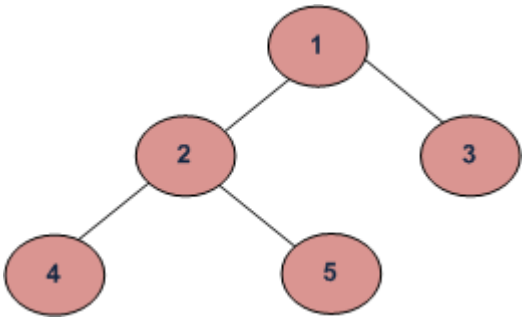
**Задача 1.** Изведете списък от елементите на двоично дърво в последователност:

- Inorder (Left, Root, Right) : 4 2 5 1 3
- Preorder (Root, Left, Right) : 1 2 4 5 3
- Postorder (Left, Right, Root) : 4 5 2 3 1




**Задача 2.** Напишете функция която проверява дали дадено двоично дърво е валидно двоично наредено дърво. На функцията се подава само указател към корена му.

**Задача 3.** Да се напише функция tree-map(Node* root, Function f), която прилага функцията f на всяка стойност от двоично дърво с корен root.
*Функцията запазва ли двоичната форма на дървото?*
*А остава ли валидно двоично наредено дърво за търсене?*

**Задача 4.** Да се напише функция trim(Node* root), която премахва всички листа в двоично дърво с корен root.

**Задача 5.** Да се напише функция bloom(Node* root), която заменя всяко листо със стойност *x* със следното дърво:

```
    x
   / \
  x   x
```

*Функцията запазва ли "наредеността" на BST?*

**Задача 6.** Проверете дали стойностите на всички N възли на балансирано двоично наредено дърво са от даден интервал `[p, q]`.
*Забележка: Нека алгоритъмът да работи за време O(log(N)) в най-лошия.*

**Задача 7.** Напишете функция, която намира и връща **височината** на произволно дърво.

**Задача 8.** Напишете функция, която намира и връща **коефициента на разклонение (branching factor)** на произволно дърво.

**Задача 9.** Напишете функция, която по подаден корен на произволно дърво и ниво **level** извежда на конзолата всички елементи на ниво level от корена на дървото.

**Задача 10.** Напишете функция, която по подаден корен на произволно дърво връща броя на листата му.

**Задача 11.** Напишете функция, която по подадени N на брой числа (във вектор) извежда на екрана числата в сортиран вид. Нека функцията в най-лошия случай да работи за време O(N*log(N)) и да използва O(N) допълнителна памет.
*Забележка: Без да ползвате сортиращи алгоритми.*

**Задача 12.** Напишете функция, която приема един аргумент – корен на дърво, чиито възли могат да имат произволен брой деца. Всеки възел пази стойност – цяло число.

Функцията трябва да намери онзи път от корена на дървото, до някое листо, който има най-малка сума на елементите в него. Този път трябва да се върне, като вектор от възлите в него. Ако има няколко такива пътя, може да се върне който и да е от тях. Напишете кратка програма, която демонстрира работата на функцията върху примерно дърво. По-долу са дадени представянето на дървото и прототипът на функцията, която трябва да реализирате. Те трябва да са точно такива, както са показани тук.

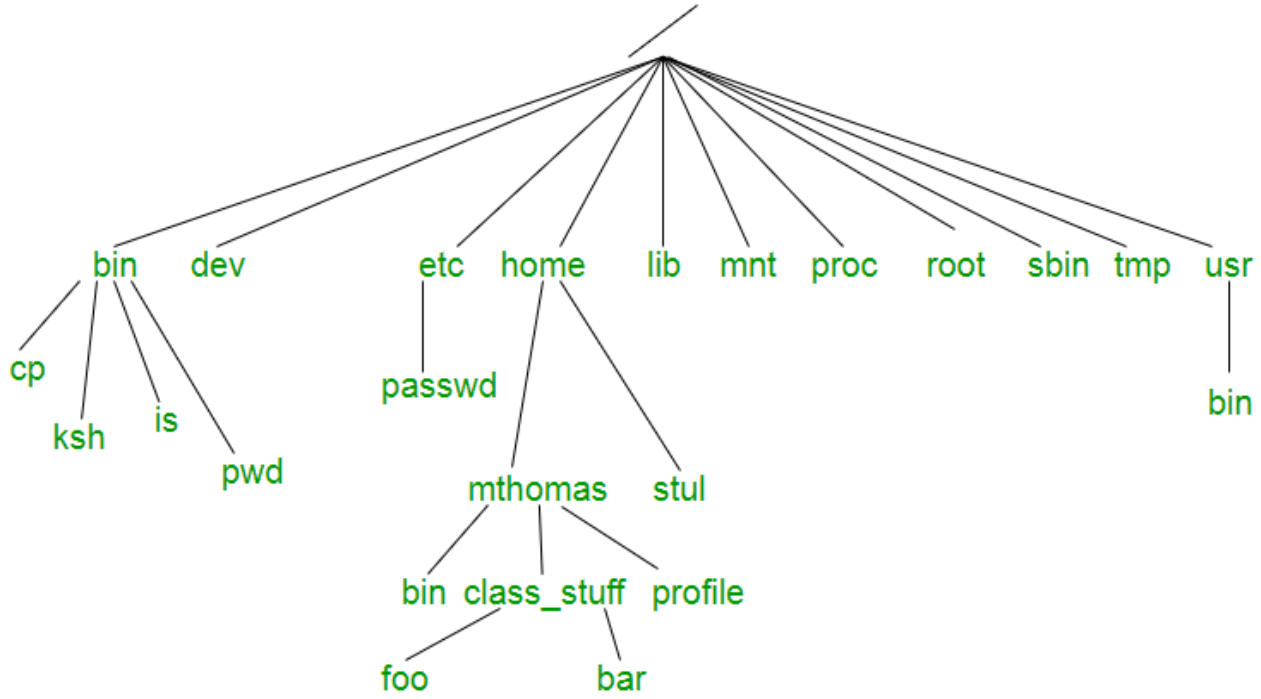
```
struct Node {
    int value;
    std::vector<Node> children;
};

std::vector<Node*> findCheapestPath(Node *root);
```

# Задачи за самоподготовка

**Задача 1.** Проверете дали съществува път от корена на дърво до някое негово листо, такъв че сбора от стойностите на всички върхове през, които минава пътя е равен на **n**.

**Задача 2.** Разглеждаме Unix файловата система. Тя е дървовидна, всяка директория е разделена с “/”, а най-отгоре на ФС стои root директорията. Напишете програма, която по подадено име на начална директория и подадено име на крайна директория или име на файл, намира релативният път на търсената/ият директория/файл.




*Пример:*

*Вход:* `home bar`

*Изход:* `./mthomas/class_stuff/bar`

**Задача 3.** Намерете броя на поддърветата на двоично наредено дърво, на които всички възли са от даден интервал `[p, q]`.