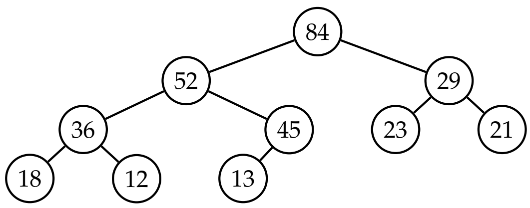
Cognoms **Nom** **DNI**

Examen Final EDA **Duració: 3 hores** **13/01/2020**

- *L'enunciat té 5 fulls, 10 cares i 4 problemes.*
- *Poseu el vostre nom complet i número de DNI a cada full.*
- *Contesteu tots els problemes en el propi full de l'enunciat a l'espai reservat.*
- *Llevat que es digui el contrari,* **cal justificar les respostes**.

Problema 1 **(1.5 pts.)**

(a) (0.5 pts.) Dibuixeu el heap resultant d'afegir l'element 64 al max-heap següent. No cal justificar la resposta.



(b) (0.5 pts.) Quina és la **recurrència** que expressa el cost de l'algorisme d'Strassen per multiplicar matrius $n \times n$? No cal justificar la resposta.

$T(n) =$

(c) (0.5 pts.) Considereu el codi següent:

```
int g(int p);
int f(int n, int p) {
    if (n == 0) return p;
    else        return 1 + f(n/2, g(p));
}
```

Si sabem que el cost de la funció g és quadràtic, quin és el cost asimptòtic en temps de f en funció de n?

Cognoms **Nom** **DNI**

Problema 2 (3 pts.)

Donat un graf dirigit acíclic (DAG) G, el *nivell* dels seus vèrtexs es defineix inductivament de la forma següent:

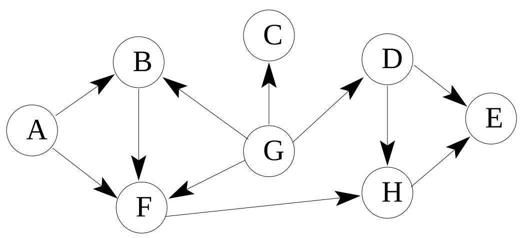
- si v és una arrel de G (un vèrtex sense predecessors) aleshores $\text{nivell}(v) = 0$
- altrament,

$$\text{nivell}(v) = 1 + \max\{\text{nivell}(u) \mid u \text{ és un predecessor de } v\}$$

A més, la profunditat de G és el nivell més gran de qualsevol vèrtex:

$$\text{profunditat}(G) = \max\{\text{nivell}(v) \mid v \text{ vèrtex de } G\}$$

(a) (0.5 pts.) Ompliu la taula següent indicant, per a cada vèrtex del DAG donat, el seu nivell. Quant val la profunditat del DAG? No cal justificar res.



nivell :

A	B	C	D	E	F	G	H

profunditat :

(b) (0.8 pts.) Per a cada afirmació donada a continuació, marqueu amb una X la casella corresponent segons si és certa o falsa. No cal justificar res.

Nota: Cada resposta correcta sumarà 0.2 punts; cada resposta equivocada restarà 0.2 punts, llevat del cas que hi hagi més respostes equivocades que correctes, en què la nota de l'exercici serà 0.

(1) Per a tot vèrtex u d'un DAG G, si u és una fulla (vèrtex sense successors) llavors $\text{nivell}(u) = \text{profunditat}(G)$.

(2) Per a tot vèrtex u d'un DAG G, si $\text{nivell}(u) = \text{profunditat}(G)$ llavors u és una fulla.

(3) La profunditat d'un DAG amb n vèrtexs és $O(n)$.

(4) La profunditat d'un DAG amb n vèrtexs és $\Omega(\log n)$.

	(1)	(2)	(3)	(4)
CERT				
FALS				

(c) (1.7 pts.) En aquest problema assumirem que els grafs es representen amb llistes d'adjacències, i que el vèrtexs s'identifiquen amb naturals consecutius 0, 1, etc.

Ompliu els buits de la funció següent:

vector <**int**> *levels* (**const** *vector* <*vector*<**int**>>& *G*);

que, donat un DAG $G = (V, E)$, retorna un vector que, per a cada vèrtex $u \in V$, conté el valor nivell(u) a la posició u. Doneu i justifiqueu el cost temporal en el cas pitjor en termes de $n = |V|$ i $m = |E|$.

```
vector<int> levels(const vector<vector<int>>& G) {
  int n = G.size ();
  vector<int> lvl(n, −1), pred(n, 0);

  for (int u = 0; u < n; ++u)
    for (int v : G[u])
      [                    ]

  queue<int> Q;
  for (int u = 0; u < n; ++u)
    if (pred[u] == 0) {
      Q.push(u);
      [                    ]
    }

  while (not Q.empty()) {
    int u = Q.front (); Q.pop();
    for (int v : G[u]) {
      −−pred[v];
      [                    ]
      if (pred[v] == 0) Q.push(v);
  } }
  return lvl ;
}
```

Cost i justificació:

Cognoms **Nom** **DNI**

Problema 3 (2.25 pts.)

