problema per zaino reale, variante dello zaino intero (zaino 0-1), il secondo riguarda invece l’algoritmo di Kruskal per trovare il minimo albero di copertura in un grafo pesato connesso non orientato. Infine sono proposte 5 domande per lezione a risposta multipla.

Sorgenti

Views

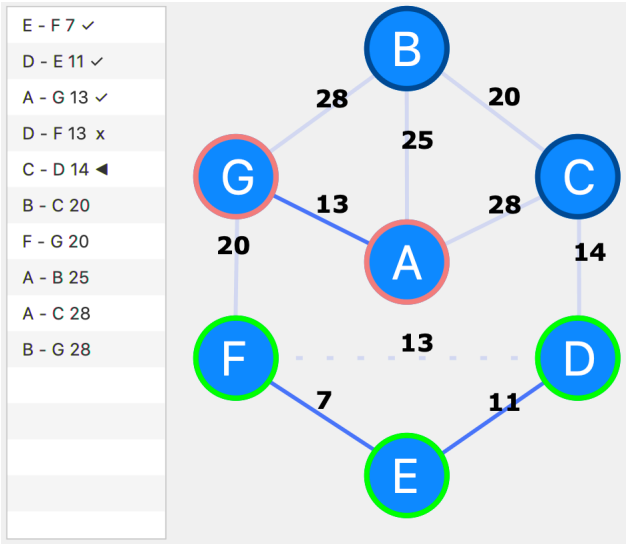
Le varie viste sono state costruite tramite SceneBuilder, comprendono:

Home View, la prima vista ad essere presentata quando l’applicazione viene eseguita, è suddivisa in 4 pagine:

- Introduzione, descrive brevemente l’argomento.
- Problema dello zaino reale, presenta la lezione riguardante il problema dello zaino reale e l’algoritmo greedy per risolverlo, da qui è possibile avviare il tutorial che mostra la sua esecuzione.
- Algoritmo di Kruskal, presenta la lezione sull’algoritmo di Kruskal per la ricerca del cammino minimo in un grafo, da qui è possibile avviare il tutorial che mostra la sua esecuzione.
- Domande, comprende due pannelli di tipo Pagination dove vengono caricate le domande sulle lezioni presentate nell’applicazione.

Real Backpack View, la vista è divisa in due, la parte superiore rappresenta l'insieme di partenza con i vari oggetti che vengono ordinati e scelti dall'algoritmo, è possibile modificare il profitto ed il volume di ogni oggetto, in fondo ad ognuno di essi è indicato il relativo profitto specifico. La parte inferiore rappresenta la zaino dove vengono caricati i vari oggetti dopo essere stati ordinati, in fondo ad ognuno di essi viene indicata la frazione dell'oggetto inserito.

Kruskal View, la vista mostra un grafo costruito attraverso tre pannelli sovrapposti, quello superiore contiene i 7 nodi del grafo; sotto si trova il pannello che contiene i vari pesi degli archi; l'ultimo contiene le linee che collegano i nodi per formare gli archi del grafo. A sinistra del grafo è presente una lista che contiene gli archi con relativo peso, a fianco viene indicato attraverso una freccia su quale di questi l'algoritmo è fermo, altrimenti viene visualizzato ✓ se l'arco è stato inserito nell'albero di copertura o X in caso contrario.



Questions Views, le domande sono implementate nell'applicazione attraverso viste, identificate da un numero progressivo, nella directory *questions* separate per le due lezioni nelle rispettive sotto-directory *kruskal* e *realbackpack*. Ad ogni domanda corrisponde una vista, così facendo è possibile aggiungere nuove domande e personalizzare ciascuna in base alle possibili risposte, aggiungendo immagini e fornire esempi/spiegazioni. In ogni vista esiste una label nascosta che riporta nel campo testo la risposta corretta alla domanda.

Controllers

AlgatApp [package: algat], la classe eredita da Application di JavaFX, implementando i metodi necessari *main* e *start*, quest' ultimo carica e visualizza HomeView. Il metodo *initialize* associa attraverso *getQuestionView* ogni pagina del pannello Pagination una domanda relativa alla lezione (e.g. pagina con index 0, lezione Kruskal, associa la vista /questions/kruskal/1.fxml). I metodi *initRealBackpack* e *initKruskal*, che sono associati ai pulsanti "Tutorial" nei rispettivi tab in HomeView per le due lezioni, caricano rispettivamente le viste RealBackpackView e KruskalView visualizzandole in una nuova finestra.

