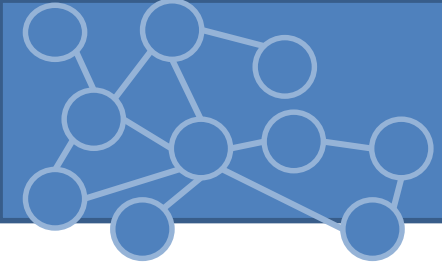
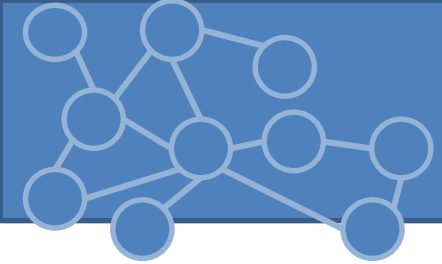


Laboratorio Reti di Calcolatori

Laurea Triennale in Comunicazione Digitale

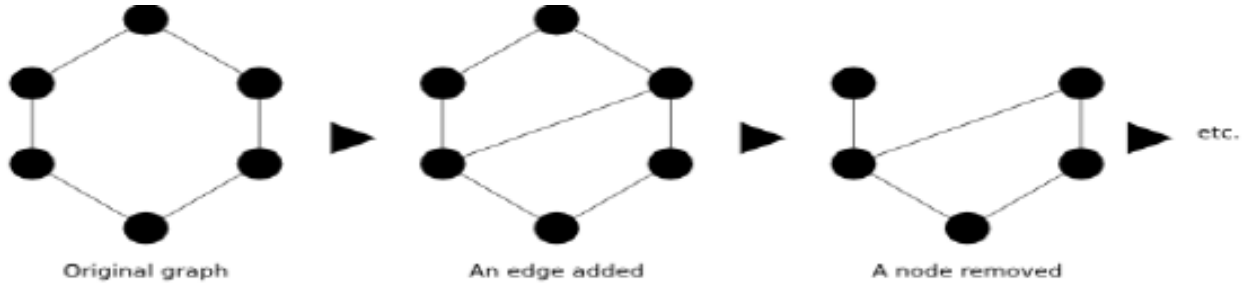
Anno Accademico 2013/2014



GraphStream

- **Graphstream** = libreria Java che gestisce grafi e aspetti dinamici sui o dei grafi => modellazione di reti di interazione di varia dimensione
- Basato su modello ad eventi
- Capacità:
  - Grafi diretti, indiretti, multigrafi
  - Qualsiasi attributo (chiave,valore) su nodi, archi e grafi
  - Evoluzione nel tempo del grafo => flusso di eventi sul grafo
  - Algoritmi e misure su grafo
  - Componenti import/export per supporto di diversi formati
- Sito del progetto:
  - http://graphstream-project.org
- Download
  - http://graphstream-project.org/download/
- Tre package:
  - Core: classi base, classi per eventi, user interface minimale (gs-core-X.Y.jar)
  - Algo: algoritmi e generatori (gs-algo-X.Y.jar)
  - UI: classi per viewer più complessi (gs-ui-X.Y.jar)
- Documentazione
  - http://graphstream-project.org/doc/
  - API, tutorial …

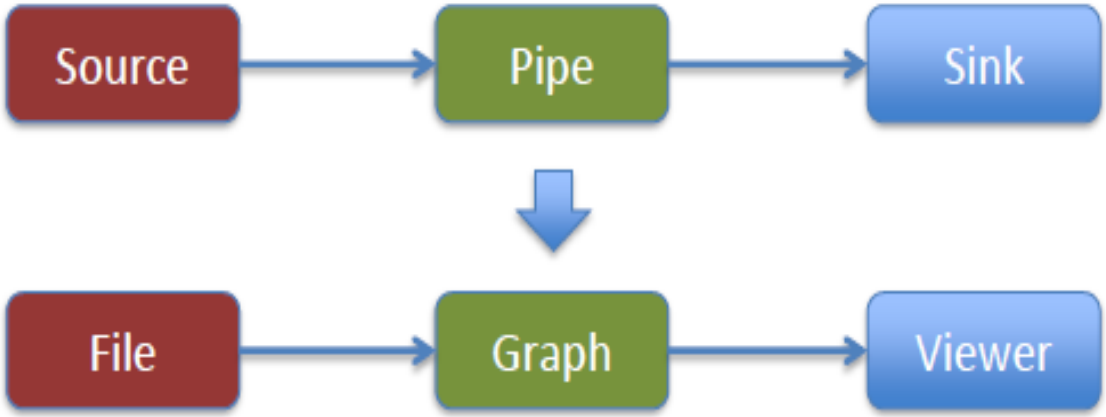
- Costruzione di un grafo espressa come un modello ad eventi
  - Evento:
    - Aggiungo, rimuovo nodi
    - Aggiungo, rimuovo archi
    - Aggiungo, rimuovo e aggiorno attributi
    - Creo degli snapshot temporali

