PRÀCTICA 2: Representació en llistes d'adjacències. Recorreguts jeràrquics i camins minimals

El codi bàsic per a les pràctiques 2 i 3 que consta en `GPrc23.zip` conté fitxers de capçalera (*.h), mòduls del programa (*.cpp), i fitxers de dades de grafs `graph?.in` i de grafs ponderats `wgraph?.in`. Pots obrir-ho amb Visual Studio C++ clicant el fitxer de solució `GPrc23.sln`.

El mòdul `graph.cpp` conté:

- La funció que emplena les llistes d'adjacències d'un graf complet K_n amb el nombre de vèrtexs indicat.
- La funció que emplena les llistes d'adjacències d'un graf a partir de les dades següents d'un graf en el fitxer indicat, amb el format següent:
  - nombre de vèrtexs i arestes
  - parelles de vèrtexs que conformen les arestes
- La funció que envia a un *stream* de sortida informació diversa a partir de les llistes d'adjacències d'un graf:
  - nombre de vèrtexs i arestes
  - llistes d'adjacències dels vèrtexs
  - parelles de vèrtexs conformant les arestes

El mòdul `wgraph.cpp` conté:

- La funció que emplena les llistes d'adjacències ponderades d'un graf complet WK_n amb el nombre de vèrtexs indicat ponderat amb pesos aleatoris fins a un pes màxim.
- La funció que emplena les llistes d'adjacències ponderades d'un graf a partir de les dades següents d'un graf en un fitxer amb el format següent:
  - nombre de vèrtexs i arestes
  - parelles de vèrtexs que conformen les arestes amb el pes corresponent
- La funció que envia a un *stream* de sortida informació diversa a partir de les llistes d'adjacències ponderades d'un graf:
  - nombre de vèrtexs i arestes
  - llistes d'adjacències ponderades dels vèrtexs
  - parelles de vèrtexs conformant les arestes amb el pes corresponent

El mòdul `graphTS.cpp` conté funcions de cerca jeràrquica (Tree Search) a partir de les llistes d'adjacències:

- la funció `DFS` fa la cerca en profunditat (Depth First Search) i retorna el nombre de components del graf,
- la funció `BFS` fa la cerca en extensió (Breadth First Search) i retorna el nombre de components del graf,
- la funció `Dijkstra` troba la distància de tots els vèrtexs a un vèrtex donat, entenent la distància entre dos vèrtexs com la longitud mínima, amb nombre d'arestes, dels camins que els uneixen.

Els fitxers de capçalera `graph.h`, `wgraph.h` contenen els prototips de les diverses funcions i s'inclouen en els mòduls corresponents.

El programa principal que consta en el mòdul `GPrc23.cpp`:

- Construeix en `K10` i `G0` les llistes d'adjacències de graf complet de 10 vèrtexs K_{10} i del graf en `graph0.in`. A continuació, escriu la informació corresponent en `K10.out` i `graph0.out`, respectivament, incloent-hi el nombre de components trobats via DFS i BFS i les distàncies (en nombre d'arestes) al vèrtex 0.
- Construeix en `WK10` i `WG0` les llistes d'adjacència ponderades d'un graf complet de 10 vèrtexs ponderat amb pesos aleatoris fins a un pes màxim de 9, WK_{10} i del graf ponderat corresponent al fitxer `wgraph0.in` (els pesos apareixen en la tercera columna de les arestes). A continuació, escriu la informació corresponent en `WK10.out` i `wgraph0.out`, respectivament.

Exercici 8 Amplia el programa per tal que tracti els grafs bipartits complets K_{n_1,n_2}, els grafs cicles C_n, els grafs estrelles S_n i els grafs rodes W_n, de la mateixa manera que els grafs complets K_n, sense i amb ponderació aleatòria.

