HF08 - Utak

Kiíró: Németh Dániel István

Adatszerkezetek és Algoritmusok 2023 ősz

Beadási határidő: 2023 November 14, 23:59

Feladat

Minta megyében még nincsenek utak. Az útépítő társaság egyesével építi ki az utakat a falvak között. Azt, hogy melyik két település között épüljön út, nem tudjuk befolyásolni, viszont jelezni tudjuk ha már nem tartunk igényt több út építésére. A cél az, hogy "A" településről el tudjunk jutni úton "B" településre. Amint ez lehetségessé válik, nem kell több utat építeni. A cél egy olyan program megírása, amivel azonnal detektáljuk, ha már lehetséges "A" településről "B" településre eljutni.

A fentiek alapján tehát egy irányítatlan gráfot építünk fel élenként, és amint létezik az előre megadott két csúcs között út, ezt jelezni kell.

A kiadott skeleton kódban az alábbi osztályt kell megvalósítani: .

```
class GraphSolver{
public:
    GraphSolver(int start, int target) {
        //TODO
    }
    ~GraphSolver(){
        //TODO
    }
    GraphSolver(const GraphSolver& _other){
        // Copy konstruktor
        //TODO
    }
    GraphSolver& operator= (const GraphSolver& _other){
        // Assignment operator
        //TODO
    }
    GraphSolver& operator= (GraphSolver&& _other)  noexcept {
        // Move assignment operator
        //TODO
    }
    bool exitsAfterPathAdded(int node1, int node2) {
        //TODO
    }
};
```

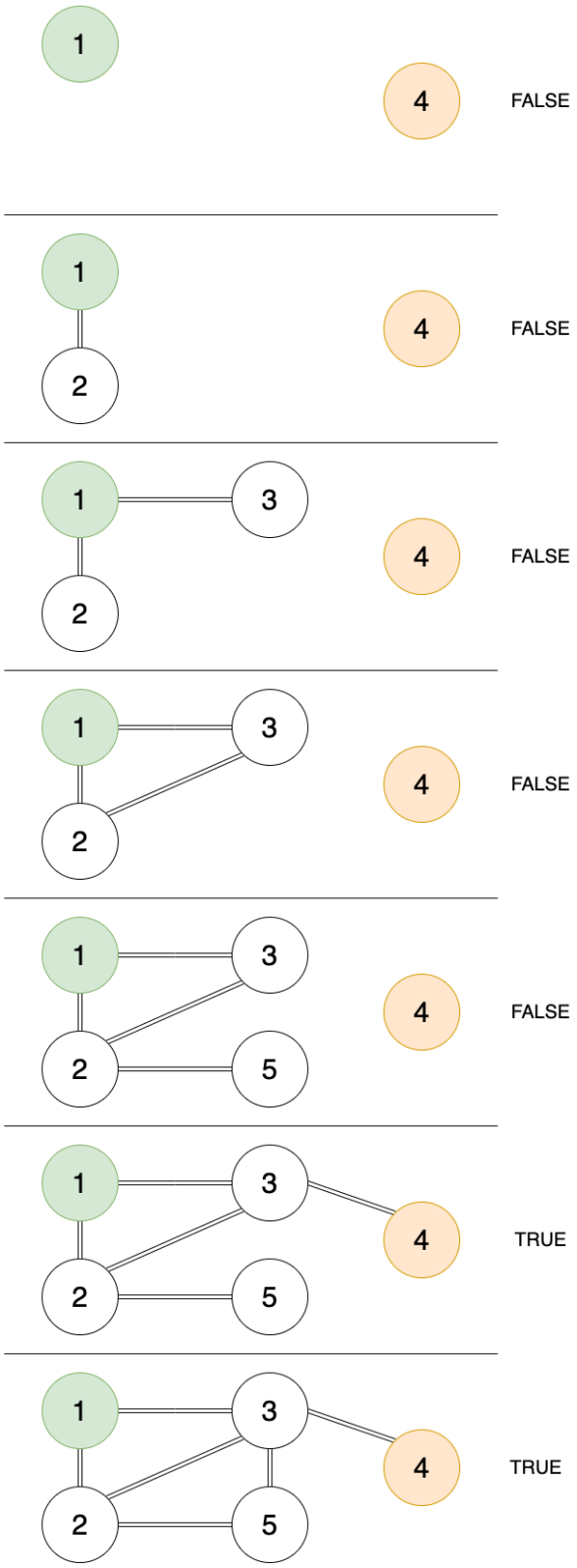
A `GraphSolver` osztályban szükséges eltárolni minden adatot, amire a megoldás során szükség lehet. A konstruktor két paramétere a gráf két csúcsa, amelyek között az utat keressük. Fontos, hogy a többi csúcs számai nem feltétlen e két szám közötti értékek. Tehát például ha `start=3;target=12;` akkor az 1, 2 és 13 és akár a 654321 is lehet valamelyik csúcs. Az egyetlen megkötés, hogy 1-nél kisebb egész szám nem jelölhet csúcsot, tehát 0 vagy -42 nem lehet csúcs. Az elején egyetlen él sincs a gráfban. Feltételezhetjük, hogy `start != target`.

A `bool exitsAfterPathAdded(int node1, int node2)` függvény hívásakor a gráfba `node1` és `node2` közötti utat szúrunk be. Feltételezhetjük, hogy `node1 != node2`. A beszúrás után a függvénynek

`true`-t kell visszaadnia, ha az eddig felépített gráf alapján a `start` és `target` csúcsok között létezik út. Ha nem, akkor a függvény visszatérési értéke `false`.

A konstruktorokat és operátorokat az előadáson és gyakorlaton tanultak alapján kell helyesen létrehozni. Mivel a feladat célja a tárgy során tanultak gyakorlása, így az `=default` megoldásokat nem fogadjuk el akkor sem, ha helyes működést eredményeznének.

Az alábbi ábrán az első teszteset lépései láthatók:



A megoldás működésének ellenőrzését néhány teszt is segíti, melyek a *main.cpp* fájlban találhatóak. Ezen teszek eredményei a console-ra íródnak ki a program futtatásakor, a console-ra kiírt pontszám a sikeresen lefutott tesztek számát jelzi, nem a feladatra kapott pontszámot. **FONTOS: A tesztek csak segítik a helyes működés ellenőrzését, de nem fednek le minden esetet, így akár helyesen lefutó tesztek estén is lehet hiba az algoritmusban, megoldásban. Ajánlott írni saját teszteket a működés további ellenőrzésére.**

A pontozásnál a kódminőség, és az algoritmus hatékonysága is számít a kód helyes működése mellett! Alkalmazzuk a kurzus során tanult módszereket és adatszerkezeteket! A kódba magyarázó kommentek is kerüljenek, amelyek leírják a kód működését! **A megoldáshoz szabad STL-t használni.** A megadott flagekkel (`-Werror -Wall -Wextra -pedantic`) nem forduló kód esetén a feladat nem értékelhető (0p).