Adatszerkezetek és Algoritmusok

Első házi feladat

Implementáljon egy gráf osztályt, majd a megvalósított tárolót felhasználva valósítsa meg a következő algoritmusokat: **gráf színezés m, vagy kevesebb színnel, Hamilton kör keresés, és Dijkstra algoritmus**.

A Gráf osztály formai követelményei:

1. OOP szemlélet alkalmazása (jól strukturált kód, láthatósági szintek alkalmazása . . .)
2. Külön header és source file a megfelelő kódrészekkel
3. A gráf megvalósítása láncolt ábrázolással (pointer aritmetika)

A Gráf osztály alkalmazási követelményei:

1. A gráfot lehessen *.txt* fájlból felépíteni:
   - a fájl első sora a csúcsok száma a gráfban, majd
   - soronként tartalmazza a csúcsok attribútumait: (csúcs id / szomszéd csúcsok: id_1 - súly_1, id_2 - súly_2, . . . , id_n - súly_n)
2. A gráf osztály valósítsa meg a következő függvényeket:
   - color(const unsigned int m)
   - hamilton()
   - dijkstra(const unsigned int source_id)
3. Segítség a gráf fölépítéséhez:
   - Először is, szükség lesz egy belső osztályra, amely egy enkapszulált `Node` osztály.
   - A `Node` osztály a gráf egy adott csúcsát hivatott reprezentálni, tehát kell lennie, egy *id* változónak (a csúcs egyedi azonosítója), illetve valamilyen tárolónak, amely tartalmazza a csúcsból kimutató éleket és a hozzájuk tartozó súlyokat. Ezekben a tárolókban pointerek legyenek, amelyek más `Node` osztály példányokra mutatnak (amik csúcsokat reprezentálnak).
   - Ezt a `Node` osztályt kell inicializálni a fájlból beolvasott adatok szerint, majd pointerek segítségével "összeláncolni" a különböző csúcsokat (`Node` osztályok).
   - Tehát végeredményben, annyi `Node` osztály legyen a `Graph` osztályban példányosítva, ahány csúcsunk van, és ezek tárolják el a kapcsolati információkat (élek, súlyok).
   - Ez a belső `Node` osztály megvalósíthat praktikus függvényeket is, például visszaadhatja az összes csúcsot, ahova él indul belőle, vagy dijkstránál tárolhatja az adott költségfüggvényeket is.

Gráf színezés m színnel: color(const unsigned int m)

m: a maximálisan felhasználható színek száma

Határozza meg, hogy az adott gráf kiszínezhető-e maximum **m** különböző szín felhasználásával, ha igen, akkor adja vissza az egyes csúcsokhoz rendelt színeket. (Két szomszédos csúcs nem lehet ugyanolyan színű!)