Continua el programa principal:

- Construint en `(K6_4, K4_6), C10, S10, W10` les llistes d'adjacències de ($K_{6,4}$, $K_{4,6}$), C_{10}, S_{10}, W_{10}, i escrivint-ne la informació en `K6_4.out`, `C10.out`, `S10.out`, `W10.out`.
- Construint en `G1, G2, G3` les listes d'adjacències dels grafs en `graph1.in`, `graph2.in`, `graph3.in` i escrivint-ne la informació en `graph1.out`, `graph2.out`, `graph3.out`.
- Construint en `(WK6_4, WK4_6), WC10, WS10, WW10` les llistes adjacències ponderades dels grafs amb pesos aleatoris ($WK_{6,4}, WK_{4,6}$), WC_{10}, WS_{10}, WW_{10} i escrivint-ne la informació en `WK6_4.out, WC10.out, WS10.out, WW10.out`.
- Construint en `WG1, WG2, WG3` les listes d'adjacències ponderades dels grafs ponderats en `wgraph1.in, wgraph2.in, wgraph3.in` i escrivint-ne la informació en `wgraph1.out`, `wgraph2.out`, `wgraph3.out`.

Exercici 9 Completa el programa amb funcions `DFS_Trees` i `BFS_Trees` que guardin els arbres DFS i BFS corresponents als recorreguts jeràrquics dels diferents components i enviïn la informació següent a un *stream* de sortida:

- de cada vèrtex: l'identificador, l'índex en el recorregut, l'identificador del seu predecessor en l'arbre, si escau, i la seva profunditat respecte a l'arrel;
- de cada aresta: els identificadors dels vèrtexs extrems i si és o no dels arbres expansius.

Continua el programa principal per tal que, a partir de `K10`, (`K6_4`, `K4_6`), `C10`, `S10`, `W10`, `G0`, `G1`, `G2`, `G3`, escrigui la informació anterior en `K10.out`, `K6_4.out`, `C10.out`, `S10.out`, `W10.out`, `graph0.out`, `graph1.out`, `graph2.out`, `graph3.out`.

Compara les distàncies al vèrtex 0 trobades pel mètode de Dijkstra amb els nivell trobats amb els mètodes BFS i DFS en els arbres amb arrel en el vèrtex 0.

Considerem ara els grafs corresponents al moviment de les peces d'un tauler d'escacs n_1xn_2. Els vèrtexs del graf són les caselles i, per a cada peça, les caselles adjacents a cada casella són aquelles a les quals es pot anar amb un moviment qualsevol de la peça.

Exercici 10 Completa el mòdul *graph.cpp* amb funcions que construeixen els grafs corresponents al moviment de les peces següents: rei (*king*) $Kg_{n1,n2}$, torre (*rook*) $Rk_{n1,n2}$, alfil (*bishop*) $Bp_{n1,n2}$ i cavall (*kight*) Kt_{n_1,n_2}, i amb funcions que trobin les distàncies d'una casella a totes les altres.

Continua el programa principal fent un tractament similar del grafs $Kg_{6,6}$, $Rk_{6,6}$, $Bp_{6,6}$ i $Kt_{6,6}$ que l'exercici anterior i trobant el nombre mínim de salts que calen per arribar des de la casella $(2, 0)$ a totes les altres i escrivint els resultats en `Kg6_6.out`, `Rk6_6.out`, `Bp6_6.out` i `Kt6_6.out`.

Les caselles s'haurien d'escriure en coordenades (i_1, i_2) i les informacions relatives als graus de les caselles i de distàncies a una casella donada dels vèrtexs en matrius n_1xn_2.

