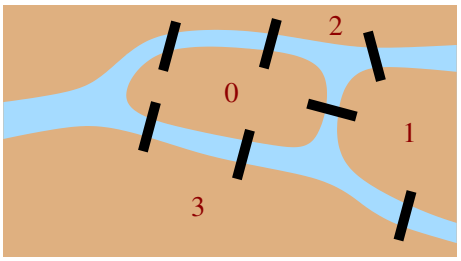
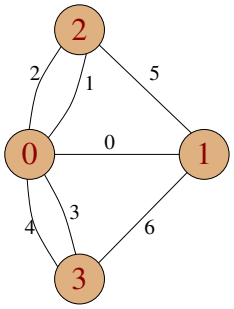
Matemàtica computacional i analítica de dades
Algorítmia i combinatòria en grafs...
Curs 2024–25

3 Dades d'un graf i camins d'arestes.

En el món dels grafs tothom està d'acord en dir que l'inici d'aquesta disciplina està en el problema proposat per Euler sobre la possibilitat o impossibilitat de realitzar un circuit sobre els ponts de la ciutat de Königsberg (de la seva època) que passi per tots ells sense repetir-ne cap. En els esquemes següents es pot veure la distribució dels ponts i el graf resultant en considerar cada regió de la ciutat[1] com un node i els ponts com arestes que els uneixen.





Encara que és ben conegut que aquest circuit (equivalent a passar per totes les arestes del graf, sense repetir-ne cap, anant d'un node a un altre de connectat) és impossible i que en l'argument per provar aquest fet no cal materialitzar cap projecte de camí concret, és un bon exercici per a codificar grafs fer un programa de l'estil de l'exemple que trobareu a continuació, en el que podreu observar que es codifica el graf a partir de les seves arestes i es demana que l'usuari provi de fer una camí que passi per les set arestes del graf.

```c
#include <stdio.h>
#define NAR 7
#define MAXARNO 5

typedef struct{
    unsigned n1,n2,estat;
}aresta;
typedef struct
{
    unsigned nodes[NAR+1];
    unsigned arvis[NAR];
    int naresvisit;
}estructuracami;

unsigned arestesdelnode(unsigned ,unsigned []);

aresta larestes[NAR]={{0,1,0},{0,2,0},{0,2,0},{0,3,0},{0,3,0},
    {1,2,0},{1,3,0}};

int main(){
    unsigned i,arestespos[MAXARNO],npos=0,seguir=0;
    estructuracami cami;

    cami.naresvisit=0;
    for(i=0;i<NAR;i++){
        printf("Aresta %d -> [%d,%d]\n",i,larestes[i].n1,larestes[i].n2);
```

[1] Les regions 0 i 1 són dues grans illes que forma el riu i la 2 i la 3 són les dues parts a banda i banda del riu.

```
27      }
28      printf("Node Inici: ");
29      scanf("%u",&cami.nodes[0]);
30      npos=arestesdelnode(cami.nodes[0], arestespos);
31      seguir=1;
32      do{
33          if(npos > 0){
34              printf("Teniu %d arestes per triar: \n",npos);
35              for(i=0;i<npos;i++){
36                  printf("%d -> [%d,%d]\n",arestespos[i],larestes[arestespos[
    i]].n1,
37                          larestes[arestespos[i]].n2);
38              }
39              printf("Trieu: ");
40              scanf("%u",&cami.arvis[cami.naresvisit]);
41              larestes[cami.arvis[(cami.naresvisit)]].estat=1;
42              (cami.naresvisit)++;
43              if(larestes[cami.arvis[(cami.naresvisit)-1]].n1==cami.nodes[(
    cami.naresvisit)-1]){
44                  cami.nodes[cami.naresvisit]=larestes[cami.arvis[(cami.
    naresvisit)-1]].n2;
45              }
46              else{
47                  cami.nodes[cami.naresvisit]=larestes[cami.arvis[(cami.
    naresvisit)-1]].n1;
48              }
49              for(i=0; i<cami.naresvisit; i++){
50                  printf("%d -(%d)-> ",cami.nodes[i],cami.arvis[i]);
51              }
52              printf("%d\n",cami.nodes[cami.naresvisit]);
53              npos=arestesdelnode(cami.nodes[cami.naresvisit],arestespos);
54          }
55          else{
56              printf("No queden arestes disponibles.\n");
57              printf("Arestes visitades (%d):\n",cami.naresvisit);
58              for(i=0;i<cami.naresvisit;i++){
59                  printf("%d ",cami.arvis[i]);
60              }
61              seguir=0;
62          }
63      }while(seguir);
64      printf("\n");
65      return 0;
66  }
67
68  unsigned arestesdelnode(unsigned nn,unsigned llistap[]){
69      unsigned i,nombre=0;
70      for(i=0;i<NAR;i++){
71          if(larestes[i].estat==0 && (larestes[i].n1==nn||larestes[i].n2==nn)
    ){
72              llistap[nombre]=i;
73              nombre++;
74          }
75      }
76      return nombre;
77  }
```

Si compileu i executeu el programa veureu que us demana un node inicial, mentre és possible, va mostrant i oferint per quines arestes es pot passar a un altre node. Quan s'arriba a un cert node i no queden arestes sense visitar que hi surtin, el programa acaba i ensenya el recorregut realitzat i el nombre total d'arestes visitades.

