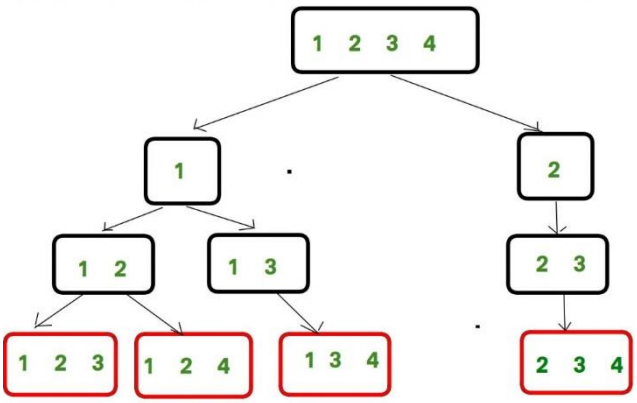
Hľadanie kombinácií r elementov v poli s veľkosťou N – bez opakovania

Funkcionalita

Povedzme, že máme pole a[n] = { 1, 2, 3, 4}; a chceme vytvoriť kombinácie r = 2 elementov. Výsledkami v takom prípade sú kombinácie (1, 2); (1, 3); (1, 4); (2, 3); (2, 4); (3,4). Počet vzniknutých kombinácií vieme dopredu ľahko vypočítať matematickým vzorcom.

Teraz si vezmime ako príklad pole **a[n]** = { 1, 2, 3, 4}; a **r = 3**. Pri vytváraní kombinácie máme pri každom prvku poľa dve možnosti – použiť ho, alebo ho nepoužiť. Keďže ale r = 3, vieme, že nemôžeme vylúčiť všetky prvky, lebo 3 použiť musíme.

Vezmime si teda **prvý element, 1**. V prípade, že ho použijeme, máme zatiaľ kombináciu s dĺžkou 1, zostali nám 4 elementy, a vieme, že máme použiť ešte dva. Takto sa môžeme posúvať po poli kým nedostaneme kombináciu **(1, 2, 3)** a nezastavíme.



Vtedy sa môžeme vrátiť po kombinačnom strome naspäť na kombináciu (1, 2) a **vynecháme trojku**, lebo vieme, že tú kombináciu sme už vypísali. Takto sa posunieme na 4 a získame kombináciu **(1, 2, 4).**

Vrátime sa zasa naspäť na kombináciu (1, 2) a keďže sme už vypísali všetky kombinácie až po 4, vrátime sa naspäť až ku kombinácií iba (1), a namiesto použitia čísla 2, použijeme ďalší prvok, číslo 3. Dostaneme tak kombináciu (1, 3) a posunieme sa na 4 ktorú použijeme a získame kombináciu **(1, 3, 4)**. Použili sme všetky elementy pre kombináciu (1, 3). Keďže máme vytvoriť kombináciu r = 3 prvkov, vieme, že môžeme vylúčiť maximálne jeden prvok zo štyroch, a tým pádom **posunúť sa na kombináciu (1, 4) nemá zmysel**, lebo by sme vylúčili viac ako jeden prvok, a nezostal by nám tretí prvok do kombinácie.

Posunieme sa teda na kombináciu (2) a pokračujeme rovnako ako doteraz. Získame tak kombináciu **(2, 3, 4).**

Emma Macháčová

Program

```c
#include <stdio.h>

void combinationUtil (int arr[], int data[], int start, int end, int index, int r);

// hlavná funkcia, ktorá vypíše všetky kombinácie o veľkosti r z poľa arr[] s veľkosťou n
void printCombination(int arr[], int n, int r)  {
    // pomocné pole na postupné ukladanie všetkých kombinácií
    int data[r];
    // vypísanie kombinácií pomocou poľa data[]
    combinationUtil (arr, data, 0, n-1, 0, r);
}

// arr[] --> vstup, data[] --> pomocné pole,
// start a end --> začiatočný a konečný index pola arr[]
// index --> aktuálne používaný index v poli data[], r --> želaná veľkosť kombinácie
void combinationUtil(int arr[], int data[], int start, int end, int index, int r) {

    // ak máme kombináciu o správnej dĺžke
    if (index == r) {
        for (int j = 0; j < r; j++)
            printf ("%d ", data[j]);
        printf ("\n");
        return;
    }

    // postupne nahradíme index v poli data[] všetkými možnýli elementmi pola arr[]
    // podmienka zabezpečí, že zahrnutie elementu do indexu spraví kombináciu
    // s ostatnými elementami na zostávajúcej pozícií
    for (int i = start; i <= end && end-i+1 >= r-index; i++) {
        data[index] = arr[i];
        combinationUtil (arr, data, i+1, end, index+1, r);

    // zabránenie duplikátnym kombináciam
       while (arr[i] == arr[i+1])
           i++;
    }
}

int main()  {
    int arr[] = {1, 2, 3, 4};
    int r = 3;
    int n = sizeof(arr) / sizeof(arr[0]);

    // volanie hlavnej rekurzívnej funkcie
    printCombination (arr, n, r);
}
```

Užitočné odkazy

https://www.youtube.com/watch?v=GuTPwotSdYw

https://drive.google.com/file/d/1LwvCYLx6FU2egVvUGRb3VGRtyZsmNZNZ/view

https://www.geeksforgeeks.org/print-all-possible-combinations-of-r-elements-in-a-given-array-of-size-n/

https://ideone.com/ywsqBz