A tall d'exemple, les matrius de graus i de distàncies des de la casella $(2, 0)$ del graf associat als moviments del rei en un tauler 6x6 s'escriurien així:

$$\begin{matrix} 3 & 5 & 5 & 5 & 5 & 3 \\ 5 & 8 & 8 & 8 & 8 & 5 \\ 5 & 8 & 8 & 8 & 8 & 5 \\ 5 & 8 & 8 & 8 & 8 & 5 \\ 5 & 8 & 8 & 8 & 8 & 5 \\ 3 & 5 & 5 & 5 & 5 & 3 \end{matrix} \qquad \begin{matrix} 2 & 2 & 2 & 3 & 4 & 5 \\ 1 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 2 & 3 & 4 & 5 \\ 3 & 3 & 3 & 3 & 4 & 5 \end{matrix}$$

Exercici 11 Estén el programa amb un altre mòdul `wgraphMP.cpp` que contingui funcions que implementin el mètode de Dijkstra de cerca de camins minimals (Minimal Paths) en grafs ponderats i enviïn els resultats a un *stream* de sortida.

Més concretament, es demanen funcions que retornin:

- la distància entre dos vèrtexs, entesa ara com el pes mínim dels camins que els uneixen atenent als pesos de les arestes,
- la distància màxima entre un vèrtex i tots els altres,
- el diàmetre: la distància màxima entre qualsevol parella de vèrtexs,

i funcions que escriguin en l'*stream*:

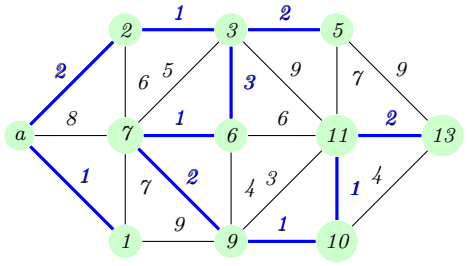
- les distàncies d'un vèrtex donat a tots els altres,
- l'arbre de camins minimals trobat des d'un vèrtex donat i els camins des de tots els vèrtexs accessibles a ell.

Continua el programa principal per tal que, a partir de `WK10`, (`WK6_4`, `WK4_6`), `WC10`, `WS10`, `WW10`, `WG0`, `WG1`, `WG2`, `WG3`, escrigui en `WK10.out`, `WK6_4.out`, `WC10.out` `WS10.out`, `WW10.out`, `wgraph0.out`, `wgraph1.out`, `wgraph2.out`, `wgraph3.out`, la informació següent:

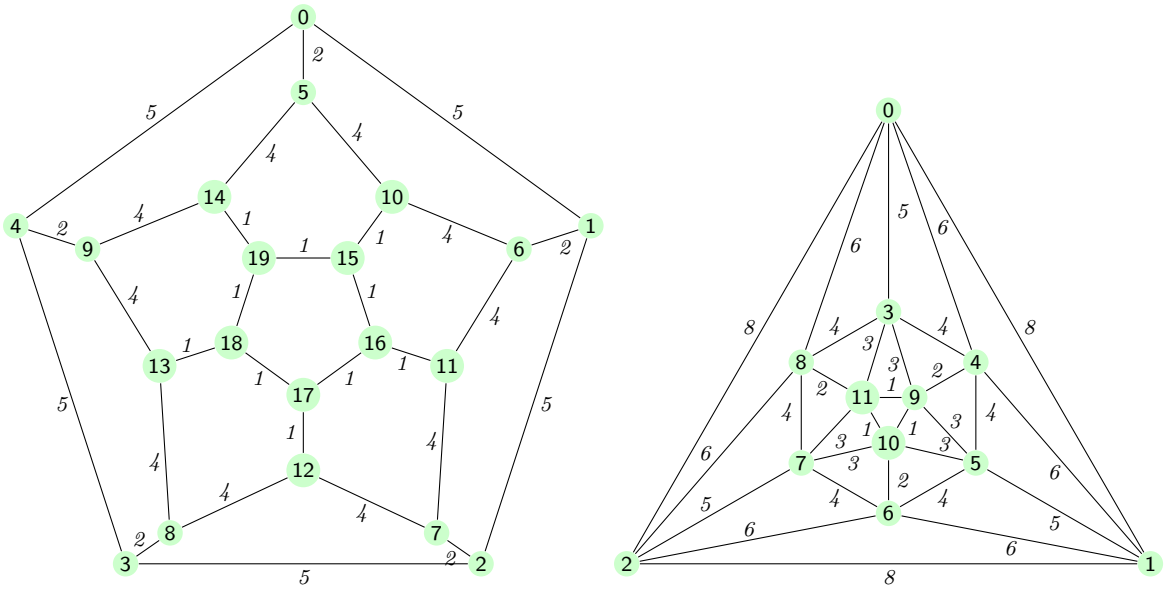
- la distància entre el vèrtex 0 i el darrer,
- la distància màxima entre el vèrtex 0 i tots els altres,
- les distàncies del vèrtex 0 a tots els vèrtexs accessibles, indicant els que ho són,
- el diàmetre,
- els camins de tots els vèrtexs accessibles al vèrtex 0.

