



Quarto giorno: mattina

Agenda 7/8

— GraphFrames e altre librerie

Packages

Spark SQL, Spark Streaming, Spark ML, GraphFrames

Spark Streaming

Breve introduzione: ci sarà un corso dedicato

Spark ML

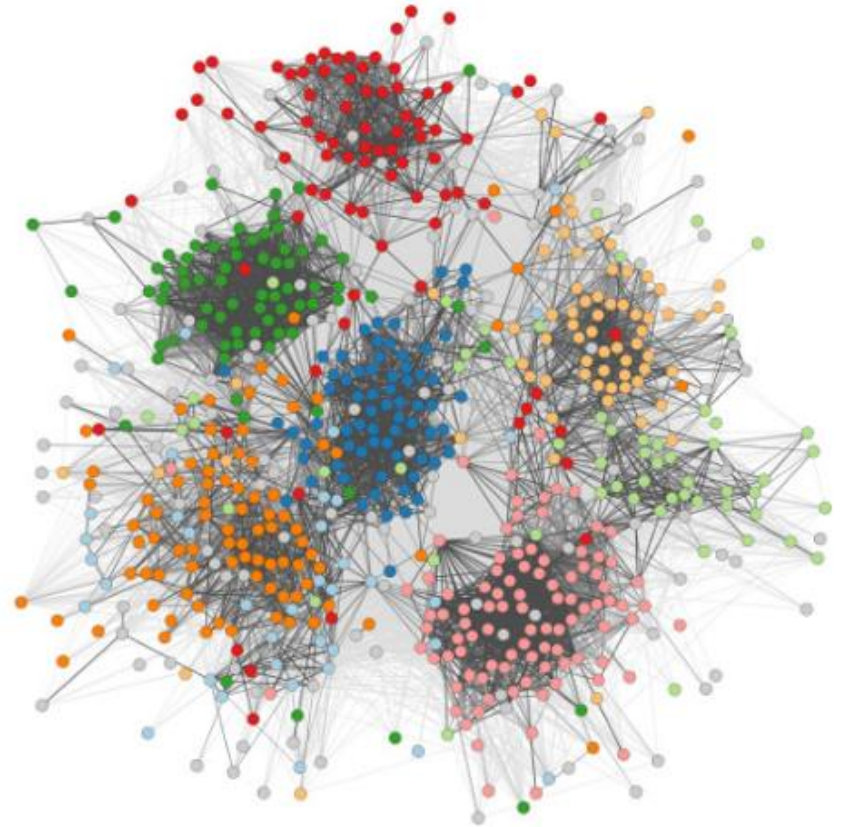
Breve introduzione: ci sarà un corso dedicato

GraphFrames

Nota: GraphFrames (basato su DataFrames; 2016)
vs. GraphX (basato su RDD; 2014)