RealBackpackCntroller [package: algat.controller], il controller gestisce la lezione sullo zaino reale implementando l'algoritmo greedy per risolverlo. È possibile modificare i vari profitti e volumi nei vari oggetti proposti nell' "insieme di partenza", il controller obbliga ad ordinare gli oggetti, prima di proseguire, tramite il pulsante "Ordina" collegato al metodo *handleSort*, questo controlla che i valori inseriti siano tutti numeri interi (*allNumbersCheck*) e notifica (*invalidInput*) in caso il vincolo non sia rispettato bloccando l'ordinamento; se tutti gli oggetti sono conformi il metodo calcola il loro rapporto profitto/volume e lo visualizza in fondo al relativo oggetto attraverso *setPV*, quest'ultimo metodo inoltre aggiunge in una lista di coppie <Pane, Double> chiamata *startingSet* i vari oggetti associandoli al loro "profitto specifico"; infine quando è stato calcolato il rapporto profitto/volume per tutti gli oggetti questi vengono ordinati nel pannello "Insieme di partenza", vengono disabilitati i vari input così da non compromettere l'ordinamento ed il pulsante "Prosegui" viene abilitato, questo essendo collegato metodo *handleNextStep,* permette di aggiungere il primo oggetto (o una sua frazione) allo "Zaino" eliminandolo dall' "Insieme di partenza"; ad ogni passaggio è aggiornato il

volume residuo e il profitto progressivo dello zaino attraverso il metodo *updateBackpack.* Il controller si blocca quando tutti gli elementi sono stati spostati nello “zaino”.

KruskalController [package: algat.controller], il controller gestisce la lezione sul problema del minimo albero di copertura per un grafo implementando l’algoritmo di Kruskal. Il metodo *initialize* crea per ogni linea nel pannello linksView (che contiene gli archi tra i vari nodi del grafo) un oggetto JFXGraphLink che rappresenta un arco nel grafo mantenendo il peso, scelto casualmente, una referenza all’oggetto JavaFX Line e i nodi che questo collega, inserendolo nella lista links; infine al centro di ogni linea viene aggiunto graficamente il peso dell’ arco tramite il metodo *addTextWeigth*, successivamente gli archi vengono ordinati per peso nella lista links e inseriti nella JavaFX ListView (posta a sinistra del grafo) come stringhe del tipo “nodo1 - nodo2 peso” tramite il metodo *toString* di JFXGraphLink. Il pulsante “Prosegui” permette di avanzare nell’ algoritmo di Kruskal, è associato al metodo *kruskalHandler* questo verifica che la dimensione dello spanning tree non superi n-1 dove n è il numero di nodi del grafo, se il vincolo è rispettato chiama in ordine i metodi *kruskalNextStep* e *updateGraphView*: il primo implementa le istruzioni interne al ciclo dell’ algoritmo di Kruskal, utilizzando la struttura MFSet aggiunge l’arco corrente a Set<JFXGraphLink> minSpanningTree nel caso i nodi che questo collega non abbiano un rappresentante uguale. Il secondo metodo per ogni arco inserito nello spanning tree evidenzia (aumentando la sua opacità) la linea che lo rappresenta; inoltre colora in base alle componenti connesse nella struttura MFSet il bordo dei nodi per distinguerli*;* infine *updateGraphView* aggiorna la lista a sinistra del grafo spostando la freccia all’ arco successivo e mettendo a fianco di quello corrente ✓ se è stato scelto o x se è stato scartato, questo tramite il metodo *forwardListItem*. Il controller non permette di avanzare oltre se l’albero di copertura min-

imo è stato trovato, disabilitando il pulsante “Prosegui” e mostrando il peso totale degli archi che formano l’albero.

QuestionsController [package: algat.controller], il controller gestisce le varie domande caricate dall’applicazione, quando una risposta viene selezionata il metodo *answerSelected* controlla se è quella corretta evidenziandola in verde, in caso contrario la evidenzia in rosso e mostra quella corretta. Il pulsante per passare alla prossima domanda viene abilitato solo se è stata scelta una risposta (*toggleNextButton*), quando questo viene premuto il metodo associato *toggleNextQuestion* sposta la visione alla domanda successiva cambiano l’indice corrente nel pannello JavaFX Pagination.

Utility

MFSet [package: algat.util], classe che gestisce la struttura dati merge find set utilizzata dall’algoritmo di Kruskal, implementando “compressine dei percorsi” ed “euristica sul rango”.

JFXGraphLink [package: algat.util], classe che implementa un arco di un grafo pesato, offre metodi che vengono utilizzati dal controller per la lezione sull’algoritmo di Kruskal.

Note

Il progetto è stato sviluppato senza l’ausilio di nessun IDE su sistema operativo OSX 10.14.5.

Per compilare, avviare ed archiviare (in formato Jar) l’ applicazione sono stati creati rispettivamente gli script *build*, *run* e *pack* che si trovano nella directory del progetto, per il loro funzionamento si rimanda al README del progetto.