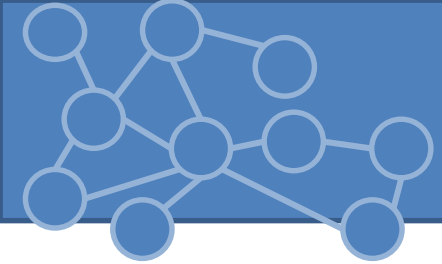


- Il **flusso di eventi** modifica la struttura del grafo
- Sources, sinks e pipes

Source, Sink e Pipe

- Sources: componenti che generano flussi di eventi (seguendo meccanismo dei listener) Source
- Sinks: componenti che ricevono e processano eventi Sink
- Pipe: componenti che ricevono eventi e ne producono (grafi sono pipe, filtri) Pipe

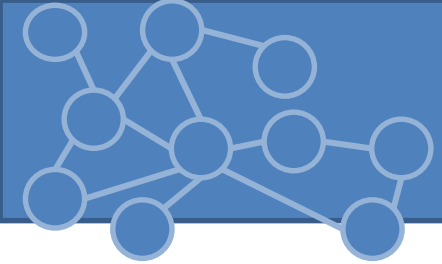




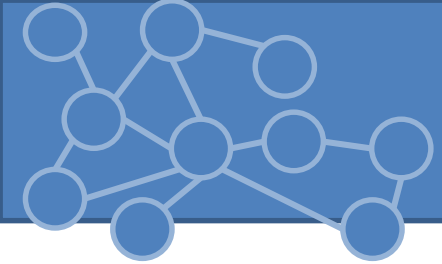
Interfaccia Graph

- Tutti i grafi implementano l'interfaccia **Graph**
  - AdjacencyListGraph, DefaultGraph, MultiGraph, SingleGraph

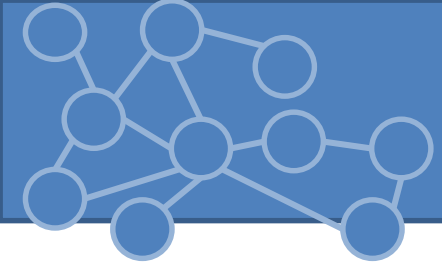
  Metodi principali
    - addEdge/Node
    - getEdge/Node
    - getEachEdge/Node
    - getEdgeIterator/Node
    - Read/write
    - removeEdge/Node
    - setAutoCreate
- **SingleGraph**: 1-graph orientato o meno, estende AdjacencyListGraph (liste di adiacenza)
- Come popolare un grafo:
  1. A mano: invoco continuamente add, remove, update
  2. Produttore di eventi: associo il grafo ad un produttore di eventi (source)
- Posso inserire solo gli archi
  - setAutoCreate()



- Ogni inserimento nodo/arco => oggetto corrispondente implementa interfacce Node e Edge
- Identificatore nodo e arco = stringa univoca
- Per ottenere riferimento dato identificatore:
  - getNode(<stringa identificatore>)
  - getEdge(<stringa identificatore>)
- Per iterare sui nodi/archi
  - getEachNode(), getNodeSet(), getNodeIterator()
  - getEachEdge(), getEdgeSet(), getEdgeIterator()