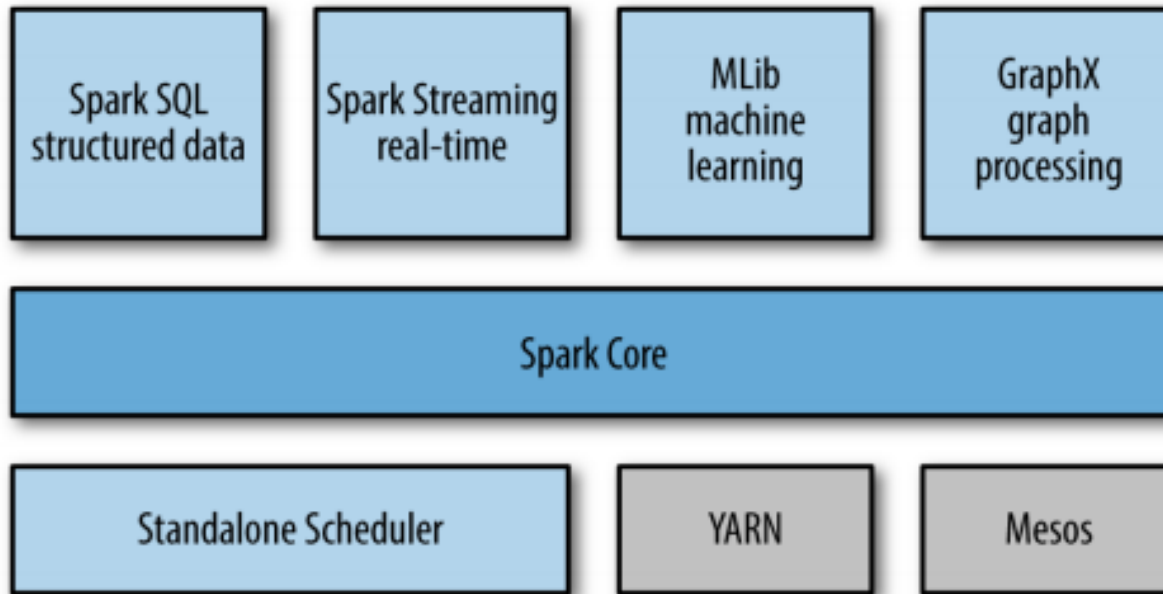
Esercizi di laboratorio

Esercizi di recapitolazione



Le librerie standard di Spark

- **Spark SQL**
- **MLib** : machine learning
- **Spark Streaming** (e il nuovo **Structured Streaming**)
- **GraphX** : analisi dei grafi.



Le librerie standard di Spark

Spark SQL

La libreria offre un'API strutturata di alto livello

Nei giorni scorsi abbiamo studiato l'astrazione basata sui *DataFrame* (definita in *org.apache.spark*)

La libreria permette di esprimere gli stessi costrutti con la sintassi SQL:

```
df.createOrReplaceTempView("myTable")
spark.sql("""
select city,richter from myTable
where richter>6.6
order by richter desc
""")
```

res24: org.apache.spark.sql.DataFrame = [city: string, richter: double]

1 Took: 2.186s, at 2019-03-14 22:05

city	richter
"erzincan"	7.2
"kutahya"	7
"bursa"	7
"van"	6.7
"cankiri"	6.7
"canakkale"	6.7
"karabuk"	6.7
"tekirdag"	6.7

Le librerie standard di Spark

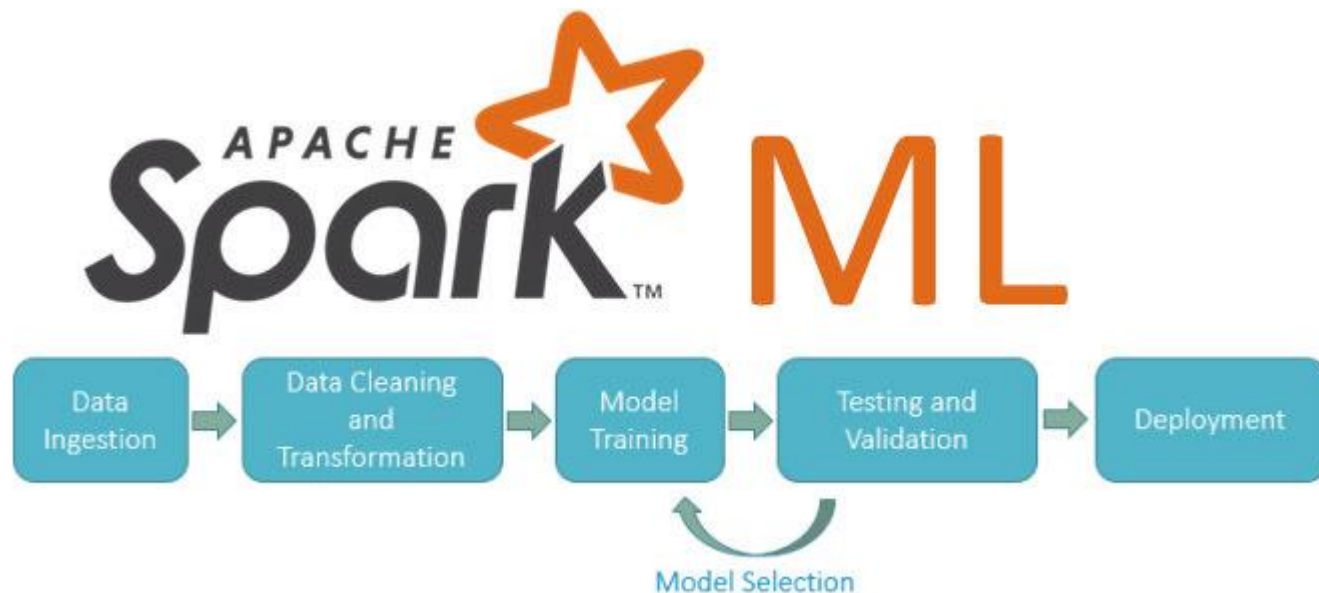
MLlib

Libreria per il machine learning

In realtà ci sono due librerie:

- ***Spark ML*** (recente, basata sui *DataFrame*) e
- ***Spark MLlib*** (basata su RDD)

Verrà trattata in un corso dedicato.



