Cognoms **Nom** **DNI**

Examen Final EDA Duració: 3h 10/01/2022

- *L'enunciat té 4 fulls, 8 cares, i 4 problemes.*
- *Poseu el vostre nom complet i número de DNI a cada problema.*
- *Contesteu tots els problemes en el propi full de l'enunciat a l'espai reservat.*
- *Llevat que es digui el contrari, sempre que parlem de cost ens referim a cost asimptòtic en temps.*
- *Llevat que es digui el contrari,* **cal justificar les respostes.**

Problema 1 (2.5 pts.)

Per simplificar l'anàlisi, en tot aquest problema podeu assumir que una crida al mètode *insert* de la classe *unordered_set* té sempre cost $\Theta(1)$.

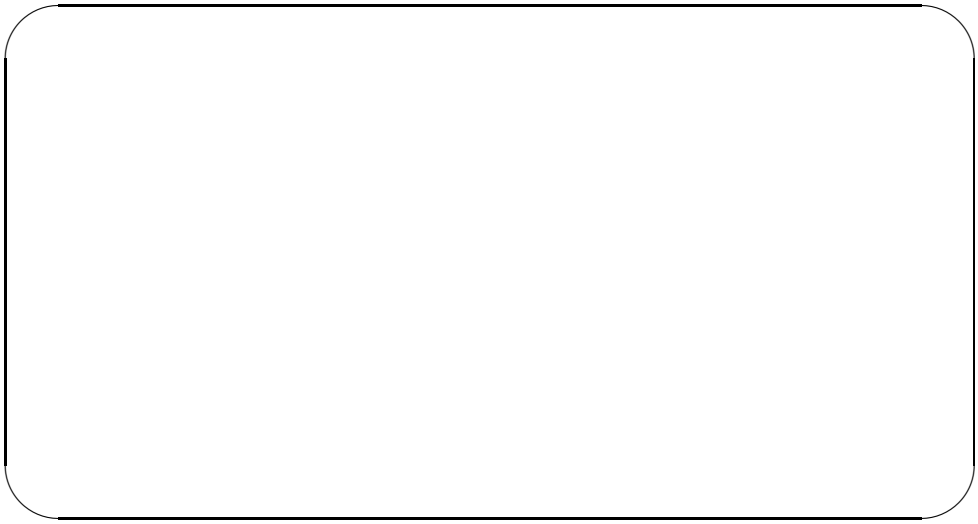
(a) (1 punt) Recordem que la classe *string* en C++ té un mètode *substr(int i, int l)* tal que donat un *string s*, la crida *s.substr(i, l)* retorna l'*string* que comença a la posició i i té llargada l. Assumirem que el cost d'una crida a aquest mètode és $\Theta(l)$. Per exemple, si s és $''paraula''$, aleshores $s.substr(1,4)$ retorna *arau*. Considereu el codi següent:

```
void mystery(const string & s, unordered_set<string>& res){
  res . insert (s );
  if (s . size () > 1) { // call to size has cost Theta(1)
    mystery(s . substr (1, s . size ()-1), res );
    mystery(s . substr (0, s . size ()-1), res );
  }
}

int main( ){
  string s; cin >> s;
  unordered_set<string> res;
  mystery(s, res );
  for (string x : res) cout << x << endl;
}
```

Si l'*string* que es llegeix al *main* és *"pep"*, quants *strings* s'escriuran per la sortida estàndard? Quins són aquests *strings*? No cal raonar la resposta.

