Solució de l'Examen Final EDA 08/01/2024

Proposta de solució al problema 1

(a) Si apliquem el teorema mestre de recurrències divisores de l'estil $T(n) = aT(n/b) + \Theta(n^k)$, identifiquem $b = 4$ i $k = 2$. Aleshores, necessitem comparar $\alpha = \log_4(a)$ amb $k = 2$. Els dos valors coincidiran quan $\log_4(a) = 2$, és a dir, quan $a = 16$. Aleshores:

- Si $a = 16$, tenim $\alpha = k$ i per tant, $T(n) = n^k \log n = n^2 \log n$.
- Si $a < 16$, tenim $\alpha < k$ i per tant, $T(n) = n^k = n^2$.
- Si $a > 16$, tenim $\alpha > k$ i per tant, $T(n) = n^\alpha = n^{\log_4(a)}$.

(b) El 38 no pot ser l'últim element afegit perquè això implicaria que 32 era l'anterior arrel, però això és impossible perquè és menor que 33 (fill dret de l'arrel).

La llista dels elements que poden ser l'últim afegit és $15, 20, 32$.

Proposta de solució al problema 2

(a) Recordem que un map en C++ equival essencialment a un AVL i, per tant, les operacions d'afegir un parell $(clau, informacio)$, esborrar una clau o modificar la informació associada a una clau tenen cost $\log m$ si m és el nombre d'elements que hi ha al map.

La creació del map buit té cost $\Theta(1)$. A continuació, hi ha un bucle que itera sobre els n elements x de v. Per cada x, si no apareix al map, l'afegeix com a clau amb informació 1 i en cas contrari n'incrementa la seva informació. En la primera d'aquestes operacions el map té mida com a molt 0, en la segona com a molt 1, i així successivament fins a la darrera operació on el map tindrà com a molt $n-1$ elements. Com que, en cas pitjor, cada operació és logarítmica en la mida del map, el cost total és com a molt $\log(1) + \log(2) + \cdots + \log(n-1) = \Theta(n \log n)$.

A continuació iterem sobre els elements del map, que conté com a molt n elements. Per tant ho podem fer amb temps com a molt $\Theta(n)$. Cada element l'afegim al vector *res* amb temps constant. Per tant el segon bucle té cost com a molt $\Theta(n)$.

Finalment, retornar el vector *res*, que té mida com a molt n, ens costarà com molt $\Theta(n)$.

Per tant, el cost total en cas pitjor és $\Theta(n \log n) + \Theta(n) + \Theta(n) = \Theta(n \log n)$.

(b) La idea és, enlloc d'utilitzar un map, utilitzar un vector m de mida 100 on a la posició i hi guardarem el nombre d'aparicions del nombre i. El vector l'inicialitzarem a 0. El primer bucle no canvia en absolut.

El segon bucle es reemplaçarà per:

```
for (int i = 0; i < 100; ++i)
    if (m[i] > 0) res.push_back({i, m[i]});
```

(c) Una possible solució és:

```
vector<pair<int,int>> priority (const vector<int>& v) {
  priority_queue <int> q;
  for (int x : v) q.push(x);
  vector<pair<int,int>> res;
  int current = -1;
  while (not q.empty()) {
    int x = q.top ();
    q.pop ();
    if (x ≠ current) {
      current = x;
      res .push_back({current ,1});
    }
    else ++res.back (). second;
  }
  return res ;
}
```

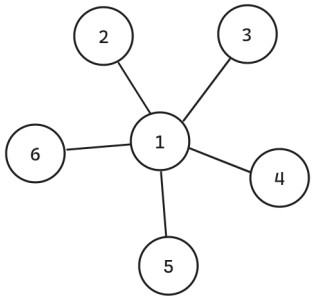
Proposta de solució al problema 3

```
pair<int,int> mov_5 (const pair<int,int>& p) { // A cap a B
  if (p. first + p.second ≤ cap_B) return {0,p. first +p.second};
  else return {p. first + p.second - cap_B,cap_B};
}

pair<int,int> mov_6 (const pair<int,int>& p) { // B cap a A
  if (p. first + p.second ≤ cap_A) return {p. first +p.second ,0};
  else return {cap_A, p. first + p.second - cap_A};
}

int operacions (const pair<int,int>& ini, int k) {
  vector<vector<int>> dist(cap_A + 1, vector<int>(cap_B + 1,INF));
  queue<pair<int,int>> Q;

  Q.push(ini );
  dist [ ini . first ][ ini .second] = 0;

  while (not Q.empty()) {
    pair<int,int> u = Q.front ();
    Q.pop ();
    vector<pair<int,int>> veins = un_pas(u );
    for (pair<int,int> v : veins ) {
      if ( dist [v. first ][v.second] == INF) {
        dist [v. first ][v.second] = dist [u. first ][u.second] + 1;
        if (v. first == k or v.second == k) return dist [v. first ][v.second ];
        Q.push(v);
      }
```

```
        }
      }
      return −1;
    }
```

Proposta de solució al problema 4

(a) Prenem el graf següent:



Clarament és una instància positiva de **3-COL**: es pot colorejar amb 3 colors pintant, per exemple, el node 1 amb el color 1 i els altres nodes amb el color 2.