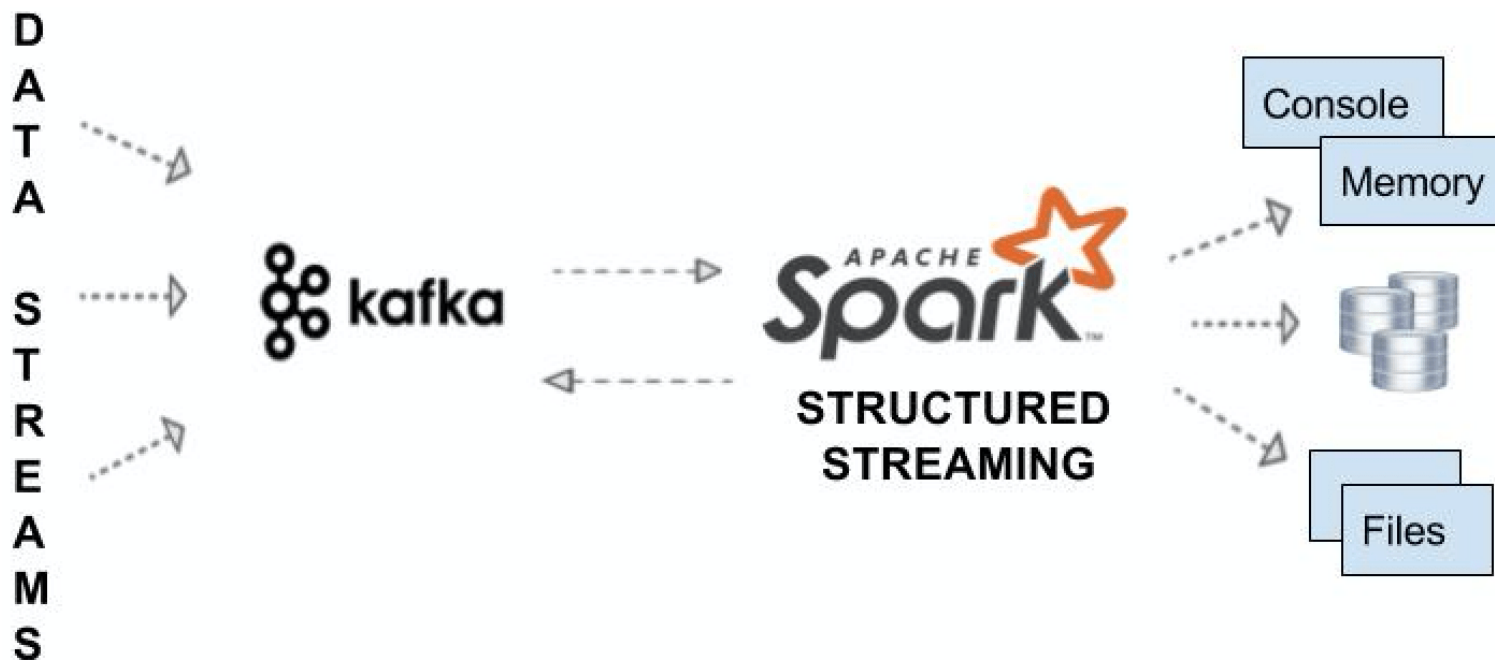
Le librerie standard di Spark

Structured Streaming

Serve a gestire i flussi di dati in (quasi) real-time.

Anche in questo caso ci sono due librerie: *Spark Streaming* e *Structured Streaming*.

Verrà trattata in un corso dedicato.