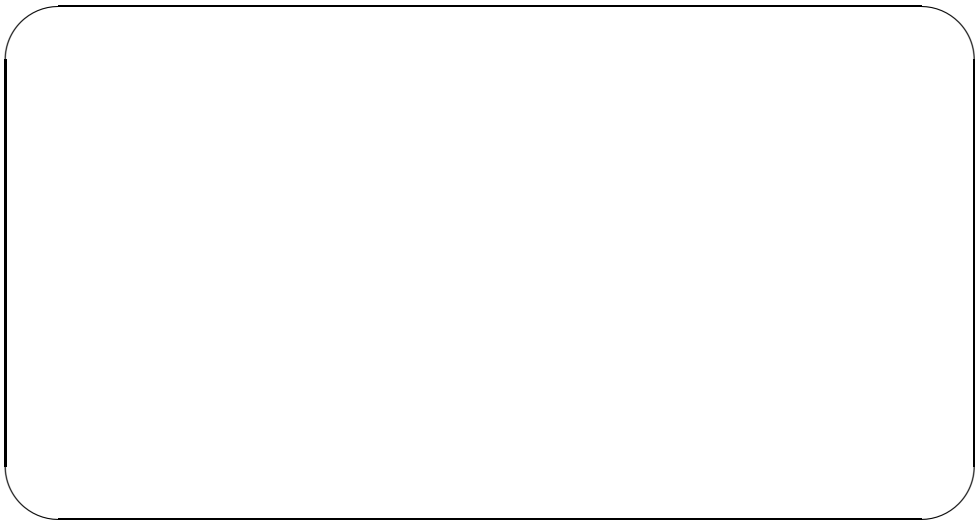
Si dins el *main* es llegeix un *string* de mida n, quin cost té la crida a *mystery*?



(b) (1 punt) Considerem ara una nova funció *mystery*:

```
void mystery(const string& s, unordered_set<string>& res) {
  for (int i = 0; i < s.size(); ++i)
    for (int j = i; j < s.size(); ++j)
      res.insert(s.substr(i, j-i+1));
}
```

Si dins el *main* es llegeix un *string* de mida n, quin cost té ara la crida a *mystery*?



(c) (0.5 pts.) Ompliu el següent codi perquè calculi el mateix que l'apartat anterior:

```
void mystery(const string& s, unordered_set<string>& res){
  for (int k = 1; k <= s.size(); ++k)
    for ( [          ] )
      res.insert(s.substr(i, k));
}
```

Cognoms **Nom** **DNI**

Problema 2 (2.5 pts.)

Després de molts anys, el professorat d'EDA decideix renovar el funcionament del Joc. A partir d'ara, les partides seran entre 3 jugadors. El resultat de cada partida és una tripleta (j_1, j_2, j_3) que indica que el jugador j_1 ha guanyat la partida, j_2 ha estat el segon, i j_3 ha estat el pitjor jugador. Enlloc de les clàssiques rondes, on a cada ronda s'eliminava un jugador, ara es realitzaran un munt de partides entre tripletes de jugadors i en guardarem els resultats. Finalment, per determinar la nota del joc volem establir un rànquing de jugadors, és a dir, una llista ordenada de jugadors on els millors jugadors haurien de sortir a les primeres posicions. Per tal que cap estudiant se senti agreujat, volem garantir que per tot parell de jugadors j_1 i j_2, si j_1 ha quedat en millor posició que j_2 en alguna partida, aleshores j_1 ha d'aparèixer abans que j_2 al rànquing.

a) (0.75 pts.) Ompliu les caselles del codi següent per trobar un rànquing correcte.

```
struct Match {
  int first; int second; int third;
  Match(int f, int s, int t): first(f), second(s), third(t){}
};
int n;
vector<Match> matches;
bool good_ranking (const vector<int>& pos_in_ranking) {
  [                                                    ]
}

bool find_ranking (vector<int>& ranking, vector<int>& pos_in_ranking,
                   vector<bool>& used, int idx){
  if (idx == n) return good_ranking(pos_in_ranking);
  else {
    for (int i = 0; i < n; ++i) {
      if (not used[i]) {
        ranking[ [    ] ] = [    ];
        pos_in_ranking[ [    ] ] = [    ];
        used[i] = true;
        if (find_ranking(ranking, pos_in_ranking, used, idx+1)) return true;
        used[i] = false;
      } } }
  return false; }
```

```
int main () {
  cin >> n; // Students are numbers from 0 to n-1
  int f, s, t; // Read results of matches
  while (cin >> f >> s >> t) matches.push_back(Match(f,s,t));
  vector<int> ranking(n), pos_in_ranking (n);
  vector<bool> used(n,false);
  bool b = find_ranking (ranking, pos_in_ranking ,used ,0);
  cout << "Ranking found: " << b << endl;
  if (b) print_vector (ranking);
}
```

b) (0.5 pts). Assumim que tenim m partides i que ens donen una mala implementació de *good_ranking* que sempre triga temps $\Theta(m)$. En funció d'n, quantes vegades es crida a la funció *good_ranking* en el cas pitjor? A partir d'aquest nombre dóna una fita inferior en funció d'n i m del cost en cas pitjor del codi anterior. Valorarem la precisió d'aquest fita.

