PermPy_2

July 10, 2022

1 Permutazioni (con un tocco di python) - parte II

1.1 Come manipolare le liste python

Abbiamo già definito la lista come un insieme di elementi (anche eterogeneo) racchiusi tra parentesi quadre.
Vediamo ora qualche metodo esposto dalla *classe* lista:

[26]:
```
# inizializzo una lista l
l = list(range(5))
l
```

[26]:
```
[0, 1, 2, 3, 4]
```

1.1.1 append(n)

Inserisce l'elemento n in ultima posizione

[27]:
```
l.append(8)
l
```

[27]:
```
[0, 1, 2, 3, 4, 8]
```

1.1.2 insert(x,n)

Inserice l'elemento n in posizione x

[28]:
```
l.insert(1,10)
l
```

[28]:
```
[0, 10, 1, 2, 3, 4, 8]
```

1.1.3 pop(x)

Restituisce l'elemento in **posizione** x *rimuovendolo* dalla lista (se x non è specificato rimuove l'elemento in ultima posizione)

[29]:
```
a=l.pop(2)
a
```

[29]: 1

1.1.4 remove(n)

Rimuove l'**elemento** n (se presente nella lista). Verificare prima l'esistenza dell'elemento nella lista con l'operatore **in**

[30]:
```
if 5 in l:
    l.remove(5)
else:
    print('Elemento non presente')
```

```
Elemento non presente
```

1.1.5 index(n,k)

Restituisce la **prima** posizione in cui è presente l'elemento n a partire dalla posizione k (se specificata, altrimenti dalla posizione 0)

[31]:
```
l.index(3,1)
```

[31]: 3

1.2 Factoradic e permutazioni ordinate

Abbiamo già detto che le permutazioni possono essere ordinate secondo un consueto criterio (alfabetico/numerico).
Senza perdere in generalità consideriamo gli elementi dell'insieme come numeri interi consecutivi che partono da 0 e definiamo come permutazione base la prima dell'ordinamento.
Per esempio p0 = [0, 1, 2, 3] su un insieme di 4 elementi.
Affianchiamo a ciascuna permutazione il *factoradic* espresso sullo stesso numero di elementi che costituiscono la permutazione corrispondente alla posizione:

0. p0 = [0, 1, 2, 3] factoradic(0) = [0, 0, 0, 0]
1. p1 = [0, 1, 3, 2] factoradic(1) = [0, 0, 1, 0]
2. p2 = [0, 2, 1, 3] factoradic(2) = [0, 1, 0, 0]

e così via...

22. p22 = [3, 2, 0, 1] factoradic(22) = [3, 2, 0, 0]

23. p23 = [3, 2, 1, 0] factoradic(23) = [3, 2, 1, 0]

Definiamo adesso un algoritmo che ci permetta di passare da una permutazione casuale qualsiasi al suo factoradic corrispondente (e quindi alla sua posizione) e viceversa.
Applicando queste *regole* saremo in grado di:

1. Data una *posizione* generare la permutazione corrispondente nell'insieme ordinato delle permutazioni
2. Data una permutazione determinarne la sua posizione nell'insieme ordinato delle permutazioni

In entrambi i casi avremo bisogno della permutazione base (la prima) che è facilmente generabile a partire dal numero di elementi che vogliamo compongano l'insieme:

Per esempio se n = 7 (permutazioni su un insieme di 7 elementi), la permutazione base si definisce con p0 = list(range(n))

1.3 Esempio (data una posizione ammissibile, generare la permutazione corrispondente)

Dati in **input**: numero di elementi **n**, posizione **k** (ammissibile)
Una posizione k è *ammissibile* su un insieme di n elementi se $0 \leq k < n!$
Risultato in **output**: **permutazione** in posizione k

1.3.1 Algoritmo in pseudo-codice

1. Genera le permutazione base di n elementi: p0 = list(range(n))
2. Inizializza una lista vuota p = []
3. Se k è ammissibile
    1. Genera il *factoradic* di k su n posizioni
    2. Per ciascun elemento x del factoradic
        1. Estrai da p0 l'elemento in posizione x
        2. Aggiungilo a p
4. Restituisci p

1.3.2 Applicazione dell'algoritmo su dati specifici

Eseguiamo le regole sopra definite sui seguenti dati:
n = 4, k = 15
genereremo, quindi la permutazione in posizione 15 (cioè la *sedicesima*, tenendo conto che le posizioni partono da 0)

1. p0 = [0, 1, 2, 3]
2. p = []
3. k è ammissibile, quindi:
    1. f(15) = [2, 1, 1, 0]
    2. x = 2 (primo elemento del factoradic)
        1. L'elemento in posizione 2 nella p0 è **2**
        2. p0 diventa [0, 1, 3], p diventa [2]
    3. x = 1 (secondo elemento del factoradic)
        1. L'elemento in posizione 1 nella p0 è **1**
        2. p0 diventa [0, 3], p diventa [2, 1]
    4. x = 1 (terzo elemento del factoradic)
        1. L'elemento in posizione 1 nella p0 è **3**
        2. p0 diventa [0], p diventa [2, 1, 3]
    5. x = 0 (quarto elemento del factoradic)
        1. L'elemento in posizione 0 nella p0 è **0**
        2. p0 diventa [], p diventa [2, 1, 3, 0]
4. Restituisci p = [2, 1, 3, 0]

[2]:
```python
# definizione della funzione python che restituisce il factoradic di un numero intero k su n posizioni
from math import factorial
def factoradic(n, k):
    # n è il numero di posizioni
    # k è il numero che vogliamo rappresentare come factoradic
    # k dovrà essere compreso tra 0 e n! (escluso)
    s = []
    if k < factorial(n):
        while n>1:
            n = n - 1
            q = k // factorial(n)
            k = k % factorial(n)
            s.append(q)
        s.append(0) # aggiunge lo 0 finale
    return s
```

[4]:
```python
# definizione della funzione python che traduce l'algoritmo sopra descritto
def getPerm(n, k):
    p0 = list(range(n))
    p = []
    if k < factorial(n):
        f = factoradic(n,k)
        for x in f:
            e = p0.pop(x)
            p.append(e)
    return p
```

[5]:
```python
getPerm(7,4792)
```

[5]:
```
[6, 3, 5, 2, 4, 0, 1]
```

[ ]:
```
```