Le librerie standard di Spark

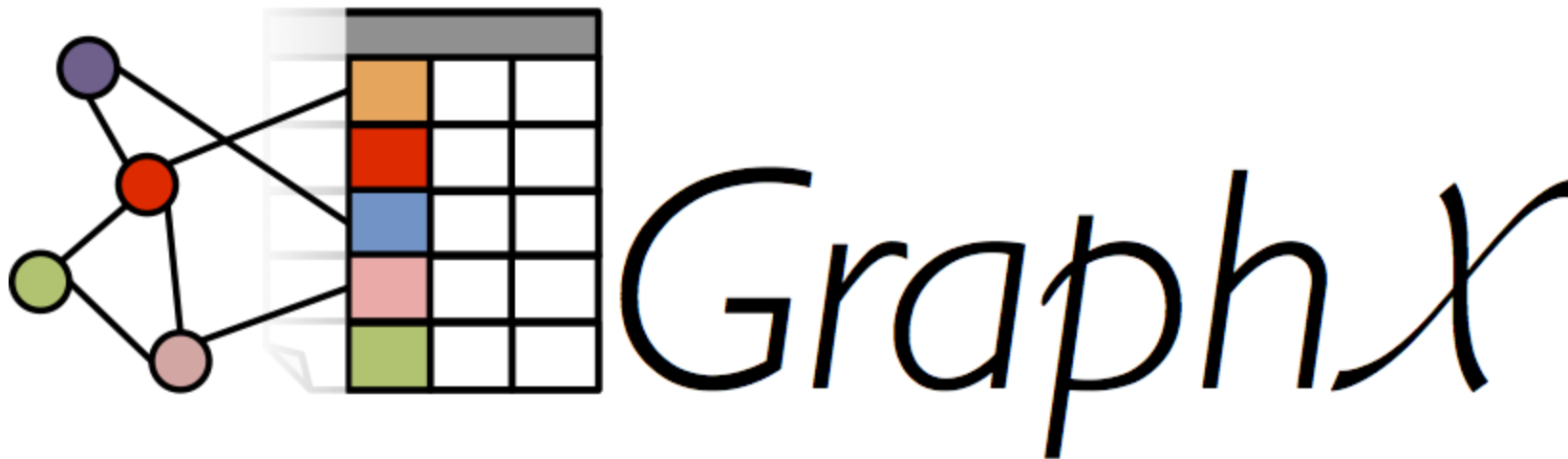
GraphX

Analisi di grafi.

Anche in questo caso ci sono due librerie:

- GraphX basata su RDD; parte delle librerie standard
- GraphFrames basata su DataFrame; package separato

Questa mattina parliamo di **GraphFrames**!



I Packages di Spark

<https://spark-packages.org/>

The screenshot shows the Spark Packages website interface. At the top, there's a navigation bar with the Spark Packages logo, links for Feedback, Register a package, Login, and Find a package (with a search icon). Below the navigation bar, a heading states "A community index of third-party packages for Apache Spark." followed by "Showing packages 1 - 50 out of 442". A horizontal menu lists various categories with their respective package counts: All (442), Core (14), Data Sources (50), Machine Learning (92), Streaming (54), Graph (20), PySpark (18), Applications (15), Deployment (12), Examples (25), and Tools (31). The first package listed is "spark-als" by @mengxr, described as "Another, hopefully better, implementation of ALS on Spark (already merged into MLlib)". It shows a latest release of 0.1.0 from 2014-11-27, a BSD 3-Clause license, a 4-star rating, and 1 user. Below it, the second package is "mllib-grid-search" by @spark-ml, described as "An example project for doing grid search in MLlib". It shows a latest release of 0.0.1 from 2014-11-27, a BSD 3-Clause license, a 4-star rating, and 2 users. At the bottom, a disclaimer states: "Spark Packages is a community site hosting modules that are not part of Apache Spark. Your use of and access to this site is subject to the terms of use."

I Packages di Spark

Fra gli altri:

- **GraphFrames** (miglioramento rispetto a *GraphX*)
- Diversi packages di *machine learning* e *deep learning*
- Connettori per vari *datasources* (*Cassandra*, *Redshift*, *Avro*, ecc.) e motori di ricerca (*ElasticSearch*, *Google BigQuery*, ecc.)
- E in genere più di 400 packages fra cui scegliere

Per usare un package:

- Specificare la dipendenza nel file di build
- Fare download dei file *JAR* precompilati e includerli nel class path
- Includere il package a runtime aggiungendo un parametro ai comandi *spark-shell* o *spark-submit*

