PRÀCTICA 4: Camins hamiltonians i eulerians i coloració

En aquesta pràctica cal modificar el projecte anterior en un projecte nou `GPrc4`. Consta de diverses parts. La primera part està dedicada a la cerca de camins hamiltonians, la segona part, a la coloració de vèrtexs. Es proposa la utilització de mètodes de backtracking en ambdues parts. Després d'una part dedicada a les llistes d'adjacències amb indentificació d'arestes, segueixen dues parts dedicades a la coloració d'arestes i als camins eulerians.

Cicles hamiltonians i problema del viatjant

El mètode de Robert i Flores per a la cerca de camins i circuits hamiltonians fa una recerca exhaustiva de totes les possibilitats.

La idea bàsica consisteix a sortir d'un vèrtex i anar traçant un camí, triant en cada pas un vèrtex de la llista d'adjacències que no estigui encara en el camí fins que, eventualment, s'hagi passat per tots els vèrtexs. En el moment que no es pugui continuar el camí des d'un vèrtex v perquè tots els vèrtexs adjacents ja són al camí fet, cal fer backtracking. Això és, cal treure el vèrtex en qüestió del camí i anar al vèrtex precedent i triar, si és possible, un altre vèrtex adjacent i així successivament. Quan tots els vèrtexs són al camí, es té un camí hamiltonià. En el cas que el darrer vèrtex del camí sigui adjacent al primer, es té un cicle hamiltonià. En cas contrari, si es vol trobar un cicle hamiltonià, cal fer també backtracking.

Un cop trobat un cicle hamiltonià, si es volen trobar tots, es pot continuar el mateix mètode fent novament backtracking fins que s'han exhaurit totes les possibilitats i tocaria fer backtracking del vèrtex de partida.

En el cas de grafs ponderats, el problema de viatjant de comerç *TSP, Travelling Salesman Problem,* consisteix en la cerca d'un cicle hamiltonià minimal.

En la programació del mètode caldria utilitzar:

- un vector `HCv` on es van posant (i treient) els vèrtexs successius del cicle en formació,
- un vector `HCl` de booleans, inicialitzat a `false` que ens permeti marcar els vèrtexs que són al cicle hamiltonià en formació amb `true`,
- un vector `ind` que ens indiqui, en la llista d'adjacències de cada vèrtex, l'índex d'incidència a tenir en compte en la propera visita al vèrtex, inicialitzat a l'índex 0 per a tots els vèrtexs. Aquest vector d'índexs permet portar el control de totes les possibilitats sense deixar-se'n cap.

Exercici 18 Modifica el programa amb un altre mòdul `graphHC.cpp` que contingui funcions `HamiltonianCycle, HamiltonianCycles` que implementin el mètode de backtracking de Robert i Flores per a la cerca de cicles hamiltonians, un o tots. Aquestes funcions haurien de retornar el nombre de cicles hamiltonians trobats i escriure'ls com a successió de vèrtexs a un $stream$ de sortida.

Implementa de nou un programa principal per tal que escrigui aquesta informació dels cicles hamiltonians trobats en els grafs $K_n, K_{n_1,n_2}, C_n, S_n, W_n$ en els fitxers `Kn.out, Kn1_n2.out, Cn.out, Sn.out, Wn.out`, per a diferents valors de n, n_1, n_2 permesos per un temps d'execució raonable.

Continua el programa principal per tal que trobi el nombre de cicles hamiltonians en els grafs Kt_{n_1,n_2} del moviment del cavall en taulers de diferents mides n_1xn_2 i els escrigui en el fitxer `Ktn1_n2.out`.

Exercici 19 Estén el programa amb un altre mòdul `wgraphTSP.cpp` que completi la implementació del mètode de backtracking anterior per a resoldre el problema del viatjant en grafs ponderats. La funció `TravellingSalesmanProblem` implementada hauria de retornar el pes del cicle de pes mínim trobat i escriure'n la seqüència de vèrtexs en un $stream$ de sortida.

Continua el programa principal per tal que escrigui aquesta informació del cicle hamiltonià minimal trobat en els grafs $WK_n, WK_{n_1,n_2}, WC_n, WS_n, WW_n$ en `WKn.out, WKn1_n2.out, WCn.out, WSn.out, WWn.out`, per a diferents valors de n, n_1, n_2 permesos per un temps d'execució raonable.

Exercici 20 Considera la taula següent de distàncies (en centenars de milles) entre Londres (L), Mèxic DF (Mx), Nova York (NY), París (Pa), Pekin (Pe) i Tòquio (T):

	L	Mx	NY	Pa	Pe	T
L		56	35	2	51	60
Mx			21	57	78	70
NY				36	68	68
Pa					51	61
Pe						13

Usa el programa anterior per tal que resolgui el problema del viatjant entre totes aquestes ciutats i n'escrigui el resultat a `ciutats.out`.

Exercici 21 Completa el programa escrivint en `D.out` i `I.out` tots els cicles hamiltonians i la solució del problema del viatjant corresponents respectivament al dodecàedre ponderat i a l'icosàedre ponderat de la pràctica anterior.

