Solució de l'Examen Parcial AP3 30/10/2018

Proposta de solució al problema 1

(a) Donat un graf $G = (V, E)$, la reducció retorna el graf $G' = (V', E')$ obtingut en afegir un nou vèrtex $u^* \notin V$ (o sigui, $V' = V \cup \{u^*\}$) i connectar aquest nou vèrtex amb tots els altres (és a dir, $E' = E \cup \{\{u, u^*\} \mid u \in V\}$).

Aquest algorisme funciona en temps polinòmic (de fet, lineal) en la mida del graf d'entrada. A més, si C és un k-colorejat de G, aleshores podem obtenir un $(k+1)$-colorejat de G' estenent C amb $C(u^*) = k+1$. En efecte, les arestes de E continuen tenint els extrems pintats amb els colors de C. I per tota aresta de la forma $\{u, u^*\}$ tenim $C(u) \leq k < k+1 = C(u^*)$, i per tant $C(u) \neq C(u^*)$.

Falta veure que si G' admet un $(k+1)$-colorejat C, aleshores G és k-colorejable. Intercanviant colors si cal, podem suposar que $C(u^*) = k+1$. Com que per tot $u \in V$ tenim una aresta $\{u, u^*\}$ a G', llavors $C(u) \leq k$ per tot $u \in V$. Per tant C restringit a V només pren valors a $\{1, \ldots, k\}$, de manera que C és un k-colorejat de G.

(b) Per a $k \geq 3$, el problema k-COLOR és NP-complet. De forma que per tot $k \geq 3$, el problema de $(k+1)$-COLOR (que pertany a NP) es redueix polinòmicament a k-COLOR (donat que és NP-hard).

Per altra banda, el problema 2-COLOR pertany a P. Això vol dir que, si existís una reducció polinòmica de 3-COLOR a 2-COLOR, aleshores 3-COLOR es podria resoldre en temps polinòmic i tindríem P = NP. Com que aquest és un problema obert, per a $k = 2$ no podem afirmar que hi hagi tal reducció.

Proposta de solució al problema 2

(a) El programa genera cadascuna de les n^n seqüències de n nombres entre 0 i $n-1$. Per cadascuna d'elles es crida la funció *ok*, que té cost $\Omega(n)$ per la inicialització del vector de marques. Així que el programa té cost $\Omega(n^n \cdot n) = \Omega(n^{n+1})$.

(b) Suposem que $v_0, v_1, \ldots, v_{n-1}$ és una ordenació topològica de G. Llavors:

- el grau d'entrada de v_0 és 0. En efecte, si fos estrictament positiu existiria una aresta de la forma $(v_i, v_0) \in E$. Però això contradiria que $v_0, v_1, \ldots, v_{n-1}$ és una ordenació topològica.
- $v_1, \ldots, v_{n-1}$ és una ordenació topològica de G_{v_0}, donat que per tota aresta $(v_i, v_j) \in E_{v_0}$ es compleix $(v_i, v_j) \in E$, i per ser $v_0, v_1, \ldots, v_{n-1}$ ordenació topològica tenim $i < j$.

Vegem la implicació contrària. Assumim que el grau d'entrada de v_0 és 0 i que $v_1, \ldots, v_{n-1}$ és una ordenació topològica de G_{v_0}. Prenem una aresta $(v_i, v_j) \in E$. Si v_0 apareix a (v_i, v_j), llavors ha de ser de la forma (v_0, v_j) amb $j > 0$, perquè el grau d'entrada de v_0 és 0. Si v_0 no apareix a (v_i, v_j), aleshores $(v_i, v_j) \in E_{v_0}$ i per ser $v_1, \ldots, v_{n-1}$ ordenació topològica de G_{v_0}, tenim que $i < j$.

(c) Una possible solució:

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

void top_sorts (const Graph& G, vector<int>& sol) {
  int n = G.size();
  vector<int> indeg(n, 0);
  for (int x = 0; x < n; ++x)
    for (int y: G[x])
      ++indeg[y];
  top_sorts_rec(0, G, sol, indeg);
}
```

Proposta de solució al problema 3

(a) Una possible solució:

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

int f(int i, int x) {
  if (x < 0) return 0;
  int& res = c[i][x];
  if (res != -1) return res;
  if (i == 0) {
    if (x == 0) return res = true;
    else        return res = false;
  }
  return res = f(i - 1, x) + f(i - 1, x - m[i-1]);
}
```

```
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) Una possible solució:

```
int main() {
  int n;
  cin >> n;
  vector<int> m(n);
  for (int& k : m) cin >> k;
  int v;
  cin >> v;
  vector<int> c(v + 1, 0);
  c[0] = 1;
  for (int i = 0; i < n; ++i)
    for (int j = v; j >= m[i]; --j) c[j] += c[j - m[i]];
  cout << c[v] << endl;
}
```