```
Procedure Permutazioni(P, k):
    if k = n then Controllo(P)
            else for i ← k to n do
                scambia P[k] ↔ P[i];
                Permutazioni(P, k + 1);
                scambia P[k] ↔ P[i].
```

Il calcolo è avviato con la chiamata iniziale `Permutazioni(`$P, 1$`)`. L'algoritmo è basato sull'osservazione che le permutazioni di n elementi possono essere divise in gruppi, ponendo in ciascun gruppo quelle che iniziano con il primo, il secondo,..., l'n-esimo elemento ciascuno seguito dalle permutazioni degli altri $n-1$. Nel primo gruppo troviamo $P[1]$ seguito da tutte le permutazioni di $P[2], \ldots, P[n]$; nel secondo troviamo $P[2]$ seguito da tutte le permutazioni di $P[1], P[3], \ldots, P[n]$ e così via. Questa definizione è induttiva e la correttezza dell'algoritmo si può quindi dimostrare per induzione: occorre solo notare che la seconda operazione di scambio tra $P[k]$ e $P[i]$ è necessaria per ripristinare l'ordine degli elementi dopo la costruzione ricorsiva delle permutazioni degli elementi che seguono il primo. Più difficile è capire il funzionamento dell'algoritmo e individuare l'ordine in cui vengono costruite le permutazioni. Invitiamo il lettore a rifletterci un momento simulando a mano il comportamento dell'algoritmo sull'insieme $\{1, 2, 3\}$: otterrà nell'ordine le permutazioni: 1,2,3 - 1,3,2 - 2,1,3 - 2,3,1 - 3,2,1 - 3,1,2, e con l'ultima operazione di scambio ripristinerà l'ordine iniziale 1,2,3 degli elementi.

La costruzione di permutazioni interviene per esempio nel *problema del ciclo hamiltoniano* $\mathcal{P}_{ham}(G)$, paradigma di riferimento per molti problemi di percorsi. Ricordiamo anzitutto che un grafo $G = (V, E)$ è composto da un insieme V di vertici, e un insieme E di spigoli che connettono coppie di vertici. La struttura può essere convenientemente descritta rappresentando i vertici con gli interi $1, \ldots, n$, con $n = |V|$, e rappresentando E con una matrice binaria A *di adiacenza*, di dimensioni $n \times n$, ove si pone $A[i, j] = 1$ se esiste uno spigolo che connette il vertice i al vertice j, $A[i, j] = 0$ se tale spigolo non esiste. La dimensione della descrizione di G è dunque di ordine $\Theta(n \log n)$ per rappresentare V più $\Theta(n^2)$ per rappresentare E: quindi in totale è di ordine $\Theta(n^2)$. Un *percorso* in un grafo è una sequenza di vertici $v_1, v_2, \ldots, v_k$ tale che esiste lo spigolo da v_i a v_{i+1} per ogni coppia di vertici consecutivi nel percorso. Se v_k coincide con v_1 il percorso è detto *ciclo*. Dato un grafo arbitrario G il problema $\mathcal{P}_{ham}(G)$ chiede di stabilire se esiste un ciclo in G che contiene tutti i vertici esattamente una volta.

Se associamo al grafo G un vettore P contenente gli interi tra 1 e n, una permutazione di P rappresenta una sequenza che contiene tutti i vertici: la permutazione rappresenta quindi una soluzione di $\mathcal{P}_{ham}(G)$ (cioè un ciclo hamiltoniano) se nella

matrice di adiacenza di A si ha $A[P[i], P[i+1]] = 1$ per ogni $1 \leq i \leq n-1$, e $A[P[n], P[1]] = 1$. Si possono allora costruire tutte le permutazioni in P con l'algoritmo `Permutazioni` e risolvere $\mathcal{P}_{ham}(G)$ specificando la procedura di controllo come:

`Procedure Controllo(`P`)`:
if $(A[P[i], P[i+1]] = 1,\ 1 \leq i \leq n-1)$ **and** $(A[P[n], P[1]] = 1)$
then success.

Come visto in precedenza il comando **success** arresta il calcolo decretando che esiste una soluzione. La complessità dell'algoritmo così costruito è $T(n) = \mathrm{O}(n \cdot n!)$ ove il fattore n nel prodotto rappresenta il tempo per il controllo nella matrice A, ed è quindi esponenziale nella dimensione dell'ingresso.

Il problema $\mathcal{P}_{ham}(G)$ è stato posto in forma decisionale ma in pratica si può richiedere di comunicare all'esterno la soluzione trovata, o una soluzione con determinate proprietà. In una sua famosa versione che prende il nome di *problema del commesso viaggiatore* $\mathcal{P}_{cv}(G)$, si assegna una *lunghezza* a ogni spigolo del grafo e si chiede di trovare il ciclo hamiltoniano (se ne esiste uno) di minima lunghezza complessiva: si tratta per esempio di stabilire il percorso più breve per la testa di un trapano a controllo numerico con cui si devono fare molti fori in un pezzo meccanico. Lo schema di calcolo è sempre lo stesso e la complessità è ovviamente sempre esponenziale: anche $\mathcal{P}_{ham}(G)$ non sembra risolubile in tempo polinomiale, ma non si conosce dimostrazione in proposito.

3.4 Le classi P, NP, co-NP e NPC

Dopo aver visto come risolvere alcuni problemi in tempo polinomiale e aver proposto per altri solo algoritmi esponenziali, dobbiamo fare un po' di ordine nella teoria. Questo ci aiuterà a impostare organicamente i problemi anche se lascerà aperte alcune questioni fondamentali per cui mancano risultati definitivi.

Anzitutto ogni ente di cui trattiamo dovrà essere rappresentato in modo efficiente nel senso tecnico del termine (capitolo 2); cioè i dati, gli algoritmi, e ogni struttura che questi possano costruire durante il calcolo avranno dimensione logaritmica nel numero di elementi della classe cui appartengono. In particolare un numero N sarà rappresentato con $\Theta(\log N)$ caratteri, o precisamente con $\lceil \log_2 N \rceil$ cifre binarie. Un insieme di n elementi sarà rappresentato con $\Theta(n)$ caratteri nell'ipotesi che ogni elemento abbia dimensione costante d. Notiamo però che in questo modo si possono rappresentare al più a^d elementi distinti, ove a è la cardinalità dell'alfabeto. Poiché