Esercizio

Si scriva una procedura PERMUTAZIONI basata sulla tecnica backtrack per stampare tutte le permutazioni degli n elementi di un vettore V.

Punti d'attenzione

Utilizzare un vettore di supporto per ricordarsi gli elementi mancanti.

Soluzione proposta

L'idea base è quella di fissare la posizione 1 con uno degli n elementi, poi il secondo con i rimanenti $n-1$, e cosi via. Il backtrack si applica quando si torna sull'elemento j-esimo precedentemente fissato per fissarlo con uno dei valori non ancora utilizzati.

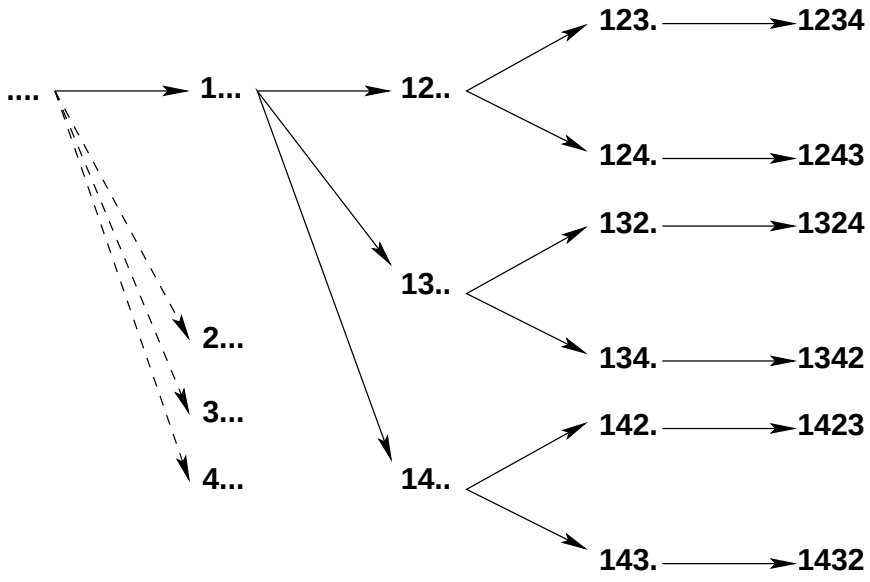

Figura 1: Esempio di permutazione di interi

La figura 1 descrive un esempio, per $n = 4$, di quanto descritto in precedenza: l'algoritmo di backtrack dovrà agire come una previsita dell'albero descritto in figura 1.

L'algoritmo utilizza due vettori: `SOL`, che contiene la permutazione corrente e `USED` che tiene traccia dagli elementi già utilizzati nella permutazione corrente. L'algoritmo utilizza la funzione `stampa(int *);` per stampare a video la permutazione corrente.

```
void PERMUTAZIONI(int * SOL, int k, boolean * USED, int n){
int i;
  if (k==n) stampa(SOL);
  else{
    for (i=0; i<n; i++)
      if (!USED[i]){
```

```
        SOL[k] = V[i];
        USED[i] = 1;
        PERMUTAZIONI(SOL, k+1, USED, n);
        USED[i] = 0;
      }
  }
}
```

La funzione `PERMUTAZIONI` viene chiamata con `PERMUTAZIONI(SOL, 0, USED, n)`.

Alla chiamata i-esima, la procedura ricorsiva genera $n - i$ chiamate ricorsive ottenute fissando agli $n - i$ interi non ancora utilizzati la posizione i-esima del vettore.

La complessità della procedura è data dalla seguente funzione:

$$T(n) = \begin{cases} n & n = 0 \\ nT(n-1) + n & n > 0 \end{cases}$$

dalla quale si ottiene, per sostituzioni successive che

$$T(n) \geq n \times n - 1 \ldots \times 1 = n!$$

ovvvero, come prevedibile, la procedura ha complessità almeno fattoriale.