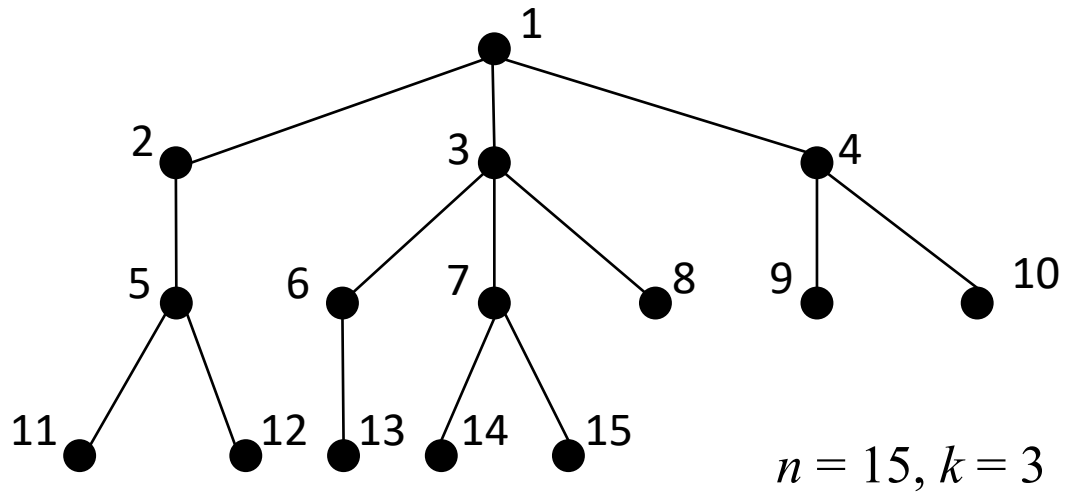
Alberi casuali in linguaggio Python

Corso Ottimizzazione Combinatoria (IN440)

Prof. M. Liverani

Alberi random

- Un **albero random** possiamo costruirlo fissando come parametri il numero n di vertici e il massimo numero di figli k che ciascun vertice può possedere (fino ad esaurimento del numero di vertici dell'albero che non può superare n)



- L'algoritmo di generazione di un albero random può essere il seguente:

Algoritmo 51 RANDOMTREE(n,k)

Input: n il numero di vertici; k il massimo numero di figli di ogni vertice

Output: l'albero T

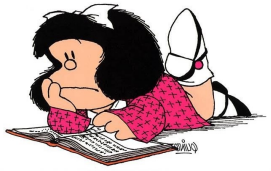
1: $T = (V,E), V = \{1,2,\ldots,n\}, E = \emptyset$
2: $x = n-1$
3: $u = 1, v = 2$
4: **fintanto che** $x > 0$ **ripeti**
5: sia y un numero casuale in $\{1,\ldots,\min(k,x)\}$
6: **per** $h = 1,\ldots,y$ **ripeti**
7: $E(T) = E(T) \cup \{(u,v)\}$
8: $v = v+1$
9: **fine-ciclo**
10: $x = x-y$
11: $u = u+1$
12: **fine-ciclo**
13: **return**

Alberi random

- In Python l'algoritmo può essere codificato come nella seguente funzione:

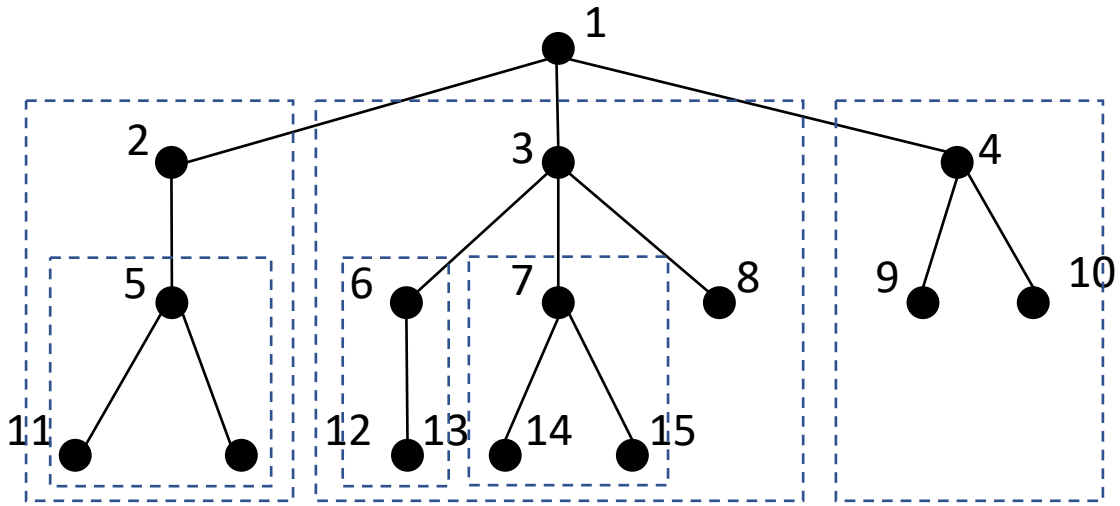
```
def randomTree(T, n, k):
  x = n - 1
  for u in range(1, n+1):
    T.addVertex(u)
  u = 1
  v = 2
  while x > 0:
    y = randint(1, min(k, x))
    for h in range(1, y+1):
      T.addEdge(u, v)
      v = v + 1
    x = x - y
    u = u + 1
  return
```