c) (1.25 pts.) És possible solucionar el problema anterior en temps polinòmic en n i m en cas pitjor? Si és possible, explica molt breument com ho faries i justifica el cost. Si no és possible, explica per què.

Cognoms **Nom** **DNI**

Problema 3 **(2.5 pts.)**

(a) (0.75 pts.) Després d'una llarga vida dedicada a obscurs negocis, el cap d'una perillosa organització mafiosa decideix reunir, a mode de comiat, els seus n col·laboradors per agrair-los la feina feta. No obstant, treballar en assumptes tan delicats ha fet que cadascun d'ells tingui una llista de col·laboradors amb qui no vol coincidir. Sabem, a més, que si A apareix a la llista de persones que B vol evitar, B apareix també a la llista de persones que A vol evitar. El cap disposa de 5 dies, i vol citar cada treballador exactament un dia de manera que es respectin els desitjos de no-coincidència. Seríeu capaços de determinar, en temps polinòmic en n, si es poden organitzar aquestes 5 trobades?

Nota: en aquesta pregunta i les següents, **cal** justificar les respostes escrivint reduccions i utilitzant que, per certs problemes que hem vist a classe, es coneixen (o no) algorismes polinòmics que els resolen. No cal demostrar que les reduccions són correctes, però heu de deixar clar des de quin problema a quin altre es fa la reducció.

(b) (1 pt.) Després de pensar-s'ho una estona, el cap decideix que només vol dedicar 2 dies per reunir a tothom. Seríeu capaços de solucionar aquest problema en temps polinòmic en n?

(c) (0.75 pts.) Finalment, el cap canvia de parer i decideix que les restriccions dels col·laboradors no tenen massa sentit i per tant, les ignorarà. Continua disposant només de 2 dies, i no vol que s'agrupin tots els col·laboradors més importants un dia, i els de menys importància l'altre. Per tal d'aconseguir-ho, sap el patrimoni de tots els seus col·laboradors i vol aconseguir que el patrimoni total dels col·laboradors reunits el primer dia sigui igual al patrimoni dels reunits el segon dia. Seríeu capaços de solucionar aquest problema en temps polinòmic en n?

Cognoms **Nom** **DNI**

Problema 4 **(2.5 pts.)**

Donat un vector v d'n enters diferents ordenats de forma creixent i un enter x, volem determinar si x apareix a v. Si és el cas, també volem saber la seva posició. Com bé sabem, aquest problema el podem solucionar en temps $\Theta(\log n)$ en cas pitjor. No obstant, ens asseguren que de fet x sempre hi apareix i que gairebé sempre ho fa en les primeres posicions del vector. Amb aquesta informació a les mans, ens interessa trobar un algorisme tal que, si l'aparició d'x dins v és a la posició i, aleshores l'algorisme triga temps $\Theta(\log i)$ en cas pitjor.

(a) (1 pt.) Completa el següent codi per resoldre, en temps $\Theta(\log i)$, el problema que acabem de presentar.

```
// Si x apareix dins v[l...r] retorna i tal que v[i] = x
// Si x no apareix dins v[l...r] retorna -1
// Cost: Theta(log(r-l+1))
int bin_search(int x, const vector<int>& v, int l, int r);

int search(int x, const vector<int>& v) {
  int n = v.size();
  if (n == 0) return -1;
  int b = 1;
  while ( [          ] and [          ] ) b *= 2;
  return bin_search(x,v,b/2, [          ]);
}
```

(b) (1.5 pts.) Demostra que, efectivament, si l'aparició d'x dins v és a la posició i, aleshores la funció *search* triga temps $\Theta(\log i)$ en cas pitjor.

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