També podem veure que és una instància negativa de **3-COL-EQUIL**: com que tenim 6 nodes, cada color s'hauria d'utilitzar dues vegades. El node central (1) s'ha de pintar amb algun color, però no podem utilitzar aquest color per cap altre node perquè estan tots connectats amb 1. Per tant, no existeix una coloració com la que busquem.

(b) Prenem com a conjunt de testimonis totes les coloracions possibles, és a dir, les funcions $c : V \to \{1, 2, 3\}$. Donada una instància $G = (V, E)$ amb n vèrtexs i m arestes (i per tant, mida $\Theta(n + m)$ si el graf ve representat amb llistes d'adjacència), clarament qualsevol testimoni és polinòmic respecte la mida de G, perquè necessitem com a molt $2n$ bits per representar una coloració.

Com a verificador prenem el següent algorisme: donat un graf $G = (V, E)$ i un testimoni (coloració) c:

1.- Per a tota aresta $\{u, v\} \in E$, si $c(u) = c(v)$ retornem 0.

2.- Calculem n_1, n_2, n_3 el nombre vèrtexs amb color 1,2 i 3, respectivament. Si algun d'ells no és n, retornem 0.

3.- Retornem 1.

Hem de comprovar que el verificador és polinòmic. En el pas 1, amb un graf representat amb llistes d'adjacència, podem recórrer les arestes en temps $\Theta(n + m)$ i, per cada una d'elles, fem un treball $\Theta(1)$. Per tant, el pas 1 té cost $\Theta(n + m)$. Pel pas 2, només cal recórrer els n vèrtexs i actualitzar n_1, n_2, n_3, cosa que clarament es pot fer en temps $\Theta(n)$. Per tant, el cost del verificador és polinòmic.

Sigui ara G una instància positiva. Aleshores, per definició del problema, existeix una coloració c tal que pinta els vèrtexs de cada aresta amb colors diferents

i tals que $|C_i| = n$ per a $1 \leq i \leq n$. Si prenem aquesta coloració com a testimoni, veiem que el verificador no pot acabar en el pas 1 perquè totes les arestes estan colorejades correctament, i tampoc en el pas 2, perquè precisament $n_i = |C_i|$. Així doncs, existeix un testimoni pel qual el verificador retorna 1.

Sigui ara G una instància negativa. Aleshores, per definició del problema, si agafem una coloració qualsevol, o bé colorejarà els dos vèrtexs d'alguna aresta amb el mateix color o bé la cardinalitat d'algun C_i no serà n. Així doncs, si agafem un testimoni qualsevol (una coloració), el verificador retornarà 0 en el pas 1 o en el pas 2, depenent del cas. Per tant, sempre retornarà 0.

(c) Clarament, $|V'| = 3n$. A més, si la mida de G és $n + m$, on m és $|E|$, la mida de G' és $3n + 3m$ ($3n$ vèrtexs i $3m$ arestes), que clarament és polinòmica respecte $n + m$. A més, també es pot calcular en temps polinòmic, ja que només fem 3 còpies de G.

Anem a veure ara que, si G és una instància positiva, G' també ho és. Efectivament, si G és una instància positiva, sigui c una coloració correcta per G. Podem construir una coloració c' que respecta les arestes de G' i té n vèrtexs de cada color de la manera següent:

- $c'(v_i^1) = c(v_i)$ per a tot $1 \leq i \leq n$
- Per a tot $1 \leq i \leq n$:

$$c'(v_i^2) = \begin{cases} 2 & \text{si } c(v_i) = 1 \\ 3 & \text{si } c(v_i) = 2 \\ 1 & \text{si } c(v_i) = 3 \end{cases}$$

- Per a tot $1 \leq i \leq n$:

$$c'(v_i^3) = \begin{cases} 3 & \text{si } c(v_i) = 1 \\ 1 & \text{si } c(v_i) = 2 \\ 2 & \text{si } c(v_i) = 3 \end{cases}$$

És fàcil veure que si $c(v_i) \neq c(v_j)$, aleshores $c'(v_i^k) \neq c'(v_j^k)$. Així doncs, si prenem una aresta de la forma $\{v_i^k, v_j^k\}$, com que $\{v_i, v_j\} \in E$, necessàriament $c(v_i) \neq c(v_j)$ i per tant c' coloreja els dos vèrtexs de l'aresta de color diferent.

Finalment, només cal veure que c' coloreja exactament n vèrtexs amb cada color. Això és fàcil de veure si observem que per a cada vèrtex $v_i \in G$, els 3 vèrtexs v_i^1, v_i^2, v_i^3 estan colorejats a c' amb 3 colors diferents. Per tant, com que cada color apareix exactament una vegada a cada tripleta $(c(v_i^1), c(v_i^2), c(v_i^3))$, si construïm totes les tripletes d'aquest tipus podrem veure que cada color es fa servir exactament n vegades (una vegada a cada tripleta).

Per acabar de demostrar que és una reducció correcta, assumim que G' és una instància positiva de **3-COL-EQUIL** i veiem que G és una instància positiva de **3-COL**. Això és fàcil de veure si observem que G' conté 3 còpies de G, totes elles ben colorejades. Per tant, podem colorejat G amb els colors utilitzats a la primera còpia de G.