Coloració de vèrtexs

Una coloració de vèrtexs consisteix a assignar colors (enters positius) als vèrtexs amb la restricció que dos vèrtexs adjacents no tinguin assignat el mateix color.

Una coloració d'un graf es pot fer utilitzant el mètode de Brélaz consistent en la coloració seqüencial dels vèrtexs. El color assignat a cada vèrtex és el color més petit permès pels seus adjacents ja acolorits: el color més baix que manca al conjunt de colors dels adjacents.

Si es desitja una coloració minimal, caldrà anar rebaixar successivament el nombre de colors utilitzats en una coloració inicial, que podria ser la de Brélaz. En cada etapa d'aquest procés, es procura tornar a colorar el graf evitant el color màxim, provant totes les possibilitats amb un mètode de backtracking. Es va repetint aquesta etapa mentre és possible fer una nova coloració amb un color menys (el més gran). Quan ja no és possible, l'anterior coloració d'aquest procés de recoloració és la coloració minimal resultant.

Per tal de fer una nova coloració amb un color menys es fa un procés de backtracking que s'inicia en el primer vèrtex `cnv` amb el color màxim `cn` i es torna a colorar el graf cada cop que es troba un nou vèrtex de backtracking `cnv`.

En cada pas del procés, es considera el vèrtex `cnav` anterior a `cnv`. Si eventualment s'arriba al vèrtex `cnv=0`, `cnav` no existeix i no es pot millorar la coloració: la coloració anterior es dóna com a coloració minimal i s'acaba el procés. Es colora el vèrtex `cnav` de nou amb un color més gran que el que té. Si cal el color `cn` en aquesta nova coloració de `cnav`, es fa backtracking sobre `cnav`: es pren `cnav` com a nou `cnv` i es repeteix el procediment. Si no cal el color `cn` per a `cnav`, es decoloren els vèrtexs posteriors a `cnav` i es tornen a colorar de nou. Si algun vèrtex d'aquests necessita el color `cn`, es pren com a nou vèrtex de backtracking `cnv` i es repeteix el procediment. Si no, s'ha rebaixat el nombre de colors necessaris de `cn` a `cn-1` que passa a ser el nou `cn`.

Exercici 22 Estén el programa amb un altre mòdul `graphCol.cpp` que contingui funcions que implementin mètodes de coloració de vèrtexs:

- Mètode de coloració de Brélaz (heurística)

- Mètode de coloració minimal per backtracking a partir de la coloració de Brélaz.

Continua el programa principal per tal que escrigui aquesta informació de coloració de vèrtexs trobada per als grafs $K_n, K_{n_1,n_2}, C_n, S_n, , W_n, Kt_{n_1,n_2}$ en els fitxers `Kn.out, Kn1_n2.out, Cn.out, Sn.out, Wn.out, Ktn1_n2.out`, per a diferents valors de n, n_1, n_2.

Exercici 23 Completa el programa escrivint una coloració minimal de dodecàedre i de l'iscosàedre en `D.out` i I.out, respctivament.

Exercici 24 Continua el programa principal aplicant-lo a la coloració minimal de les comarques catalanes.