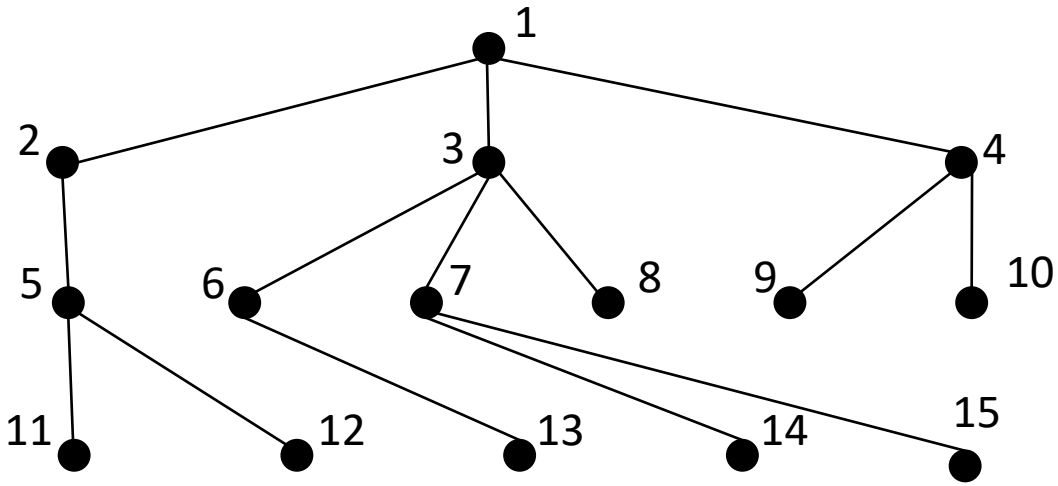
```
from in440    import *
from pythonds import *

T = Graph()
n = int(input("Numero di vertici: "))
k = int(input("Massimo numero di figli: "))
randomTree(T, n, k)
printGraph(T)
```



Disegno di alberi

- Un albero può essere disegnato nel modo tradizionale «top-down» a partire dalla radice (o da una foglia o da un vertice di scelto in modo arbitrario)
- Il problema principale è la distribuzione orizzontale dei vertici su ogni «livello» dell'albero (ossia come collocare in orizzontale i vertici equidistanti dalla radice?)
- Il modo più semplice è quello di distribuirli uniformemente sull'intera larghezza della finestra grafica, ma il risultato può non essere il più gradevole/efficace





Disegno di alberi

Algoritmo 52 TREEPLOT(T,s)

Input: L'albero T e la radice s

Output: Il disegno dell'albero

definisci una finestra di dimensione $maxX$, $maxY$
sia Q una coda e siano H,X,Y,P,N degli array tali che: H_i sia l'altezza del vertice i, (X_i,Y_i) le sue coordinate, P_i il padre e N_h il numero di vertici con la stessa altezza h
inizializza gli array con valori nulli
accoda s in Q
fintanto che $Q \neq \emptyset$ **ripeti**
    sia u un vertice estratto da Q
    per v adiacente a u **ripeti**
        $P_v = u$, $H_v = H_u + 1$
        accoda v in Q
    fine-ciclo
fine-ciclo
sia h il valore massimo in $H_1,\ldots,H_n$
$\Delta_y = maxY/h$
per ogni $u \in V(T)$ **ripeti**
    $N_{H_u} = N_{H_u} + 1$, $Y_u = H_u \times \Delta_y$
fine-ciclo
$X_s = maxX/2$
disegna la radice s nel punto (X_s,Y_s)
$oldH = 0$
per $u = 2,\ldots,n$ **ripeti**
    se $H_u \neq oldH$ **allora**
        $oldH = H_u$, $\Delta_x = maxX/N_{H_u}$, $x = \Delta_x$
    fine-condizione
    $X_u = x$, $x = x + \Delta_x$
    disegna il vertice u nel punto coordinate (X_u,Y_u)
    disegna lo spigolo da (X_{P_u},Y_{P_u}) a (X_u,Y_u)
fine-ciclo

Disegno di alberi

```
def treePlot(T, s):
  maxX = 800; maxY = 600
  win = GraphWin("Albero", maxX, maxY)
  n = len(T.getVertices())+1
  Q = Queue()
  H = np.zeros((n), dtype=int)
  X = np.zeros((n), dtype=int)
  Y = np.zeros((n), dtype=int)
  P = np.zeros((n), dtype=int)
  N = np.zeros((n), dtype=int)
  Q.enqueue(s)
  while not Q.isEmpty():
    u = Q.dequeue()
    for v in T.getVertex(u).getConnections():
      P[v.getId()] = u
      H[v.getId()] = H[u] + 1
      Q.enqueue(v.getId())
  h = max(H)
  dy = int((maxY-20)/h)
  for u in range(1, n):
    N[H[u]] = N[H[u]] + 1
    Y[u] = 10 + H[u] * dy
  X[s] = int(maxX / 2)
```

[continua ...]

[segue...]

```
  oldH = 0
  a = Circle(Point(X[s], Y[s]), 3)
  a.setFill("red")
  a.draw(win)
  Text(Point(maxX/2+8, 8), str(s)).draw(win)
  for u in range(2, n):
    if H[u] != oldH:
      oldH = H[u]
      dx = int(maxX / (N[H[u]] + 1))
      x = dx
    X[u] = x
    x = x + dx
    a = Circle(Point(X[u], Y[u]), 3)
    a.setFill("red")
    a.draw(win)
    Text(Point(X[u]+8, Y[u]-2), str(u)).draw(win)
    Line(Point(X[u], Y[u]), Point(X[P[u]], Y[P[u]])).draw(win)

  win.getMouse()
  return
```

Disegno di alberi