2 октября.

Деревья.

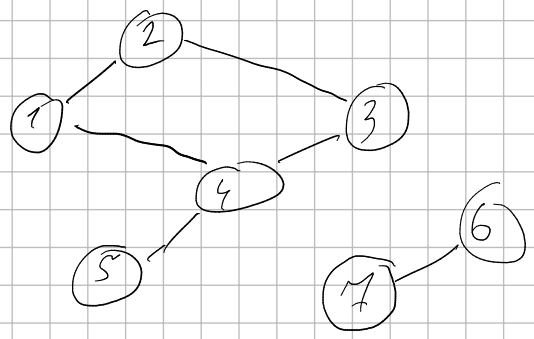
Термины теории графов:

V - множество вершин (vertex, pl. vertices)

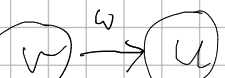
E - множество рёбер (edge, pl. edges)

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (1,4), (4,5), (3,4), (2,3)\}$



Графы могут быть ориентированными и неориентированными, взвешенными и невзвешенными



(v, u)

(v, u, w)

Связный граф - граф, в котором $\forall v \in V \quad \forall u \in V \quad \exists$ путь из v в u

Цикл - путь из u в u через другие вершины

Ациклический граф - граф, в котором невозможны циклы.

Простой путь - путь, в котором ни одна вершина не содержится дважды.

Реберно - простой путь - то же, но для рёбер.

Терминов много, с остальными разберёмся по ходу.

Список смежности - наиболее распространённый способ представления графа

1: [2, 3]

2: [1, 3]

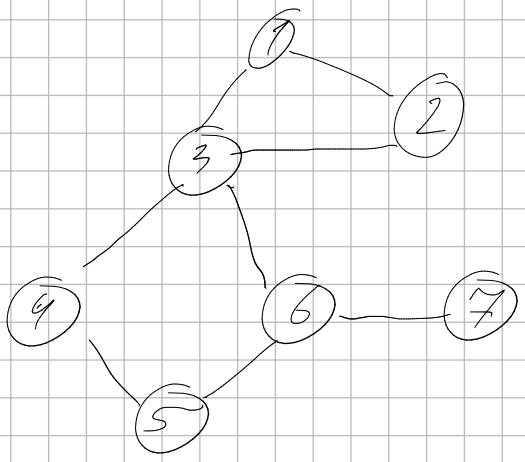
3: [1, 2, 4, 6]

$O(n)$

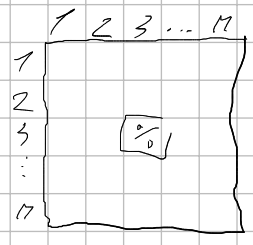
$O(n \cdot (n-1))$

$O(n+m) =$

$= O(v+e)$



Матрица смежности



$O(n^2)$

Удобно, когда нужно проверить существование вершины. Можно использовать на малых графах

Список рёбер

Храним рёбра в одномерном списке.

-1 (1, 2) (2, 1) (2, 3) (3, 2) (1, 3) (3, 1)

У нас получается что-то похожее на список

```
lst_v = [] // список рёбер O(n)
to_e = [] // куда оно ведёт O(m)
pr_e = [] // предыдущая вершина O(m)
for(int i = lst[v]; i != -1; i = pr_e[i]) {

}
```

Это имеет $O(m+n)$, но чуть меньшую по константе сложность.

В некоторых ситуациях это удобнее списка смежности, но обычно нет. Например, так лучше решать алгоритм потоков и некоторые другие продвинутые задачи.

Теперь к деревьям.

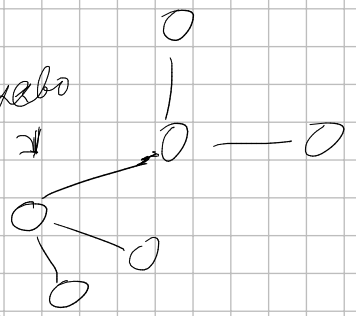
Дерево - это вид графа, который определяется по двум из трёх критериев:

1) Связность
2) Ациклическость
3) $m = n - 1$, где

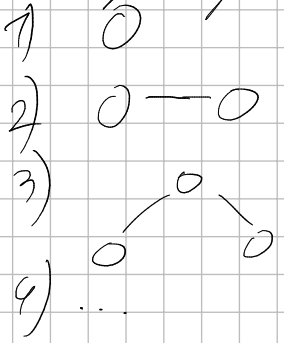
} из выполнения двух условий следует третье

