პროგრამირების აბსტრაქციები

სემინარის ამოცანები #10

კლიკების პოვნა გრაფში

ამ ამოცანის თითოეულ ნაწილში მოცემული გაქვთ გრაფი bool-ების grid-ის სახით, სადაც (i, j) პოზიციაზე წერია i-ური წვერო j-ურ წვეროს უკავშირდება თუ არა. გარანტირებულია რომ ეს grid-ი სიმეტრიული იქნება, ანუ (i, j)-ზე და (j, i)-ზე ერთიდაიგივე ეწერება.

ა) თქვენს ფუნქციას გადმოეცემა გრაფი და ვექტორი, რომელიც წვეროების ინდექსებს შეიცავს. თქვენი ამოცანაა გაარკვიოთ ამ ინდექსებიანი წვეროებით შემდგარი ქვეგრაფი არის თუ არა კლიკი (ანუ ამ ქვეგრაფში ყველა წვერო ყველას უკავშირდება თუ არა).

```
bool isConnectedToOthers(Grid<bool>& graph, int vertex, Vector<int> others) {
    foreach(int index in others) {
        if (!graph[vertex][index])
            return false;
    }
    return true;
}

bool isClique(Grid<bool>& graph, Vector<int> indices) {
    if (indices.size() <= 1)
        return true;

    int removedVertex = indices[indices.size() - 1];
    indices.remove(indices.size() - 1);

    return isConnectedToOthers(graph, removedVertex, indices)
        && isClique(graph, indices);
}
```

ბ) ამ ნაწილში ფუნქციას გადმოეცემა მხოლოდ გრაფი და თქვენ უნდა იპოვოთ რა მაქსიმალური ზომის კლიკის პოვნა შეგვიძლია ამ გრაფში. ამის გაკეთება შეგვიძლია შემდეგი რეკურსიული მეთოდით: დავიწყოთ ცარიელი სიმრავლიდან და მოცემული სიმრავლისთვის გავიგოთ რა მაქსიმალური კლიკი შეგვიძლია მივიღოთ ამ სიმრავლის გავრცობით. თუ სიმრავლე უკვე აღარ არის კლიკი, მისი გავრცობით ვერ მივიღებთ ვერანაირ კლიკს, მაგრამ თუ არის - ჩავამატოთ ახალი ელემენტი და ისევ ვცადოთ (რეკურსიულად). ბოლოს მიღებული კლიკებიდან მაქსიმალური ზომის უნდა შევარჩიოთ.

```
int maxSizeCliqueWithSubset(Grid<bool>& graph,
                            Vector<int>& chosenSet,
                            int firstFreeIndex) {
    if (firstFreeIndex == graph.numCols()) {
        if (isClique(graph, chosenSet))
            return chosenSet.size();
        else
            return -1;
    }

    int maxSizeWithoutFirst = maxSizeCliqueWithSubset(graph,
                                                      chosenSet,
                                                      firstFreeIndex + 1);

    chosenSet.add(firstFreeIndex);
    int maxSizeWithFirst = maxSizeCliqueWithSubset(graph,
                                                   chosenSet,
                                                   firstFreeIndex + 1);
    chosenSet.remove(chosenSet.size() - 1);

    return max(maxSizeWithFirst, maxSizeWithoutFirst);
}

int maxSizeClique(Grid<bool>& graph) {
    Vector<int> emptySet;
    return maxSizeCliqueWithSubset(graph, emptySet, 0);
}
```