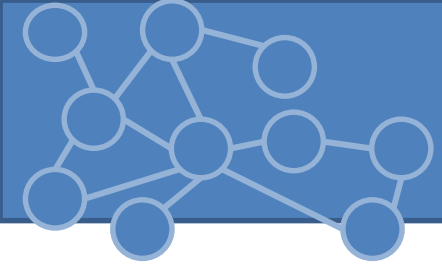


- File sources permettono di utilizzare un file come sorgente di eventi
  - Intero file
  - Step-by-step
- Posso leggere i formati: DGS (GraphStream), DOT (GraphViz), GML, TLP (Tulip), NET (Pajek), GraphML, GEXF (Gephi)
- FileSource = interfaccia che modella sorgente di eventi provenienti da file
  - readAll() per leggere il file
  - addSink(Sink) per aggiungere una sorgente ad un ascoltatore
- Possibile lasciare a libreria possibilità di associare la giusta sorgente file
  - FileSourceFactory.sourceFor()



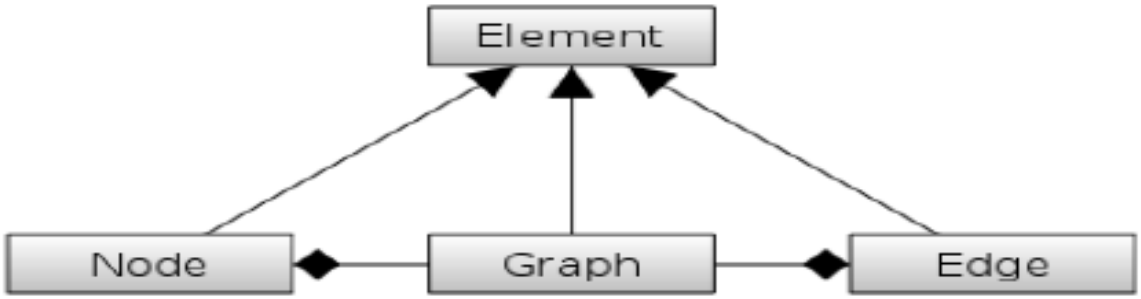
File .dgs

- File testuale
  - Ogni linea termina con separatore di line \n
  - Ogni campo separato da un carattere di spacing
- Header (due linee)
  - Versione del formato (DGS003 ultima versione)
  - Nome del grafo, #step, #eventi (ultimi due campi solo indicativi)
- Body
  - Sequenza di passi contenente eventi
  - Alcuni eventi possono avere parametri (attributi) specificabili nel formato nome(: o =)valore. Il valore può essere una stringa , un numero o un vettore (sequenza separata da ,)



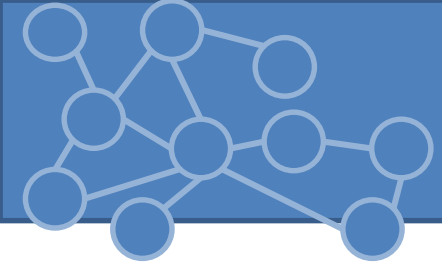
Evento	Argomenti	Descrizione
st	numero	Definisce un passo e il numero permette di identificare il passo
an	IdNodo	Inserisce un nodo con identificatore Id nodo
cn	IdNodo	Modifica le proprietà del nodo identificato da Id nodo
dn	IdNodo	Elimina il nodo identificato da Id nodo
ae	IdEdge IdNodo1 IdNodo2	Aggiuge un arco tra IdNodo1 e IdNodo2 con identificativo IdEdge. E' possibile orientare l'arco inserendo tra IdNodo1 e IdNodo2 i simboli < o > a secondo dell'orientamento
ce	IdEdge	Modifica l'arco con identificativo IdEdge
de	IdEdge	Elimina l'arco con identificativo IdEdge

- Eventi strutturali: definisce la struttura del grafo => si riferisce a nodi ed archi
- Elementi: oggetti che determinano la struttura del grafo