Considerem ara el graf ponderat de l'Exemple 3.2.2 dels apunts de teoria.

Exercici 12 Continua el programa comprovant els camins minimals des del vèrtex a, assenyalats en blau, i les distàncies a a que consten en cada vèrtex:



Exercici 13 Completa el programa, trobant camins minimals a partir del vèrtex 0 a tots els altres vèrtexs en el grafs ponderats següents en el dodecàedre i en l'icosàedre:



Aquesta pràctica es lliura conjuntament amb la pràctica 3.

PRÀCTICA 3: Arbres minimals

Continua el projecte `GPrc23` amb un mòdul de cerca d'arbres minimals i la seva aplicació.

L'objectiu d'aquest mòdul és trobar arbres minimals en tots els components d'un graf utilitzant dos mètodes diferents:

El *mètode de Kruskal* troba les arestes que conformen els arbres minimals a partir d'una aresta minimal i, a cada pas, troba una aresta minimal entre les que no produeixen cicles. Quan no existeix, s'atura el procés. Les arestes considerades poden ser de components diferents del graf i acaben formant arbres minimals en cada component.

El *mètode de Prim* troba les arestes dels arbres minimals, component a component. Parteix d'una aresta minimal inicial que no estigui en cap dels components considerats per tal de construir un arbre minimal del component a què pertany: a cada pas, parteix d'un arbre en formació i li afegeix, com a branca nova, una aresta minimal que surti de l'arbre i no formi cicle. Quan aquesta aresta no existeix, les arestes afegides conformen un arbre minimal del component.

Exercici 14 Estén el programa amb un altre mòdul `wgraphMT.cpp` que contingui funcions que implementin els mètodes de Kruskal i de Prim de cerca d'arbres minimals (Minimal Trees) en grafs ponderats. Aquestes funcions haurien de retornar el pes total dels arbres minimals trobats i enviar la informació següent a un *stream* de sortida:

- les arestes que conformen els arbres minimals amb el seu pes;
- per a cada component, els vèrtexs que hi pertanyen, el pes de l'arbre minimal trobat en la component i, per al mètode de Prim, les arestes que el conformen amb el seu pes.

Continua el programa principal per tal que escrigui aquesta informació dels arbres minimals trobats en els grafs `WK10`, (`WK6_4`, `WK4_6`), `WC10`, `WS10`, `WW10`, `WG0`, `WG1`, `WG2`, `WG2` en els fitxers `WK10.out`, `WK6_4.out`, `WC10.out`, `WS10.out`, `WW10.out`, `wgraph0.out`, `wgraph1.out`, `wgraph2.out`, `wgraph3.out`, respectivament.

