Un’entropia enciclopedica

Il giorno fatidico sarà il 4 settembre, ossia il 247° giorno dell’anno di grazia 2021: 247 è infatti il numero di distinti “Doc Ordinamenti” su un insieme di 8 elementi.

Un Doc Ordinamento (DO_n) su un insieme di n elementi {1,..,n} è una permutazione P ($a_{1\dots}a_n$) su n tale che:

$$\exists!\, a_i \; per \; cui \; a_i > a_{i+1} \;\; 1 \leq i < n$$

mentre tutti gli altri elementi sono ordinati, ossia:

$$\forall\, j \neq i \;\; a_j < a_{j+1}$$

Per induzione su n dimostriamo che il numero di distinti Doc Ordinamenti (nDO) su un insieme di n elementi è:

$$nDO = \; 2^n - (n+1)$$

Per un insieme di 2 elementi {1,2} l’uguaglianza è vera visto che l’unico Doc Ordinamento è (2 1).

Supponiamo ora che l’uguaglianza sia vera per un insieme di n elementi e dimostriamo che per un insieme di n+1 elementi risulta:

$$(n+1)DO = 2^{n+1} - (n+2)$$

Osserviamo innanzitutto che da ciascun DO_n è possibile ricavare 2 distinti DO_{n+1}:

a) Il primo si ottiene semplicemente collocando l’elemento n+1 in posizione n+1 (a_{n+1} = n+1) e lasciando tutti gli altri nella medesima posizione;
b) Il secondo si ottiene ponendo l’elemento n+1 in posizione i+1 (a_{i+1} = n+1) ossia quella subito successiva ad a_i e scalando tutti gli elementi successivi in posizione i+2,…,n+1. Questa configurazione è un DO_{n+1} perché $a_i < a_{i+1}$ e $a_{i+1} > a_{i+2}$. Questa permutazione è diversa da quella al punto a, in particolare perché n+1 è in posizione i+1 < n+1

A questi Doc Ordinamenti se ne aggiungono altri n derivanti dall’ordinamento naturale:

$$(1\; 2 \dots n \; n+1)$$

che ovviamente non è un Doc Ordinamento ma che genera n Doc Ordinamenti spostando di volta in volta l’elemento n+1 nelle n posizioni da 1 a n, mentre gli altri elementi scalano mantenendo l’ordinamento iniziale. Questi n nuovi Doc Ordinamenti sono diversi da quelli ottenuti nel punto a (n+1 non è mai in posizione n+1) e al punto b (l’elemento che precede n+1 non è mai maggiore di quello che lo segue). In definitiva:

$$(n+1)DO = 2 * nDO + n = 2 * (2^n - (n+1)) + n = 2^{n+1} - (n+2)$$

***

Fin qui la parte formale che risponde al quesito e lo generalizza al caso di un insieme di n elementi.

Dal mio punto di vista è interessante anche raccontare come sono arrivato a questo risultato: si dà il caso infatti che io sia un informatico più che un matematico (posto che una tale caratterizzazione abbia senso) e quindi ho il vizietto di affrontare questi problemi nel modo classico per un informatico, che è quello del cosiddetto approccio di forza bruta. Vero è che questi approcci generalmente diventano intrattabili per

valori grandi di n, ma visto che il problema originario era per un insieme di 8 elementi ho ritenuto che la strada fosse percorribile.

Nell'approccio di forza bruta – come è noto – si generano tutte le soluzioni possibili e se il problema lo richiede si sceglie di volta in volta la soluzione che ottimizza una certa funzione. In questo caso occorreva semplicemente contare tutte le soluzioni ammissibili, soluzioni che si potevano facilmente costruire dall'insieme delle n! permutazioni di un insieme di n elementi.

Ecco allora che dopo una breve ricerca mi sono scaricato da Internet un programma in VBA capace di generare tutte le permutazioni di un insieme di n elementi: un bel programma davvero, con un piccolo numero di righe, ricorsivo nella generazione delle permutazioni e dotato di un controllo sulla massima dimensione di n per evitare effetti indesiderati a "run time".