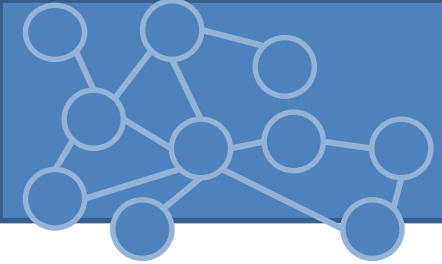


Tutte interfacce

- Interfaccia Element: metodi per memorizzare e restituire attributi



- Eventi informativi: dati memorizzati sul grafo.
- Dati associabili con un qualsiasi elemento: grafo, nodo, arco
- Dati = attributi
  - Posso assegnare qualsiasi numero
  - Ogni attributo identificato da una chiave
    - Gli attributi su ogni elemento del grafo DEVONO essere UNICI
  - Attributo: mapping chiave(string)/valore(Object)
- Interfaccia Element: metodi memorizzazione ed estrazione attributi
- Node/edge ereditarietà



- Metodo più comune per estrazione attributo

  Element.getAttrbute(String key)

  – Restituisce il valore associato alla chiave o null se non c'è alcun attributo associato

- Per memorizzare un attributo

  void Element.setAttribute(String key, Object … values)

  – Lista di argomenti variabile

  – Senza valori = l'attributo funge da boolean -> automaticamente settato a true

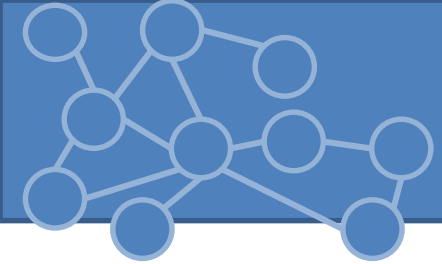
  boolean Element.hasAttribute(String key)

- Posso estrarre tutti i valori:

  Object[] array = Element.getAttribute("a lot")

- Rimuovere attributi

  Element.removeAttribute(String key)



- Rimuovere tutti gli attributi in un solo colpo

  Element.clearAttributes()
- Per farmi restituire una lista di tutte le chiavi

  Element.getEachAttributeKey

  Iterator<String> Element.getAttributeKeyIterator()
- Element.getAttribute() non restituisce Object

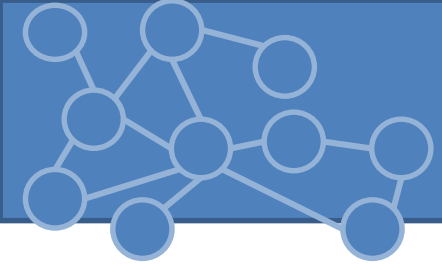
  <T> T getAttribute(String key)
  - Stesso tipo del valore dell'attributo
  - Cast senza invocare un cast => alleggerisce codice ma l'errore è dietro l'angolo
- Per assicurarsi che venga restituito un oggetto di una particolare classe

  <T> T Element.getAttribute(String key, Class<T> cls)
  - Meno errori
- Metodi per tipi di attributo comune

  getLabel() -> stringa

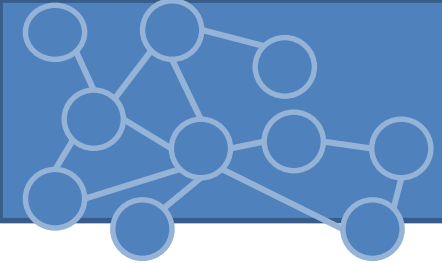
  getNumber() -> double o NaN

  setNullAttributeAreError(boolean)



Visualizzazione

- Molti viewer per GraphStream -> default nel core (piccolo e stabile)
  - Nodi come cerchi con dimensione e colore variabile
  - Arci sono segmenti con spessore e colore variabile
  - Etichetta assegnabile a nodi e archi
  - Sprites
- Viewer più avanzati
  - 2D: varie forme elementi, immagini, sfondi, frecce…
  - 3D: per grandi grafi
- Viewer 2D configurabili con CSS -> separo grafica da struttura

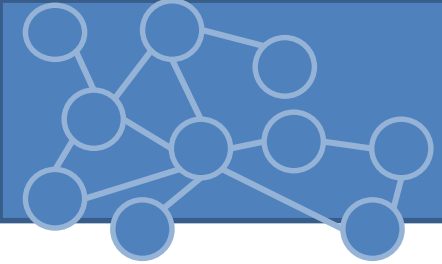


- Aggiungere etichetta

  Element.addAttribute("ui.label", "A")
- Colori modificabili con fogli di stile => specifico che appartengono ad una classe
- Aggiungere un foglio di stile