Exercici 15 Considera la taula següent de distàncies (en centerars de milles) entre Londres (L), Mèxic DF (Mx), Nova York (NY), París (Pa), Pequín (Pe) i Tòquio (T):

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

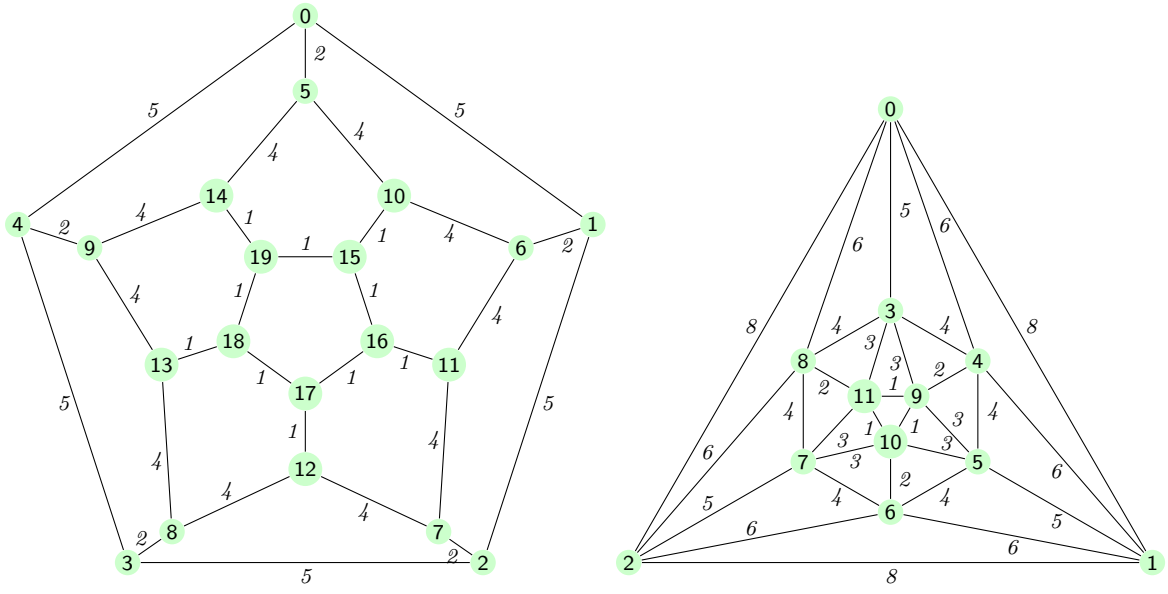
Usa el programa anterior per tal que trobi un arbre minimal de connexió entre totes aquestes ciutats i n'escrigui la informació a `cities1.out`.

Exercici 16 Considera la taula següent de distàncies (en kilòmetres) entre les ciutats xineses de Beijing (B), Chongquing (C), Guangdong (G), Nanjing (N), Shanghai (S), Tianjin (T) i Wuhan (W), i l'estació de generació hidroelèctrica de les Tres Gorges situada a Yichang (Y):

	C	G	N	S	T	W	Y
B	1456	1892	901	1068	111	1056	1116
C		968	1199	1430	1442	650	463
G			1133	1196	1820	836	866
N				266	800	459	626
S					960	681	962
T						988	1080
W							285

Usa el programa anterior per tal que trobi la forma de connexió elèctrica de mínima distància entre totes aquestes ciutats i n'escrigui la informació a `cities2.out`.

Exercici 17 Completa el programa escrivint la informació corresponent al dodecàedre i a l'icosàedre ponderats de la pràctica anterior en `WD.out` i `WI.out`, respectivament:



Comprimeix els fitxers del programa de les pràctiques 2 i 3 que has completat (*h, *.cpp) amb els de dades (*.in) i amb els de solució/projecte (*.sln, *.vcxproj, *vcxproj.filters), en un fitxer `GPrc23_CognomsNom.zip`, on constin el teu nom en `Nom` i els teus cognoms en `Cognoms`, i penja'l en el Campus Virtual en la tasca corresponent a les pràctiques 2 i 3. Els fitxers (*.out) i les altres subcarpetes no s'hi haurien d'incloure. L'avaluació d'aquestes pràctiques es farà per separat tot i lliurar-se conjuntament.