1. **Wariacją** m-elementową n-elementowego zbioru S jest każdy ciąg m elementów zbioru S. W przypadku wariacji kolejność elementów ma znaczenie. Jeżeli elementy wariacji nie mogą się powtarzać, mówimy o **wariacji bez powtórzeń**. W przeciwnym razie mamy do czynienia z **wariacją z powtórzeniami**.

   Liczba możliwych m-elementowych wariacji zbioru n-elementowego bez powtórzeń V_n^m wynosi $n\cdot(n-1)\cdot\ldots\cdot(n-m+1)$.

   Liczba możliwych m-elementowych wariacji zbioru n-elementowego z powtórzeniami $\overline{V_n^m}$ wynosi n^m

2. N-elementowe wariacje bez powtórzeń n-elementowego zbioru S nazywamy **permutacjami**. Istnieje $n! = n\cdot(n-1)\cdot\ldots\cdot 2\cdot 1$ permutacji P_n zbioru S.

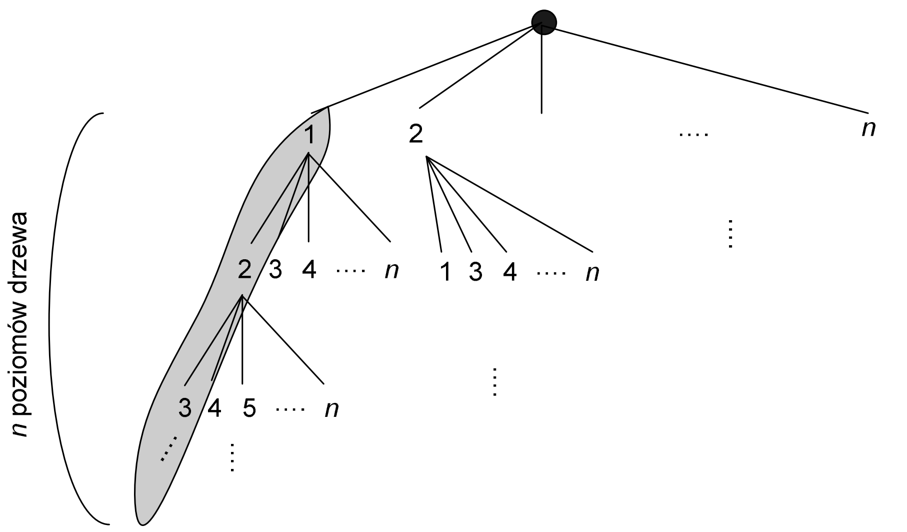
3. **Kombinacją** m-elementową zbioru *S jest każdy* m-elementowy podzbiór zbioru *S.* W przypadku kombinacji kolejność elementów nie ma więc znaczenia. Liczba możliwych m-elementowych kombinacji C_n^m zbioru n-elementowego wynosi:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

4. $V_n^m = C_n^m \cdot P_m$

Wskazówki programistyczne:

1. Generowanie permutacji n-elementowych
   Można zastosować strategię budowania i przeglądania drzewa „w głąb". Liczba poziomów drzewa (nie licząc korzenia) jest równa liczbie elementów zbioru (czyli n). Tworząc drzewo należy zadbać, aby element umieszczany w każdym kolejnym węźle nie występował wcześniej na ścieżce prowadzącej od danego węzła do korzenia drzewa. Poniższy przykład pokazuje możliwą strategię uzyskania wszystkich permutacji zbioru {1, 2, …, n}



   Identyfikacja wszystkich permutacji może polegać na wypisaniu wszystkich możliwych sekwencji elementów umieszczonych w węzłach (z pominięciem znaczenia korzenia) zgodnie z kolejnością ich występowania na ścieżkach prowadzących od korzenia drzewa do każdego liścia węzła.

2. Generowanie wariacji m-elementowych zbioru n-elementowego bez powtórzeń
   Można zastosować podobną technikę jak w punkcie 1) przy ograniczeniu liczby tworzonych poziomów drzewa do $m \le n$

3. Generowanie wariacji m-elementowych zbioru n-elementowego z powtórzeniami
   Można zastosować podobną technikę jak w punkcie 1) przy ograniczeniu liczby tworzonych poziomów drzewa do $m \le n$ oraz przy dopuszczeniu możliwości powtarzania się elementów na ścieżkach od korzenia do liści.

4. Generowanie kombinacji *m*-elementowych zbioru *n*-elementowego bez powtórzeń

W przypadku kombinacji kolejność elementów nie ma znaczenia. Przykładowo oznacza to że zbiory {1, 3, 2} i {3, 1, 2} są tożsame. Rozsądne jest więc przyjęcie określonej reprezentacji danej kombinacji np. określonej niemalejącym uporządkowaniem elementów. Dla rozpatrywanego przykładu trzech elementów uzyskujemy wtedy zbiór {1, 2, 3}.

Wygodnie jest generować wszystkie kombinacje w **porządku leksykograficznym**, określonym następująco:

$$\langle x_1, x_2, ..., x_m \rangle < \langle y_1, y_2, ..., y_m \rangle \Leftrightarrow istnieje\ k \geq 1,\quad takie\quad że\quad x_k < y_k\quad oraz\quad x_l = y_l\quad dla\quad kazdego\quad l < k$$

Porządek leksykograficzny stosuje się np. w celu uporządkowania wyrazów w słownikach. Przykładowo wyraz „koc" jest leksykograficznie większy niż „jamnik", ponieważ już na pierwszej pozycji litera „k" znajduje się na dalszej pozycji w alfabecie („k" jest po „j"), mimo iż na następnych pozycjach może być inaczej.

Inne przykłady:

$\langle 1,3,4 \rangle < \langle 2,1,4 \rangle$

$\langle 2,1,4 \rangle < \langle 2,1,5 \rangle$

yeti < zamek

Wygodnie jest zrealizować funkcję generującą C_n^m jako wypisującą wszystkie możliwe kombinacje *m*-elementowe zbioru *n*-elementowego w porządku leksykograficznym.

Przykład:

Dla zbioru {1, 2, 3, 4, 5, 6} wszystkie możliwe kombinacje czteroelementowe wypisane w porządku leksykograficznym są następujące:

1 2 3 4
1 2 3 5
1 2 3 6
1 2 4 5
1 2 4 6
1 2 5 6
1 3 4 5
1 3 4 6
1 3 5 6
1 4 5 6
2 3 4 5
2 3 4 6
2 3 5 6
2 4 5 6
3 4 5 6

Literatura:

W. Lipski, *Kombinatoryka dla programistów* (rozdział 1), WNT, seria *Klasyka informatyki*, 2004