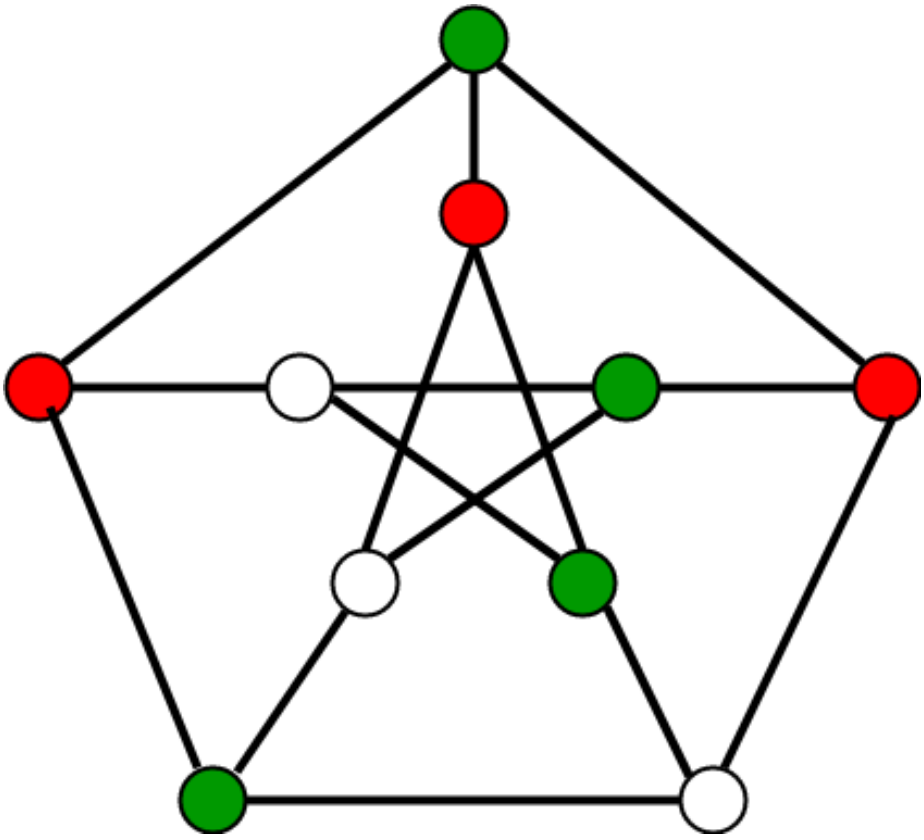

Figure 1: Graph coloring

Hamilton kör keresése

A Hamilton kör egy nem irányított gráfban: egy olyan útvonal, ami pontosan egyszer érint minden csúcsot. Határozza meg, hogy az adott gráf tartalmaz-e Hamilton kört, ha igen akkor adja vissza az útvonalat.

A Hamilton kör keresési algortmusnál megengedett kimerítő keresést használni, mivel NP-teljes probléma. Előfordulhatnak olyan bonyolult gráfok, amelyeknél

az algoritmus a “végtelenségig” fut, de a példákban ilyen nem fog szerepelni.

A megoldás az, hogy a kiinduló csúcsból az összes útvonalat megnézzük, ha van olyan, amely teljesíti a Hamilton kör definíciójából fakadó kitételeket, akkor azt visszaadjuk, ha nincs elkönyveljük, hogy a gráfban nincs ilyen kör és ezt tudatjuk a felhasználóval.

Dijkstra legrövidebb út keresés: dijkstra(const unsigned int source_id)

source_id: kiinduló csúcs azonosítója

Adott egy gráf és egy kiinduló csúcspont a gráfban. Keresse meg a legrövidebb utat a kiinduló csúcsból az összes csúcsba. A megoldás során vissza kell térni egy listával, amely tartalmazza, hogy az egyes csúcsokba mekkora minimális költséggel tudunk eljutni.

További követelmények:

1. Az algoritmusok megoldása során, kimerítő keresést nem lehet használni, kivéve a Hamilton kör esetében!
2. A feladatot egyénileg kell megoldani! Internetről vagy hallgató társról másolt kód esetén nulla pont adható!
3. Az algoritmus tervezéshez, implementáláshoz hozzá tartozik az adott algoritmus matematikai megértése is. Az algoritmusok implementálásához jelölje meg a felhasznált hiteles forrást! (Hiteles forrás lehet: könyv, tudományos folyóirat, cikk, vagy olyan weboldal, ahol ellenőrízhető a szerző és a tartalom hitelessége). (A wikipédia, Stack Overflow és egyéb fórumok nem hitelesek!)
4. A feladatok megoldását Eclipse IDE segítségével, és MinGW fordítóval végezze el. Figyeljen a megfelelő kapcsolók (Wall, Wextra, Pedantic, Werror) és a C++11 dialect beállítására. Továbbá ne felejtse el a workspace kódolását UTF8-ra változtatni, ezt a “Preferences” fül felugróablakánál a workspace kulcsszóval érheti el. (Nézze át figyelmesen az első órai diát.)
5. A projeket az első órán bevezetett konvenciók alapján kell beadni.
   - A projekt neve: <digitus>_hf_<hf_sorszáma>.
   - Beadás az SVN repositoryba: https://repo.itk.ppke.hu/adatszerk/<digitus_név>.
   - A házit ezen belül tegye a megfelelő mappába!
   - Beadni a `.cproject`, `.project`, és `src` mappákat kell!
6. A pontozás menete feladatokra lebontva a következő:
   - A gráf fölépítése fájlból, belső `Node` osztály használatával: 15 pont.

- Dijkstra algoritmus helyes implementálása (a saját struktúránkra): 15 pont.
- Coloring algoritmus megvalósítása: 10 pont.
- Hamilton kör implementálása: 10 pont.