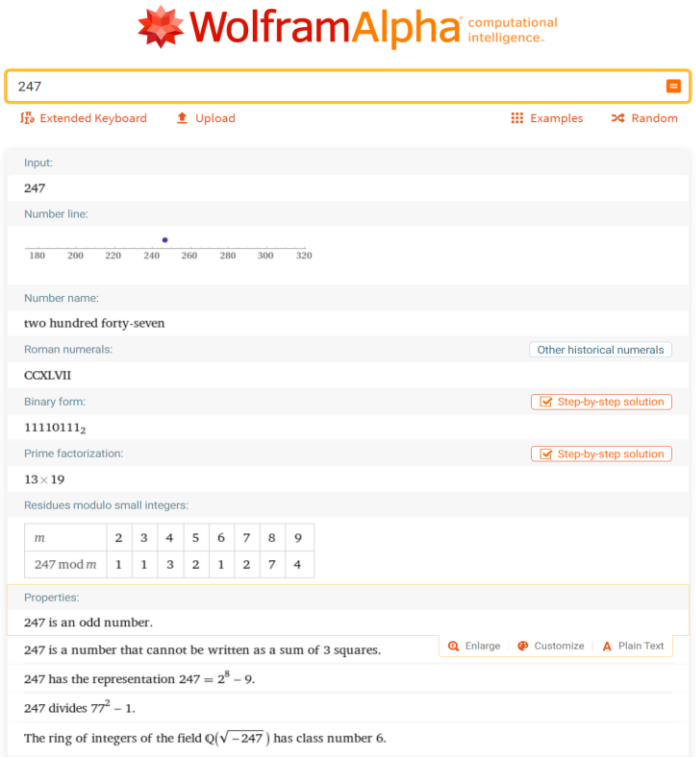
L'ho modificato perché producesse in output solo i famigerati Doc Ordinamenti: è stato piuttosto semplice perché un Doc ordinamento è una permutazione in cui un solo elemento è maggiore dell'elemento successivo. Il listato modificato (ringrazio gli autori dei KUtools) è presente in coda a questo documento.

L'ho eseguito per n=8 ed ho ottenuto la lista di tutti i 247 Doc Ordinamenti.

Ero felicissimo, naturalmente, anche perché avevo impiegato pochissimo tempo nel pensare e attuare questo approccio: ero riuscito già a rispondere al quesito e avevo anche la lista completa di tutti i Doc Ordinamenti. Mi sono tuttavia chiesto: è un approccio corretto? Come posso garantire che l'approccio è completo?

L'implementazione dell'algoritmo ricorsivo mi sembrava corretta, così come anche la modifica semplice che avevo fatto per produrre i Doc ordinamenti. Sarebbe stato dunque sufficiente darvi come risposta il listato con l'output della sua esecuzione per n=8? Penso di sì, francamente: ma sarebbe stato formalmente corretto?

Fatto sta che ho cercato di approfondire e sono partito innanzitutto dal 247. Che numero affascinante, inatteso in un certo senso. L'ho digitato su WolframAlpha ed ho ottenuto questo:



La fattorizzazione mi sembrava bellissima (13*19), ma mi aveva colpito immediatamente anche quel rifermento alla rappresentazione di 247 come differenza tra 2^8 e 9. Ecco: il numero 8 era una traccia molto promettente. Ho subito eseguito il programma per altri input, riportando i risultati nella seguente tabella:

n	nDO
2	1
3	4
4	11
5	26
6	57
7	120
8	247
9	502
10	1013

Ho potuto quindi verificare immediatamente che per i primi valori di n la formula intuibile grazie a WA era giusta, ossia che $nDO = 2^n-(n+1)$.

Una dimostrazione formale sembrava a questo punto naturale e il principio di induzione l'arma giusta da usare. Ogni esecuzione dell'algoritmo sembrava confermare la correttezza della formula, almeno finché le risorse del computer erano state sufficienti per farlo. E qui c'è stato spazio per un nuovo spunto di riflessione: ogni nuova esecuzione dell'algoritmo è una verifica che la formula ipotizzata sia vera. Quindi la correttezza dell'algoritmo unita al fatto il fatto che ogni sua esecuzione – nel limite delle risorse usabili – sia in accordo con la formula può essere considerata alla pari di una dimostrazione formale?

Mi piacerebbe approfondire questa questione: magari potreste rimandarmi a qualche vostro articolo o a qualche testo in materia.

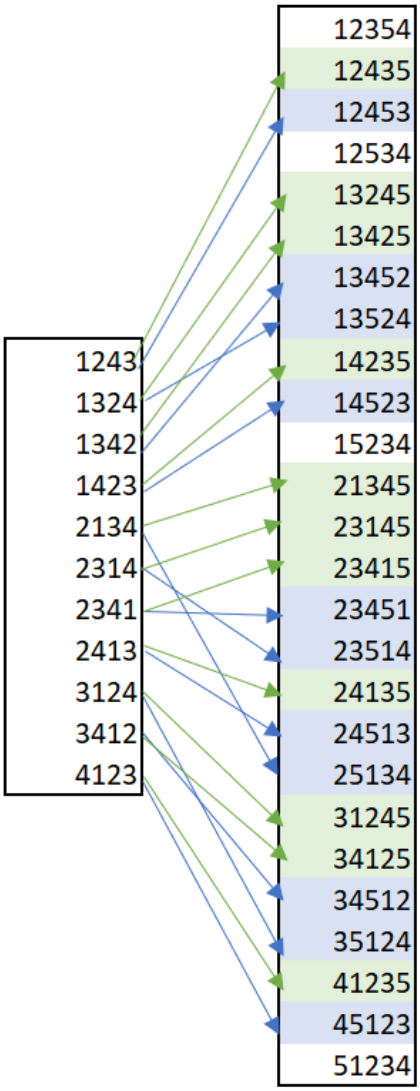
Avrei potuto quindi scrivervi: la risposta al quesito è 247, questo è il listato, mi sembra corretto. Per n=8 dà 247 negli altri casi la formula è $nDO = 2^n-(n+1)$. Stringato e semplice. Ma ho voluto comunque esplorare la cosa tramite il principio di induzione.

