Wolfram Mathematica per la teoria dei grafi

Un'introduzione all'uso di Mathematica

Marco Liverani

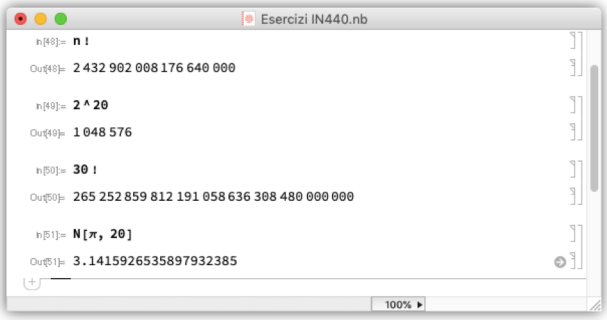
Corso IN440 – Ottimizzazione Combinatoria

http://www.mat.uniroma3.it/users/liverani/IN440/

Wolfram Mathematica

- È un **software di calcolo numerico e simbolico**, dotato di una gamma molto vasta di librerie per quasi ogni branca della Matematica, tra cui anche la **matematica discreta**, la **teoria dei grafi** e la **combinatoria su insiemi finiti**
- Il linguaggio di Mathematica adotta il paradigma dei **linguaggi di programmazione funzionali**
- È piuttosto costoso, ma è disponibile una **convenzione tra Wolfram e l'università Roma Tre** per consentire agli studenti di acquisire una licenza gratuitamente:

http://asi.uniroma3.it/page.php?page=convenzio4



Introduzione

- Mathematica presenta un **prompt** e un'**interfaccia testuale** con cui impartire comandi al sistema; i comandi devono essere completati con la sequenza **Shift + Enter**
- L'output dei comandi viene visualizzato nella stessa finestra di input, come in una lavagna elettronica interattiva

```
In[1]:= 5+4
Out[1] = 9
In[2]:= 4+3;
In[3]:= %
Out[3] = 7
```

- I calcoli vengono rappresentati in forma simbolica, a meno di non richiedere esplicitamente il calcolo di un risultato numerico (eventualmente approssimato)

```
In[1]:= 6/4
Out[1] = 3/2
In[2]:= 6./4
Out[2] = 1.5
In[3]:= N[1/3]
Out[3] = 0.333333
In[4]:= N[1/3, 10]
Out[4] = 0.3333333333
```

Help on-line e funzioni

- È disponibile un ***help on-line*** con le istruzioni per l'uso dei comandi del linguaggio; è possibile richiamare l'help per una funzione specifica con l'operatore «**?**»

```
In[1]:= ?Sqrt
Out[1] = Sqrt[z] gives the square root of z. More...
```

- Gli **argomenti delle funzioni** si indicano tra parentesi quadrate; le funzioni hanno tutte l'**iniziale maiuscola**:

```
In[1]:= Cos[π]
Out[1] = -1
```

- L'operatore «**=**» consente di **assegnare un valore ad una variabile**; l'operatore «**:=**» consente invece di **definire una funzione**

```
In[2]:= y = x
Out[2] = 10
In[3]:= z:=x
In[4]:= x = 2x
Out[4] = 20
In[5]:= y
Out[5] = 10
In[6]:= z
Out[6] = 20
```

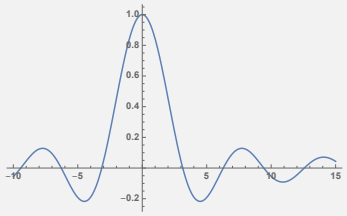
Definizione di funzioni

- I **parametri formali** di una funzione, nella sua definizione, si indicano con un simbolo di ***underscore*** dopo il nome della variabile:

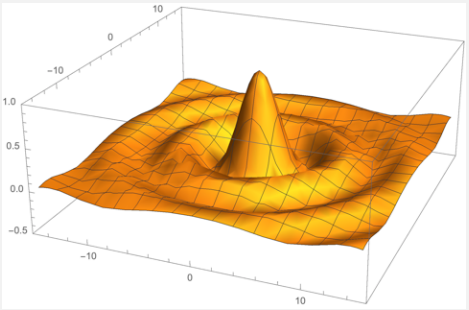
```
In[1]:= f[x_ , k_] := x^2 + k x
In[2]:= f[3,5]
Out[2] = 24
```

- È possibile visualizzare grafici di funzione e di superficie con le funzioni **Plot** e **Plot3D**:

```
In[1]:= f[x_] := Sin[x]/x
In[2]:= Plot[f[x], {x, -10, 15}]
Out[2]=
```




```
In[1]:= f[x_, y_] := Sin[Sqrt[x^2 + y^2]]/Sqrt[x^2 + y^2]
In[2] = Plot3D[f[x, y], {x, -15, 15}, {y, -15, 15}, PlotRange -> {-0.5, 1}]
Out[2]=
```