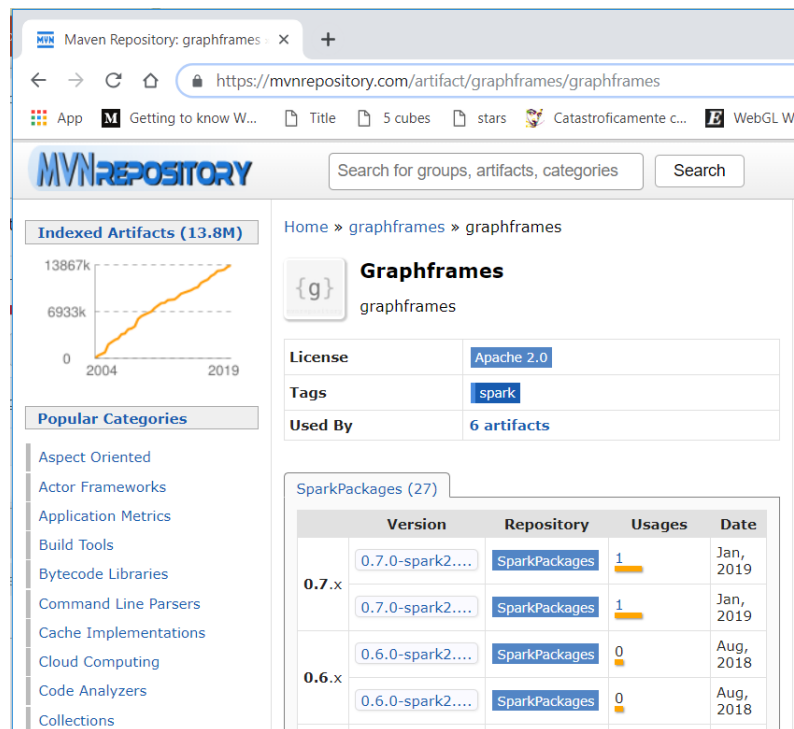
Per usare GraphFrames

Con *spark-shell* / *spark-submit*:

```
$ spark-shell \  
  --packages graphframes:graphframes:0.7.0-spark2.4-s_2.11
```

Per trovare le versioni:
Maven Repository

<https://mvnrepository.com/artifact/graphframes/graphframes>



The screenshot shows the Maven Repository page for the artifact `graphframes:graphframes`. The page includes a search bar, a sidebar with popular categories, and a main content area with artifact details and a table of SparkPackages.

Artifact Details:

- License: Apache 2.0
- Tags: spark
- Used By: 6 artifacts

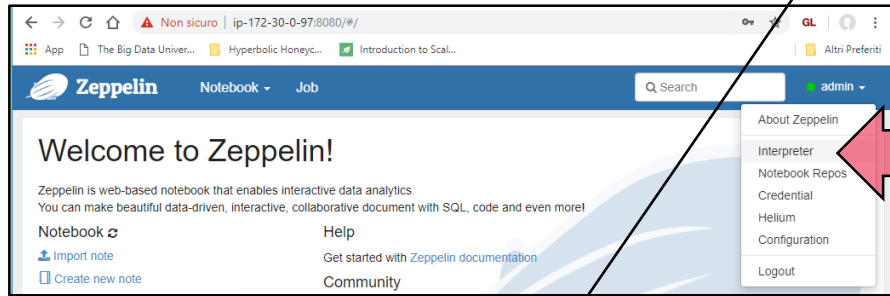
SparkPackages (27):

	Version	Repository	Usages	Date
0.7.x	0.7.0-spark2....	SparkPackages	1	Jan, 2019
	0.7.0-spark2....	SparkPackages	1	Jan, 2019
0.6.x	0.6.0-spark2....	SparkPackages	0	Aug, 2018
	0.6.0-spark2....	SparkPackages	0	Aug, 2018

Per usare GraphFrames

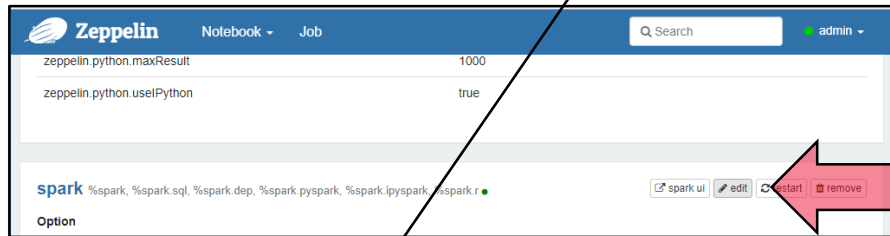
Con Zeppelin Notebook:

