Cognoms **Nom** **DNI**

Examen Parcial AP3 **Duració: 2.5 hores** **30/10/2018**

- *L'enunciat té 4 fulls, 8 cares i 3 problemes.*
- *Poseu el vostre nom complet i número de DNI a cada full.*
- *Contesteu tots els problemes en el propi full de l'enunciat i a l'espai reservat.*
- *A no ser que es digui el contrari, cal justificar les respostes.*

Problema 1 **(3 punts)**

Fixat un natural $k \geq 2$, el problema de k-COLOR consisteix en, donat un graf no dirigit $G = (V, E)$, determinar si existeix un k-colorejat del graf; això és, una funció $C : V \to \{1, \ldots, k\}$ tal que per tot $\{u, v\} \in E$ es compleix $C(u) \neq C(v)$.

(a) (1.5 pts.) Demostreu que k-COLOR es redueix polinòmicament a $(k+1)$-COLOR.

(b) (1.5 pts.) Considereu la següent afirmació:

"$(k+1)$-COLOR es redueix polinòmicament a k-COLOR"

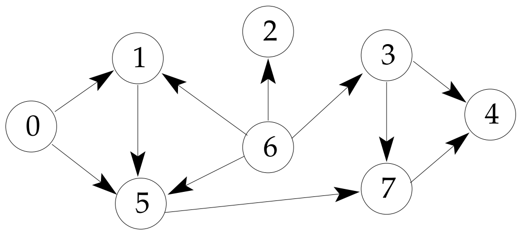
Per a quins valors de $k \geq 2$ podem assegurar que és certa? Per a quins valors de k no podem fer-ho? Justifiqueu la resposta.

Cognoms Nom DNI

Problema 2 **(4 punts)**

Un *DAG* és un graf dirigit sense cicles. Donat un DAG $G = (V, E)$, una *ordenació topològica* de G és una enumeració $v_0, v_1, \ldots, v_{n-1}$ de tots els vèrtexs de G tal que per tota aresta $(v_i, v_j) \in E$ es compleix $i < j$.

Per exemple, una ordenació topològica del següent DAG:



és $(0, 6, 1, 5, 2, 3, 7, 4)$. En canvi, $(0, 1, 6, 5, 2, 3, 7, 4)$ no ho seria perquè al DAG hi ha una aresta $(6, 1)$, i el vèrtex 1 apareix abans que el vèrtex 6 a l'enumeració.

(a) (1 pt.) Considerem el problema de, donat un DAG $G = (V, E)$, generar totes les seves ordenacions topològiques. El següent programa el resol (assumint $V = \{0, 1, ..., n-1\}$):

```
#include <iostream>
#include <vector>
using namespace std;

typedef vector<int> SuccList;
typedef vector<SuccList> Graph;

// implementation not given here
Graph read_graph ();
void print_solution (const vector<int>& sol);

bool ok(const Graph& G, const vector<int>& sol) {
  int n = sol.size ();
  vector<bool> mar(n, false);
  for (int i = 0; i < n; ++i) {
    int x = sol[i];
    if (mar[x]) return false;
    mar[x] = true;
    for (int y : G[x])
      for (int j = 0; j < i; ++j)
        if (sol[j] == y) return false;
  }
  return true;
}
```

```
void top_sorts_rec (int k, const Graph& G, vector<int>& sol) {
  int n = sol.size();
  if (k == n) {
    if (ok(G, sol))
      print_solution(sol);
  }
  else
    for (int x = 0; x < n; ++x) {
      sol[k] = x;
      top_sorts_rec(k+1, G, sol);
    }
}

void top_sorts (const Graph& G, vector<int>& sol) {
   top_sorts_rec(0, G, sol);
}
int main() {
  Graph G = read_graph();
  vector<int> sol(G.size());
  top_sorts(G, sol);
}
```

Demostreu que el cost d'aquest programa és $\Omega(n^{n+1})$, on n és el nombre de vèrtexs del DAG.

Cognoms **Nom** **DNI**

(b) (1.5 pts.) Donat un DAG $G = (V, E)$, definim el *grau d'entrada* d'un vèrtex $v \in V$ com el nombre d'arestes que arriben a v, és a dir, $|\{u \in V \mid (u, v) \in E\}|$.

Donat un vèrtex $u \in V$, definim també el graf G_u com el graf amb vèrtexs $V_u = V - \{u\}$ i arestes $E_u = E - \{(v, w) \mid v = u \lor w = u\}$.

Demostreu que $v_0, v_1, \ldots, v_{n-1}$ és una ordenació topològica de G si i només si el grau d'entrada de v_0 és 0 i $v_1, \ldots, v_{n-1}$ és una ordenació topològica de G_{v_0}.

(c) (1.5 pts.) Ompliu els buits del següent programa, que reimplementa la funció *top_sorts* de l'apartat (a) per resoldre més eficientment el problema de generar totes les ordenacions topològiques.

```
void top_sorts_rec (int k, const Graph& G, vector<int>& sol, vector<int>& indegree) {
  int n = sol.size();
  if (k == n)
    print_solution(sol);
  else
    for (int x = 0; x < n; ++x) {

    }
}

void top_sorts (const Graph& G, vector<int>& sol) {
  int n = G.size();
  vector<int> indegree(n, 0);

  top_sorts_rec(0, G, sol, indegree);
}
```

Cognoms Nom DNI

Problema 3 (3 punts)

Donats un enter $v \geq 0$ i n valors de monedes $m_0, m_1, \ldots, m_{n-1}$ diferents i estrictament positius, considereu el problema de comptar el nombre de maneres d'obtenir canvi v usant les monedes com a molt una vegada.

(a) (1.5 pts.) Ompliu els buits del següent programa de programació dinàmica top-down per què resolgui el problema.

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> m;
vector<vector<int>> c;


int f(int i, int x) {
  if (x < 0) return [          ];
  int& res = c[i][x];
  if (res ≠ −1) return res;
  if (i == 0) {
    [                                        ]
  }
  return res = f([          ], x) + f([          ], [          ]);
}


int main() {
  int n;
  cin >> n;
  m = vector<int>(n);
  for (int& k : m) cin >> k;
  int v;
  cin >> v;

  c = vector<vector<int>>(n + 1, vector<int>(v + 1, −1));
  cout << f(n, v) << endl;
}
```

(b) (1.5 pts.) Escriviu un programa bottom-up que resolgui el problema en temps $\Theta(v \cdot n)$. Es valorarà l'eficiència en espai del programa. En particular, les solucions amb cost en espai $\Omega(v \cdot n)$ només podran aconseguir, com a molt, la meitat de la puntuació.

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
  int n;
  cin >> n;
  vector<int> m(n);
  for (int& k : m) cin >> k;
  int v;
  cin >> v;

}
```