Liste, array e insiemi

- Le **parentesi graffe** devono essere utilizzate per definire liste di elementi
- Mathematica non distingue tra liste e vettori o array: gli elementi hanno una «posizione» nella lista e sono identificati da un numero progressivo (a partire da 1)
- Le **matrici** sono definite come **liste di liste**
- La funzione **Part** serve a restituire un elemento o una parte di un vettore o di una matrice, mentre la funzione **Length** restituisce la lunghezza (il numero di elementi) di una lista

```
In[1]:= x = {2, 3, 5, 7, 11}
Out[1] = {2, 3, 5, 7, 11}
In[2]:= m = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
Out[2] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
In[3]:= Part[m,2,3]
Out[3] = 6
In[4]:= Length[x]
Out[4] = 5
```

Operazioni su insiemi

- Mathematica dispone di funzioni per le operazioni elementari sugli insiemi:

```
In[1]:= Union[{1, 2, 3}, {3, 4, 5}]
Out[1]= {1, 2, 3, 4, 5}
In[2]:= Intersection[{1, 2, 3}, {2, 3, 4, 5}]
Out[2]= {2, 3}
In[3]:= Complement[{1, 2, 3}, {2, 4, 6}]
Out[3]= {1, 3}
```

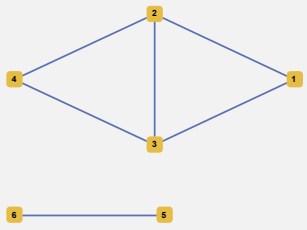
- Sono disponibili anche funzioni per **operazioni combinatorie** su insiemi finiti:

```
In[8]:= Permutations[{1, 2, 3}]
Out[8]= {{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2,1}}
In[9]:= Permute[{"a", "b", "c"}, {2, 1, 3}]
Out[9]= {"b", "a", "c"}
In[13]:= Subsets[{1, 2, 3}]
Out[13]= {{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}
In[14]:= Subsets[{1, 2, 3}, 2]
Out[14]= {{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}}
In[15]:= Subsets[{1, 2, 3}, {2}]
Out[15]= {{1, 2}, {1, 3}, {2, 3}}
```

Creazione di grafi

- È possibile costruire e rappresentare graficamente grafi e alberi
- Per creare un grafo si deve utilizzare la funzione **Graph** specificando l'insieme dei vertici e degli spigoli

```
In[1]:= G = Graph[{1, 2, 3, 4, 5, 6}, {1 <-> 2, 2 <-> 3, 2 <-> 4, 3 <-> 4, 1 <-> 3, 5 <-> 6}]
```



- Il grafo è orientato o non orientato a seconda della modalità con cui si specificano gli spigoli:

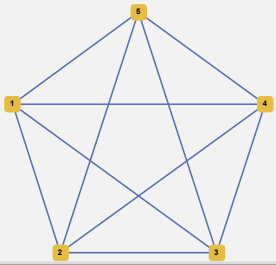
u*<->*v : non orientato

u*->*v : orientato da *u* a *v*

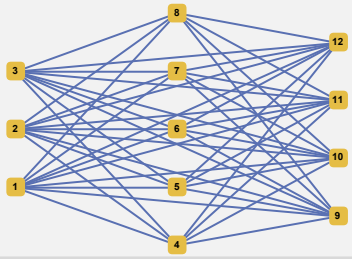
Creazione di grafi

- Si possono anche utilizzare funzioni per creare grafi casuali o sulla base di specifici parametri:

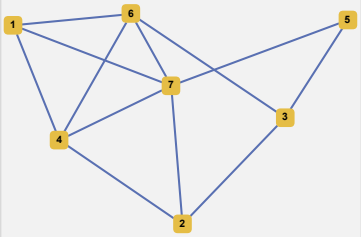
In[1]:= **G = CompleteGraph[5]**



In[1]:= **G = CompleteGraph[{3, 5, 4}]**

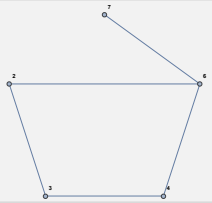


In[2]:= **G = RandomGraph{7,12}**



- È possibile modificare i grafi con le funzioni **VertexAdd**, **EdgeAdd**, **VertexDelete**, **EdgeDelete**:

```
In[1]:= G = CompleteGraph[5, VertexLabell->All]
In[2]:= G=VertexAdd[G, {6, 7}]
In[3]:= G=EdgeAdd[G, {2<->6, 6<->7, 4<->6}]
In[4]:= G=VertexDelete[G, {1, 5}]
In[5]:= G=EdgeDelete[G, {2<->4}]
```