graphframes:graphframes:0.7.0-spark2.4-s_2.11



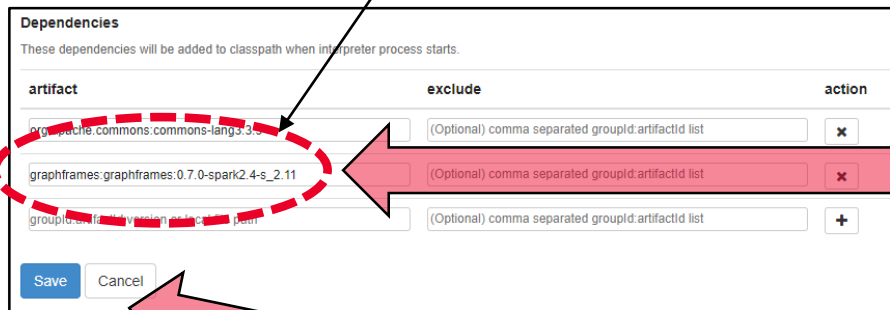
Interpreter

1



Edit

2

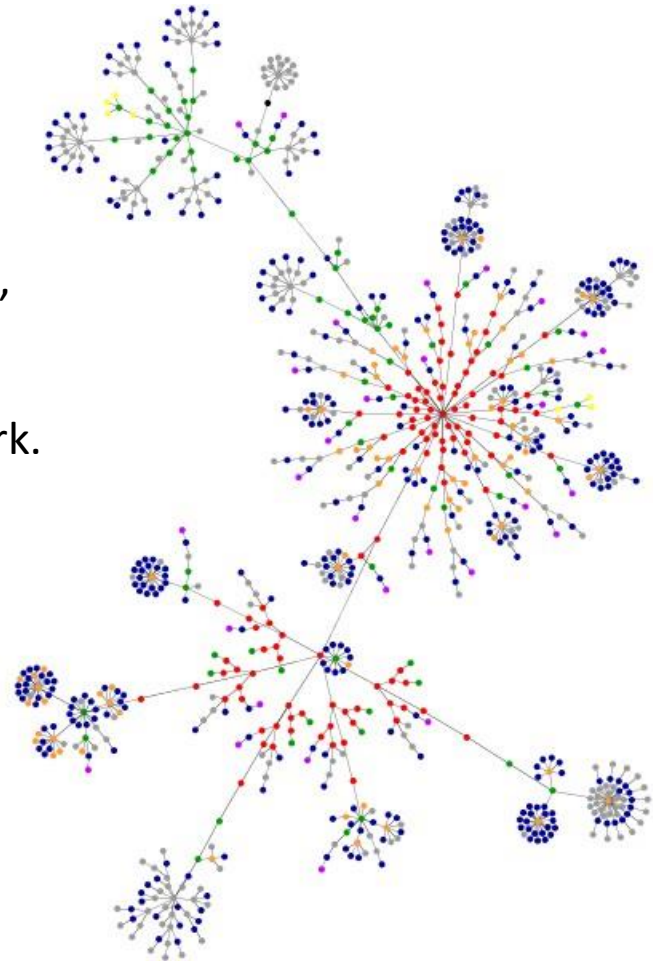


3

4

GraphFrames

- Un **Grafo** è un insieme di **Nodi** (o **Vertici**) che possono essere collegati fra loro da linee chiamate **Archi**.
- Gli archi possono essere orientati (grafo diretto) oppure non orientati.
- È una struttura matematica che rappresenta relazioni fra entità. Ad esempio legami fra persone, società, ecc.
- **GraphFrames** permette di analizzare grafi con Spark.



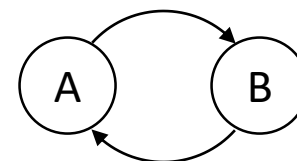
Creare un grafo

```
import org.graphframes.GraphFrame  
  
val graph = GraphFrame(vertices, edges).cache()
```

GraphFrames usa una convenzione per i nomi:

- Il *DataFrame* che contiene i vertici deve avere una colonna «**id**»
- Il *DataFrame* che contiene gli archi deve avere due colonne «**src**» e «**dst**»

La convenzione implica un grafo diretto. Per implementare un grafo non diretto bisogna definire due archi opposti per ogni coppia di nodi direttamente connessi



Creare un grafo

station_data.csv

station_id	name	lat	long	dockcount	landmark	installation
2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose	8/6/2013
3	San Jose Civic Center	37.330698	-121.888979	15	San Jose	8/5/2013
4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose	8/6/2013

trip_data.csv

Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
913460	765	8/31/2015 23:26	Harry Bridges Plaza (Ferry Building)	50	8/31/2015 23:39	San Francisco Caltrain (Townsend at 4th)	70	288	Subscriber	2139
913459	1036	8/31/2015 23:11	San Antonio Shopping Center	31	8/31/2015 23:28	Mountain View City Hall	27	35	Subscriber	95032

```
import org.graphframes.GraphFrame

val vertices = spark.read.option("header",true).csv("../data/station_data.csv")
    .withColumnRenamed("station_id","id")
    .distinct()

val edges = spark.read.option("header",true).csv("../data/trip_data.csv")
    .withColumnRenamed("Start Station", "src")
    .withColumnRenamed("End Station", "dst")

val graph = GraphFrame(vertices, edges)
```

```
import org.graphframes.GraphFrame
```

Took: 1.368s, at 2019-03-15 00:29

Query più semplici

```
graph.vertices.count()
graph.edges.count()
graph.inDegrees
graph.outDegrees
graph.degrees
```

```
graph
.inDegrees
.orderBy(desc("inDegree"))
.show()
```