Llistes d'adjacències amb identificació d'arestes

Es proposa l'arxiu de capçalera *graphe.h* del tipus següent:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <algorithm>
#include "graph.h"
 using namespace std;
 typedef pair<vertex,index> vip;
 typedef map<vip,edge> edges;
 void graphe_complete( graph &Kn, edges &KnE, index n );
 void graphe_cycle( graph &Cn, edges &CnE, index n );
 void graphe_bipartite_complete( graph &Kn1n2, edges &Kn1n2E, index n1, index n2 );
 void graphe_star( graph &Sn, edges &SnE, index n );
 void graphe_wheel( graph &Wn, edges &WnE, index n );
 void line_graphe( const graph &G, edges &GE, graph &LG, edges &LGE );
 void graphe_read( graph &G, edges &GE, string fname );
 void graphe_write( graph &G, edges &GE, ofstream& fout );
 color MinimalVertexColoring( const graph& G, vector<color>& Gc, ofstream& fout );
 color MinimalEdgeColoring( const graph& G, edges &GE, ofstream& fout );
 bool Hierholzer( const graph &G, edges &GE, ofstream &fout );
```

El tipus `vip` (vertex - index pair) consisteix en una parella vèrtex - índex d'incidència. Cada aresta del graf consta en dues parelles de tipus `vip`. Es fa servir `map<vip,edge>`, tipificada com a `edges` per a fer-ne la identificació com a única aresta: a la parella vèrtex - índex i a la parella vèrtex - índex simètrica se li assigna la mateixa aresta.

Exercici 25 Modifica el mòdul *graph.cpp* en un nou mòdul *graphe.cpp* per tal de poder treballar amb llistes d'adjacències amb parelles de tipus `vip` identificades en `edges`.

Adequa el programa principal per tal que construeixi les llistes d'adjacències amb identificacions en arestes per als grafs $K_n, K_{n_1,n_2}, C_n, S_n, W_n$ i dels grafs E_0, E_1, E_2, E_3 corresponents als fitxers `E0.in`, `E1.in`, `E2.in`, `E3.in` (inclosos en el fitxer *GPrc4.zip* adjunt a la pràctica), i escrigui la informació en els fitxers `Kn.out`, `Kn1n2.out`, `Cn.out`, `Sn.out`, `Wn.out`, `E0.out`, `E1.out`, `E2.out`,`E3.out`, respectivament

---

Exercici 26 Amplia el mòdul *graphe.cpp* amb la funció `line_graphe` que trobi el graf de línia com a llistes d'adjacències LG amb parelles identificades en arestes LGE, d'un graf donat com a llistes d'adjacències G amb parelles identificades en arestes GE.

Coloració minimal d'arestes

Es volen trobar coloracions de les arestes d'un graf amb el mínim nombre de colors, de forma que no s'assigni el mateix color a dues arestes que tinguin un vèrtex en comú.

Exercici 27 Estén el mòdul *graphCol.cpp* amb la funció `MinimalEdgeColoring` que implementi la coloració minimal d'arestes, aplicant el mètode de coloració minimal de vèrtexs `MinimalVertexColoring` al seu graf de línia. El resultat de les coloracions intermèdies s'ha d'escriure en l'stream de sortida `fout`. Per a la coloració final, cal escriure-hi també els grups d'arestes de cada color.

Continua el programa principal per tal que escrigui aquesta informació de coloració d'arestes per a grafs K_n, K_{n_1,n_2}, C_n (amb temps d'execució raonables) i per als grafs E_0, E_1, E_2, E_3 i escrigui la informació en els fitxers `Kn.out`, `Kn1n2.out`, `Cn.out`, `E0.out`, `E1.out`, `E2.out`,`E3.out`, respectivament

---

Exercici 28 Aplica el mètode per tal que escrigui els partits que cal jugar en les diverses jornades d'una lliga entre n equips: $n = 4, 6, 8, 10, 12, 14, ...$ fitxers `lliga*.out`.

Circuits eulerians

Estén el projecte amb un altre mòdul *graphEC.cpp* que contingui funcions que permetin implementar el mètode de Hierholzer per a la cerca de circuits eulerians. La funció `Hierholzer` hauria de fer prèviament tests de connectivitat i de paritat de graus per tal d'assegurar que sigui eulerià. En el cas que ho sigui, la funció hauria de trobar-lo utilitzant el mètode de Hierholzer, escriu-re'l a un *stream* i retornar `true`. En cas contrari, hauria de retornar `false`.

El mètode de Hierholzer es basa en la descomposició en cicles de qualsevol graf eulerià. El circuit eulerià surt de la concatenació adequada d'aquests cicles disjunts per arestes.

Els vèrtexs del circuit eulerià es van emmagatzememant en el vector `ECv`, les arestes són inicialment etiquetades a `Gl` com a `true` i s'etiqueten com a `false` quan entren a formar part del circuit. Els graus del graf residual es van actualitzant al vector `rdeg`: cada cop que una aresta entra a formar part del circuit, els graus residuals dels vèrtexs extrems es rebaixen en 1. El procediment comença trobant un primer cicle i guardant-lo en un vector de vèrtexs `ECv`, les arestes del cicle trobat són etiquetades `true` en `Gl` i els graus rebaixats a `rdeg`. En cada etapa següent, cal trobar un vèrtex `v0` d'`ECv` amb `rdeg` positiu, fer una rotació del circuit trobat fins al moment per tal que `v0` aparegui al començament, trobar un cicle a partir de `v0`, com s'ha fet amb el primer cicle, i concatenar-lo al final del circuit. S'atura el procès quan el circuit és complet.

Exercici 29 Continua el programa principal per tal que escrigui en els fitxers `*.out` corresponents el resultat de l'aplicació de la funció `Hierholzer` als grafs de l'exercici 25.

Exercici 30 [**Opcional**] Completa el mòdul nou amb una funció `ChinessePostmanProblem` que resolgui el problema del cartex xinès per a grafs ponderats, no necessàriament eulerians. Caldrà per això:

- trobar els camins mínims entre parelles de vèrtexs de grau senar;
- construir un graf complet entre els vèrtexs de grau senar, ponderat segon la distància trobada entre ells;
- trobar un aparellament mínim en aquest graf complet;
- construir el graf euleritzat que és l'extensió del graf inicial que es troba duplicant les arestes dels camins mínims corresponents a l'aparellament trobat;
- trobar finalment una solució del problema del carter xinès: un circuit eulerià del graf euleritzat via el mètode de Hierholzer.

Completa el programa principal, resolent el problema del carter xinès per a tots els grafs anteriors que no siguin eulerians, per als quals la funció `Hierholzer` retornaria `false`.

Comprimeix els fitxers del programa que has completat (*h, *.cpp) amb els de dades (*.in) i amb els de solució/projecte (*.sln, *.vcxproj, *vcxproj.filters), en un fitxer `GPrc24_CognomsNom.zip`, on consti el teu nom en `Nom` i els teus cognoms en `Cognoms`, i penja'l en el Campus Virtual en la tasca corresponent a la Pràctica 4. Els fitxers (*.out) i les altres subcarpetes no s'hi haurien d'incloure.