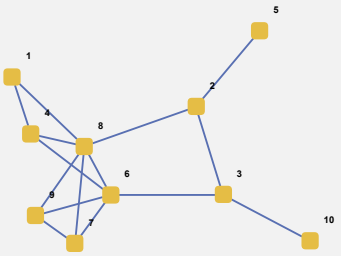


Proprietà del grafo

- È possibile verificare se il grafo possiede o meno determinate **proprietà**; le funzioni di Mathematica che termino con la lettera «**Q**» (*query*) sono funzioni booleane che restituiscono **True** o **False**:

```
In[1]:= G=RandomGraph[{10,15}, VertexLabels->All]
In[2]:= AcyclicGraphQ[G]
Out[2]= False
In[3]:= CompleteGraphQ[G]
Out[3]= False
In[4]:= ConnectedGraphQ[G]
Out[4]= True
In[5]:= PlanarGraphQ[G]
Out[5]= False
```

```
In[6]:= HamiltonianGraphQ[G]
Out[6]= False
In[7]:= EulerianGraphQ[G]
Out[7]= False
```



- È possibile estrarre anche **semplici informazioni** sul grafo:

```
In[1]:= EdgeCount[G]
Out[1]= 15
In[2]:= VertexCount[G]
Out[2]= 10
In[3]:= VertexDegree[G]
Out[3]= {2, 3, 3, 3, 1, 5, 3, 6, 3, 1}
```

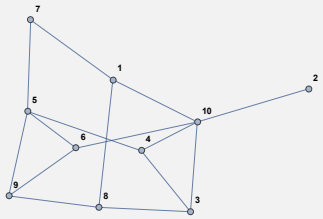
```
In[4]:= TableForm[AdjacencyMatrix[G]]
Out[4]=
0 1 0 0 1 … 0 1
0 0 0 1 0 … 1 0
```

Operazioni sui grafi

- È possibile ricavare alcuni sottografi attraverso operazioni di visita del grafo stesso:

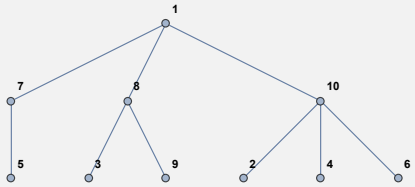
In[1]:= **G=RandomGraph[{10,15}, VertexLabels->Automatic]**

Out[1]=



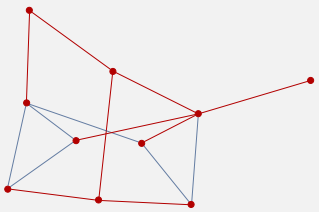
In[2]:= **FindSpanningTree[G, VertexLabels -> Automatic]**

Out[2]=



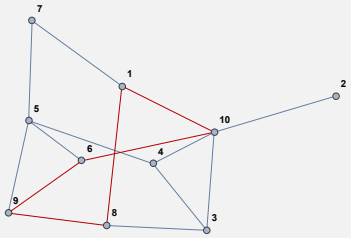
In[3]:= **HighlightGraph[G, FindSpanningTree[G]]**

Out[3]=



In[4]:= **HighlightGraph[G, FindCycle[G]]**

Out[4]=

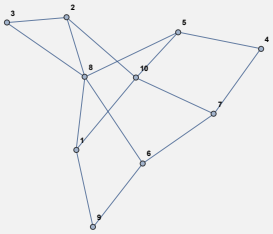


Operazioni sui grafi

- È possibile ricavare alcuni sottografi attraverso operazioni di visita del grafo stesso:

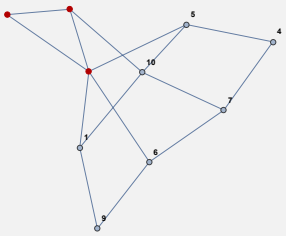
In[1]:= **G = RandomGraph[{10, 15}]**

Out[1]=



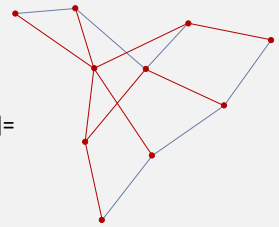
In[2]:= **HighlightGraph[G, FindClique[G]]**

Out[2]=



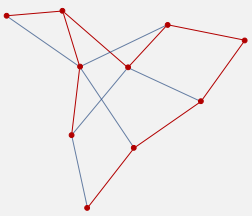
In[3]:= **HighlightGraph[G, TreeGraph[VertexList[G], BreadthFirstScan[G]]]**

Out[3]=



In[4]:= **HighlightGraph[G, TreeGraph[VertexList[G], DepthFirstScan[G]]]**

Out[4]=



Altre informazioni e riferimenti

- Dispense del corso IN440 – Ottimizzazione combinatoria (Mathematica ver. 10 con pacchetto «Combinatorica»):

  http://www.mat.uniroma3.it/users/liverani/doc/disp_oc_03.pdf

- Wolfram Language & System Documentation Center – Graphs & Networks:

  https://reference.wolfram.com/language/guide/GraphsAndNetworks.html