La base induttiva è importantissima: ho visto che per n=2 i conti tornavano semplicemente.

Ma la prova induttiva non è stata così immediata: l'ipotesi e la tesi erano chiare, ma come procedere? E allora sono andato a lavorare sui Doc Ordinamenti generati dall'algoritmo: quella è stata la parte più divertente, lavorare con le stringhe numeriche, da operaio della matematica, confrontarle e ragionare. Ho scelto in particolare di confrontare i 4DO (i Doc Ordinamenti con 4 elementi) con i 5DO: mi sembrava la dimensione più adatta per fare un confronto, non troppo grande, né troppo piccola. La Gestalt è stata efficace (vedi figura successiva). Mi è venuta subito un'illuminazione, quella per me più intuitiva, ossia che mettendo il 5 subito dopo l'elemento che nel 4DO individuava la discesa avrei ottenuto una 5DO (in blu in figura). Avevo inizialmente ipotizzato - per simmetria - che potessi ottenere un 5DO anche mettendo il 5 prima del medesimo elemento, ma ovviamente i conti non tornavano (c'erano due discese consecutive). Guardando i 5DO si vede subito però che mettendo il 5 in quinta posizione ogni 4DO ne generava una (in verde in figura). Alleluja, questo significava che ogni 4DO generava 2 5DO. Ero vicino alla soluzione che è balzata agli occhi quando ho esaminato i 5DO residui, quelli – per intenderci – senza freccia in figura. Nell'ordine:

12354, 12534, 15234, 51234 (in totale 4 Doc Ordinamenti)

Qui lo schema era chiarissimo, basta veder scivolare allegramente il 5 verso sinistra una posizione alla volta.



Mi sono messo quindi a verificare lo stesso procedimento da 3DO a 4DO e da 5DO a 6DO: tutto si è concluso esattamente come mi aspettavo.

A quel punto ho fatto i conti finali: poiché ogni nDO ne genera 2, raddoppiando l'nDO ed aggiungendo n ho potuto calcolare (n+1)DO e verificare che il valore da dimostrare nella tesi tornava perfettamente. Evviva.

Nella dimostrazione formale ho solo messo in rilievo come nessun nuovo DO_{n+1} fosse costruito algoritmicamente in modo da generare duplicazioni. Già, alla fine la dimostrazione formale per induzione è una dimostrazione squisitamente algoritmica.

Concludo questo mio excursus con un consiglio a Doc: la prossima volta, regala al GC i favolosi Quindici: certo una forma di enciclopedia più semplice e meno universale, un po' vintage ma sarà un regalo certamente graditissimo. Poi ti puoi divertire con i 32.752 Doc Ordinamenti: con uno al giorno fanno poco meno di 90 anni, questa sì che è una sfida.

Il programma in VBA

Il programma originale è alla URL che segue; io l'ho modificato solo nelle righe in bold per determinare i DOC Ordinamenti:
https://www.extendoffice.com/documents/excel/3657-excel-generate-all-permutations.html

```
Sub GetString()
'Updateby Extendoffice
   Dim xStr As String
   Dim FRow As Long
   Dim xScreen As Boolean
   xScreen = Application.ScreenUpdating
   Application.ScreenUpdating = False
   xStr = Application.InputBox("Enter text to permute:", "Kutools for Excel", , , , , , 2)
   If Len(xStr) < 2 Then Exit Sub
   If Len(xStr) > 10 Then
      MsgBox "Too many permutations!", vbInformation, "Kutools for Excel"
      Exit Sub
   Else
      ActiveSheet.Columns(1).Clear
      FRow = 1
      Call GetPermutation("", xStr, FRow)
   End If
   Application.ScreenUpdating = xScreen
End Sub

Sub GetPermutation(Str1 As String, Str2 As String, ByRef xRow As Long)
   Dim i As Integer, xLen As Integer, TmpStr As String, cntsucc As Integer
   xLen = Len(Str2)
   If xLen < 2 Then
      'This is to calculate a Doc Order (cntsucc =1)
      TmpStr = Str1 & Str2
      cntsucc = 0
      For i = 1 To Len(TmpStr) - 1
         If Mid(TmpStr, i, 1) > Mid(TmpStr, i + 1, 1) Then
            cntsucc = cntsucc + 1
         End If
      Next
      'Output only if a Doc Order
      If cntsucc = 1 Then
         Range("A" & xRow) = Str1 & Str2
         xRow = xRow + 1
      End If
   Else
      For i = 1 To xLen
         Call GetPermutation(Str1 + Mid(Str2, i, 1), Left(Str2, i - 1) + Right(Str2, xLen - i), xRow)
      Next
   End If
End Sub
```