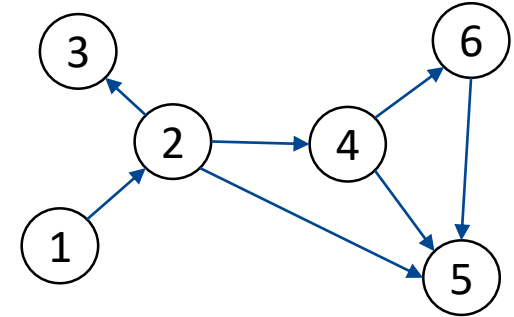
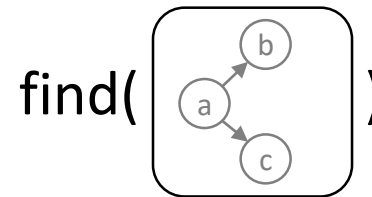
```
graph.edges
.groupBy("src", "dst").count()
.orderBy(desc("count"))
.show(10)
```

```
scala> graph.edges.
| groupBy("src", "dst").count().
| orderBy(desc("count")).
| show(10)
```

src	dst	count
San Francisco Cal...	Townsend at 7th	3748
Harry Bridges Pla...	Embarcadero at Sa...	3145
2nd at Townsend	Harry Bridges Pla...	2973
Townsend at 7th	San Francisco Cal...	2734
Harry Bridges Pla...	2nd at Townsend	2640
Embarcadero at Fo...	San Francisco Cal...	2439
Steuart at Market	2nd at Townsend	2356
Embarcadero at Sa...	Steuart at Market	2330
Townsend at 7th	San Francisco Cal...	2192
Temporary Transba...	San Francisco Cal...	2184

only showing top 10 rows

I *motif*: schemi strutturali nei grafi



```
val graph = GraphFrame(  
  Seq(1,2,3,4,5,6).toDF("id"),  
  Seq((1,2),(2,3),(2,4),(4,5),(4,6),(6,5),(2,5)).toDF("src","dst"))
```

```
val motifs = graph  
  .find("(a)-[ab]->(b); (a)-[ac]->(c)")  
  .where("b.id < c.id")  
motifs  
  .selectExpr("a.id as a","b.id as b","c.id as c")  
  .show()
```

```
+---+---+---+  
|  a|  b|  c|  
+---+---+---+  
|  2|  3|  4|  
|  4|  5|  6|  
+---+---+---+
```

```
motifs: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [a: struct<id: int>, ab: struct<src: int,  
dst: int> ... 3 more fields]
```

Took: 1.262s, at 2019-03-15 02:00

Sotto-grafi

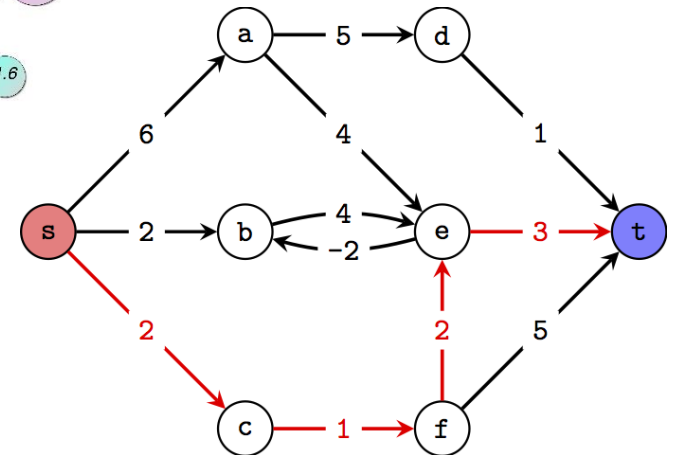
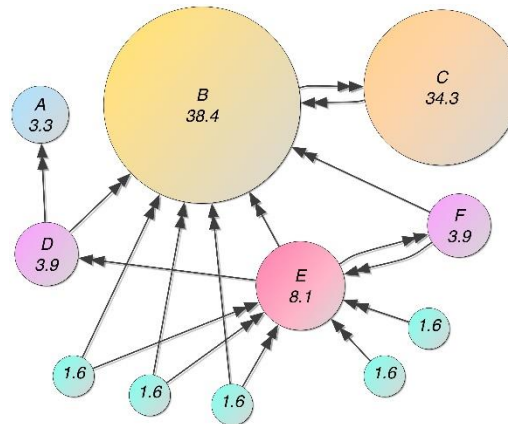
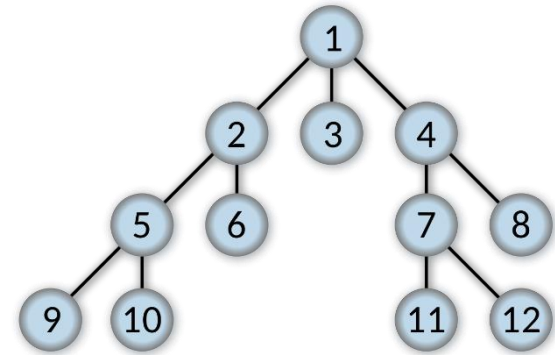
```
val g2 = g
  .filterEdges("relationship = 'friend'")
  .filterVertices("age > 30")
  .dropIsolatedVertices()
```

```
val paths = g.find("(a)-[e]->(b)")
  .filter("e.relationship = 'follow'")
  .filter("a.age < b.age")
val g3 = GraphFrame(
  g.vertices,
  paths.select("e.*"))
```

```
val g = GraphFrame(
  Seq(
    ("a", "Alice", 34),
    ("b", "Bob", 36),
    ("c", "Charlie", 30),
    ("d", "David", 29),
    ("e", "Esther", 32),
    ("f", "Fanny", 36),
    ("g", "Gabby", 60)
  ).toDF("id", "name", "age"),
  Seq(
    ("a", "b", "friend"),
    ("b", "c", "follow"),
    ("c", "b", "follow"),
    ("f", "c", "follow"),
    ("e", "f", "follow"),
    ("e", "d", "friend"),
    ("d", "a", "friend"),
    ("a", "e", "friend")
  ).toDF("src", "dst", "relationship"))
```