Encara que l'exemple és un projecte molt incomplet (en els exercicis hi ha unes quantes

propostes per a millorar el seu funcionament) cal que us fixeu en els detalls següents:

- Per tal de codificar i mantenir la configuració del graf, l'única informació que s'utilitza sobre els nodes és quants n'hi ha. La referència a un node en concret només es fa a través d'un índex entre 0 i 3 (hi ha 4 nodes). Aquesta informació s'introdueix en la variable global `larestes` que és un vector d'arestes.
- Les arestes es codifiquen amb un vector d'estructures que contenen tres dades: els dos nodes que connecta i, com que ens interessa saber quines són les arestes que ja s'han visitat en el camí que s'està construint, un tercer índex que servirà per determinar l'estat (visitat/no visitat) de cada node.
- La funció `arestesdelnode` és el nucli dels càlculs del programa. És la que determina, quan s'arriba a un node, quines arestes hi ha disponibles (i quantes) per tal de sortir-ne. Els seus arguments són: l'índex del node que s'està visitant i el vector on guardar la llista d'arestes disponibles, i retorna quantes arestes disponibles hi ha.
- L'estructura `estructuracami` és la que serveix per guardar de forma compacta les dades del camí que es va construint i recuperar la informació al final del recorregut. Conté la llista, ordenada, dels índexs dels nodes que s'han visitat (`nodes`), la llista de les arestes (`arvis`, també codificades per l'índex que els hi correspon en el moment en què es defineix el graf) i el nombre d'arestes que en formen part (`naresvisit`) (de nodes n'hi haurà sempre un més).

3.1 Exercicis

Afegiu com a les dues primeres línies del programa i en format comentari els vostres `Nom`, `Cognom` i `NIU`.

El nom dels programes que contenen el codi ha de ser de la forma `PrXExY.c`, on `X` fa referència a la pràctica i `Y` a l'exercici. Per exemple, el nom del programa de l'apartat següent hauria de ser `Pr3Ex311.c`.

Exercici 3.1.1: Eviteu que el recorregut pugui fer "salts quàntics". És a dir, feu que només s'accepti un origen dins del rang correcte i que les arestes que s'introdueixin en cada pas estiguin realment a la llista d'arestes disponibles (podeu comprovar, per exemple, que ara el programa accepta que s'entri l'aresta 5 quan estem al node 0).

Exercici 3.1.2: Dissenyeu una funció que calculi el nombre d'arestes que surten de cada node i determini quin és el node amb més arestes (valència).

Exercici 3.1.3: En el programa d'exemple el nombre de nodes i arestes és una dada que ja ve donada, però en general heu de preveure que aquesta informació vingui donada de forma implícita en la descripció inicial del graf. Penseu i implementeu un mecanisme per tal d'obtenir aquestes dades a partir d'una llista d'arestes obtinguda des d'un fitxer com el `Konigs.txt` que hi ha al Campus Virtual: la primera línia té el nombre de vèrtexs, la segona el d'arestes i la resta de línies els extrems de cada aresta. En particular, aquest mecanisme ha de servir perquè el programa de fabricar camins es pugui executar, sense fer-ne modificacions internes, sobre qualsevol graf. Podeu fer la prova amb `GrafInventat.txt`.

Exercici 3.1.4: Feu que el programa primer demani un node i després mostri **tots els camins maximals possibles amb inici aquell node i que no repeteixin cap aresta** (en aquest cas, diem que el camí és **maximal** si per a poder continuar ha de repetir alguna aresta de forma obligatòria).

Exercici 3.1.5: Feu que el programa calculi i mostri **tots els camins maximals possibles que no repeteixin cap aresta** i, per tant, comprovi que no es pot trobar un recorregut a Königsberg que passi per tots els ponts sense repetir-ne cap.

Instruccions finals

Quan acabeu la pràctica, feu el lliurament dels fitxers de codi (que tenen els noms de la forma `Pr3ExY.c` segons el que hem indicat anteriorment) a través del Campus Virtual des de l'apartat de lliuraments de l'assignatura. Recordeu que al principi de cada fitxer hi ha d'haver el vostre nom i NIU.