El president d'una companyia petrolera ens demana ajuda per a decidir en quines ciutats col·locar les gasolineres d'un país molt llunyà. Coneixem el cost d'instal·lar una gasolinera a cada ciutat i també disposem de la llista de totes les carreteres del país, identificades per parells $\{c_1, c_2\}$, on c_1 i c_2 són les dues ciutats que uneix aquesta carretera. El nostre objectiu és decidir a quines ciutats cal posar una gasolinera de manera que el cost total d'instal·lació sigui mínim i que, per a tota carretera $\{i, j\}$ hi hagi una gasolinera a i, a j o en ambdues ciutats.

a) (2 pts) Completeu els buits en el programa següent perquè resolgui el problema anterior. Les ciutats s'identifiquen amb nombres consecutius 0, 1, 2, etc.

```
struct GasStations {
  int n;
  vector<int> cost;
  vector<vector<int>> roads;
  vector<bool> best_solution;
  int best_cost ;

  void rec(int i , vector<bool>& partial_sol, int partial_cost ) {
    if (i == n) {
      best_cost = partial_cost ;
      best_solution = partial_sol ;
    }
    else {
      if ( ________________ < best_cost) {
        partial_sol [i] = ________ ;
        rec( ______ ,partial_sol, ____________________ );
      }

      bool needed = false;
      for (auto& c : roads[i])
        if ( ________ and ______________ ) needed = true;

      if (not needed) {
        partial_sol [i] = ________ ;
        rec( ______ ,partial_sol, __________ );
  } } } // tanquem rec

  GasStations (const vector<int>& c, vector<vector<int>>& r) {
    n = c.size (); best_cost = INT_MAX;
    this->cost = c; this->roads = r;
    vector<bool> partial_sol(n,false);
    rec(0, partial_sol ,0);
} }; // (main a la pàgina següent)
```

```
int main (){
  int n; // nombre de ciutats
  cin >> n;
  vector<int> c(n); // cost de cada ciutat
  for (int i = 0; i < n; ++i) cin >> c[i];
  vector<vector<int>> r(n); // carreteres (graf com a llista d'adjacència)
  int c1, c2;
  while (cin >> c1 >> c2) { // Carretera entre c1 i c2
    r[c1].push_back(c2);
    r[c2].push_back(c1);
  }
  GasStations gas(c,r);
  cout << "Best cost: " << gas.best_cost << endl;
  cout << "Chosen cities:";
  for (int i = 0; i < n; ++i)
    if (gas.best_solution[i]) cout << " " << i;
  cout << endl;
}
```

b) (0.25 pts) Si ara ens canvien el problema lleugerament i ens demanen maximitzar el cost total d'instal·lació, què canviaríeu del codi anterior? En cas que considereu que heu de modificar massa coses, podeu explicar a alt nivell quin algorisme implementaríeu.

Cognoms **Nom** **DNI**

Problema 4 (3.25 pts.)

Recordem el problema de **3-SAT**: una *variable booleana* només pot prendre el valor 0 (fals) o el valor 1 (cert). Un *literal* és una variable booleana x o la seva negació $\neg x$. Una *clàusula* és una disjunció de literals. Una *3-CNF* és una conjunció de clàusules que contenen exactament 3 literals. Una assignació I de valors a les variables booleanes satisfà un literal x sii $I(x) = 1$, i satisfà un literal $\neg x$ sii $I(x) = 0$. El problema de **3-SAT** consisteix a, donada una 3-CNF F, determinar si existeix una assignació que satisfà **com a mínim** un literal de cada clàusula. Fins aquí, res de nou.

Introduïm ara el problema de **ONE-IN-THREE-SAT** que consisteix a, donada una 3-CNF F, determinar si existeix una assignació que satisfà **exactament** un literal de cada clàusula de F.

(a) (0.75 pts.) Considereu les 3 clàusules:

$$\begin{array}{ccccc} x_1 & \vee & \neg x_2 & \vee & x_3 \\ x_1 & \vee & x_4 & \vee & x_5 \\ \neg x_2 & \vee & x_4 & \vee & x_5 \end{array}$$

Escriviu una assignació que satisfaci exactament un literal de cada clàusula.

$I(x_1) = \square$, $I(x_2) = \square$, $I(x_3) = \square$, $I(x_4) = \square$, $I(x_5) = \square$.

Existeix alguna assignació I que satisfaci exactament un literal de cada clàusula i tal que $I(x_1) = 1$? Justifiqueu la vostra resposta.

Escriviu un conjunt **petit** de clàusules de 3 literals que sigui una entrada positiva de **3-SAT** però no ho sigui de **ONE-IN-THREE-SAT**. Acceptarem conjunts de com a molt 4 clàusules, tot i que amb 3 ja n'hauríeu de fer prou. Justifiqueu la vostra resposta.

(b) (1.5 pts.) Donada una clàusula C de 3 literals $C = l_1 \vee l_2 \vee l_3$, definim el conjunt de clàusules $R(C) = \{C_1, C_2, C_3\}$, on

$$\begin{array}{rcccccc} C_1 = & \neg l_1 & \vee & a & \vee & b \\ C_2 = & l_2 & \vee & b & \vee & c \\ C_3 = & \neg l_3 & \vee & c & \vee & d \end{array}$$

i a, b, c, d són variables booleanes noves. *Nota:* si x és una variable i el literal l és $\neg x$, entendrem que $\neg l$ es correspon a x.

Demostreu que si I és una assignació que satisfà exactament un literal de cada clàusula de $R(C)$, aleshores I satisfà almenys un literal de C.

Demostreu que donada una assignació I que satisfà almenys un literal de C, es pot construir una assignació I' que satisfà exactament un literal de cada clàusula de $R(C)$ i que coincideix amb I sobre l_1, l_2 i l_3.

Cognoms **Nom** **DNI**

(c) (0.5 pts.) Demostreu que **ONE-IN-THREE-SAT** és NP-difícil.

(d) (0.5 pts.) Demostreu que **ONE-IN-THREE-SAT** és NP-complet.

Aquesta cara estaria en blanc intencionadament si no fos per aquesta nota.