Algoritmi «standard» per i grafi

- Ricerca in ampiezza (*breadth-first search, BFS*)
- Ricerca componenti connesse
- Componenti fortemente connesse
- *Label propagation*
- *PageRank*
- Cammino minimo
- Conteggio dei triangoli



Algoritmi «standard» per i grafi

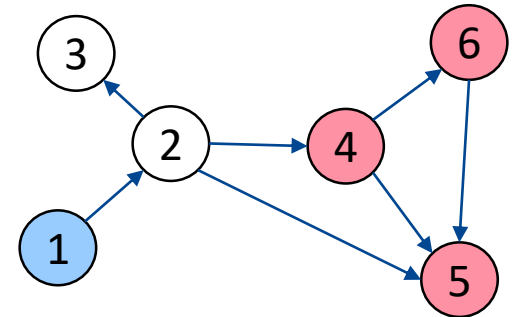
- La forma generale:

```
val result = graph.<algorithm>.<parameter>(...).run()
```

- Spesso hanno bisogno di fare dei «checkpoint»

```
spark.sparkContext.setCheckpointDir("D:\\spark\\temp\\")
```

BFS – Breadth-first search



```
val paths = graph.bfs.fromExpr("id = 1").toExpr("id >= 4").run()
paths.show()
```

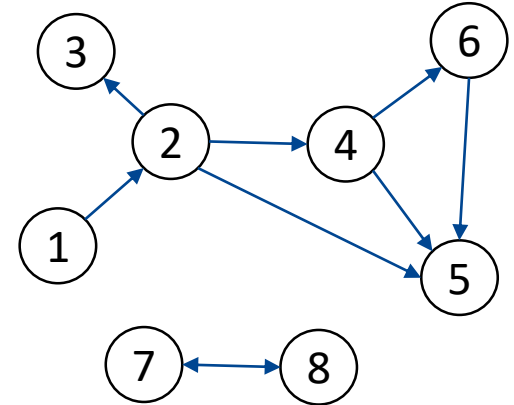
```
+---+---+---+---+---+
|from|  e0| v1|  e1| to|
+---+---+---+---+---+
| [1]| [1,2]| [2]| [2,5]| [5]|
| [1]| [1,2]| [2]| [2,4]| [4]|
+---+---+---+---+---+
```

```
paths: org.apache.spark.sql.DataFrame = [from: struct<id: int>, e0: struct<src: int, dst: int> ... 3 more
fields]
```

Took: 1.818s, at 2019-03-15 08:52

Label propagation

- Serve ad individuare le comunità
- Relativamente poco costoso, ma non garantisce risultati



```
graph.labelPropagation.maxIter(10).run()
res29: org.apache.spark.sql.DataFrame = [id: int, label: bigint]
```

1

id	label
8	8
1	1
2	2
3	1
4	1
5	1
6	2
7	7

Took: 21.028s, at 2019-03-15 09:21