  Element.addAttribute("ui.stylesheet", string)
  - String può essere le specifica o nome file o URL
- CSS usa selettori per associare una stile ad un elemento del grafo
  - Selettore: node, edge, graph

  Lo stile è nella forma <selettore>{} contenente coppie proprietà:valore;
    - Proprietà = parola riservata che indica quale aspetto viene interessato da cambiamento

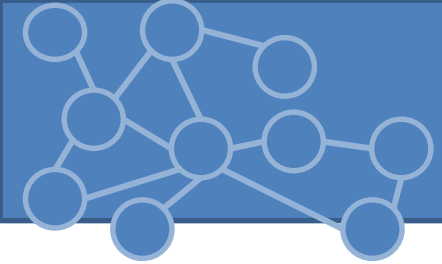


Per definire una classe

```
node {
        fill-color: black;
}
node.marked {
        fill-color: red;
}
```

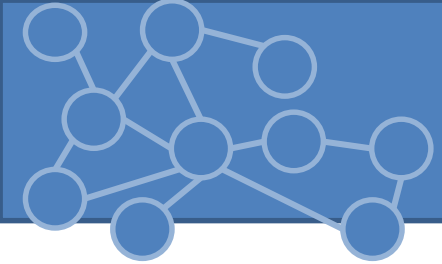
- Di default element non appartengono ad alcuna classe di stile. Per specificare classe

```
n.setAttribute("ui.class", "marked");
```



Algoritmo di Dijkstra

- Calcola il cammino minimo da un nodo verso i rimanenti => albero dei cammini
- Solo pesi non negativi
- Complessità: O(n logn + m)
- parametro lengthAttribute
- Metodi:
  - getPathLength(Node)
  - getPathEdges
  - getTreeEdges
  - getAllPathsIterator



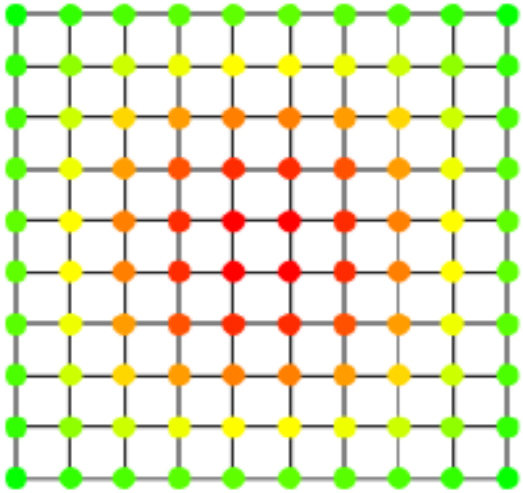
Algoritmo di Johnson

- Calcolo cammino minimo per ogni coppia di nodi
- Complessità: O(n^3)
- Implementazione con due classi
- Posso farmi restituire il path usando classe ASPSInfo, oggetto memorizzato in ogni nodo.

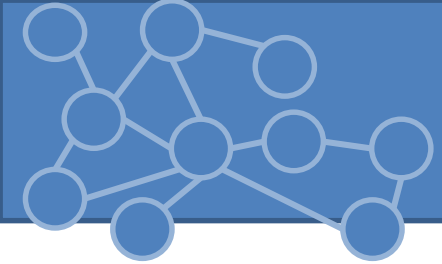
```
APSPInfo info =
graph.getNode("F").getAttribute(APSPInfo.ATTRIBUTE_NAME)

info.getShortestPathTo("A")
```

Betweenness Centrality

Calcola quanti cammini minimi passano per un nodo



- setWeighted(), setUnweighted(), setWeightAttributeName(String)
- Complessità: O(mn) nel caso di grafo non pesato (Algoritmo di Brandes)



Il coltellino svizzero

org.graphstream.algorithm.Toolkit

- degreeDistribution(Graph)
- averageDegree(Graph)
- density(Graph)
- diameter(Graph)
- clusteringCoefficient(Node)
- clusteringCoefficients(Graph)
- averageClusteringCoefficient(Graph)
- randomNode(Graph)