m - количество рёбер

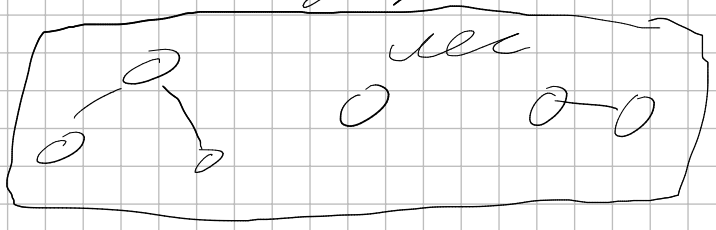
n - количество вершин



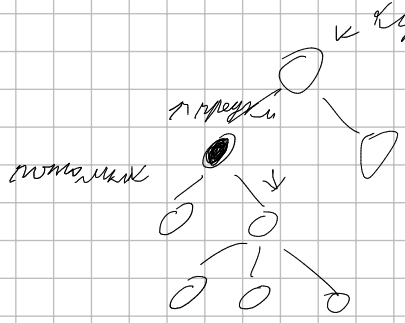
Другие примеры деревьев:



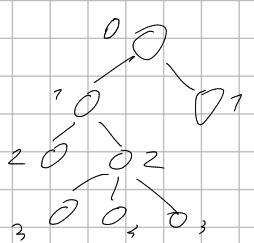
Лес - это когда каждая компонента связности является деревом:



В любом дереве можно выбрать корневую вершину и считать от неё детей, внуков и дальнейших потомков.



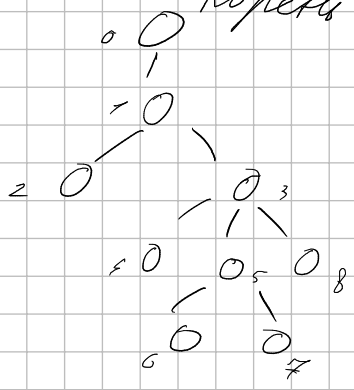
Глубина вершины - расстояние от вершины до корня



Удобная фича дерева: между любыми двумя вершинами $\exists!$ путь.

Высота вершины - то те, по сути вверх: расстояние от вершины до самого глубокого (дальнего) её потомка.

Самый популярный обход - обход в глубину.



Получается стек нашего пути: каждый раз закидываем в конец, куда нам идти дальше.

Извлекая же вершины из стека, мы их посещаем.

Таким образом мы пройдём все вершины. Это утверждение доказывается от обратного: мы придём к тому, что мы не посещали стартовую вершину.

Список родителей - ещё один способ представления дерева. Он неудобен для DFS, потому что тяжело определить детей, братьев и отца.

DFS - это depth-first search.

```cpp
#include <bits/stdc++.h>
using namespace std;
vector<vector<int>> g;
void dfs(int v) {
    // в случае с деревом нам не надо помнить список посещённых вершин,
    // но в общем случае это мастхев
    cout << v << endl;
    for(int u : g[v]) {
        if(u == v)
            continue;
        dfs(u); // рекурсия просимулирует стек,
                // так как на низком уровне мы как раз
                // и используем стек вызовов
    }
}
int main() {
    int n, m;
    cin >> n >> m;
    g.resize(m);
    // примечание: на входе данные в 1-индексации, а мы будем работать в 0-инд.
    for(int i= 0; i < n; ++i) {
        int v, u;
        cin >> v >> u;
        --v; --u; // переходим в 0-индексацию
        g[v].push_back(u);
        g[u].push_back(v);
    }

}
```

-Wl --stack=256000000 // переопределяем размер стека в g++

В MSVC в начале кода:
#pragma comment(linker, "/STACK:256000000")

Задача. Найти самый длинный вертикальный путь в дереве

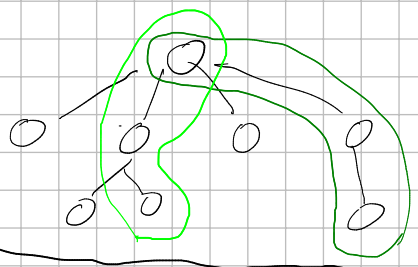
```cpp
#include <bits/stdc++.h>
using namespace std;
vector<vector<int>> g;
int dfs(int v) {
    int m = 0;
    for(int u : g[v]) {
        if(u == v)
            continue;
        m = max(m, dfs(u));
    }
    return m + 1;
}
int main() {
    int n, m;
    cin >> n >> m;
    g.resize(m);
    // примечание: на входе данные в 1-индексации, а мы будем работать в 0-инд.
    for(int i= 0; i < n; ++i) {
        int v, u;
        cin >> v >> u;
        --v; --u; // переходим в 0-индексацию
        g[v].push_back(u);
        g[u].push_back(v);
    }

}
```

Задача. Ищем корень с максимальной высотой.

ответ: $h_{u_1} + h_{u_2} + 2$

считаем по рёбрам.



```
x = -∞
y = -∞
for(int c: a) {
  if(c == x) {
    y = x
    x = c
  } else if(c > y) {
    y = c
  }
}
```

вставить это в DFS

```
nth_element(a.begin(),
            a.begin() + a.size()-1,
            a.end());
```

ставим $\ni - m$ th