Representantes ordenados

Problema

Dada una lista de grupos de trabajos prácticos $g_1, g_2, \ldots, g_n$, queremos saber si es posible elegir un representante de cada grupo de tal forma que nos queden ordenados alfabéticamente.

Representantes ordenados

Problema

Dada una lista de grupos de trabajos prácticos $g_1, g_2, \ldots, g_n$, queremos saber si es posible elegir un representante de cada grupo de tal forma que nos queden ordenados alfabéticamente.

Ejemplo 1

[[Andrea, Juan], [Pedro, Juana, Roberta], [Sandro]
Rta: Se puede, por ejemplo tomando Andrea, Juana, Sandro.

Ejemplo 2

[[Andrea, Juan], [Pedro, Juana, Roberta], [Celia]]
Rta: No se puede.

1. Definir subproblemas

1. Definir subproblemas

Propuesta:

- $S_{i,s} =$ "decidir si existe una solución para los grupos $g_i, \ldots, g_n$ tal que el primer representante es mayor o igual a s".
- el problema completo sería $S_{1,}$""

1. Definir subproblemas

Propuesta:

- $S_{i,s}$ = "decidir si existe una solución para los grupos $g_i, \ldots, g_n$ tal que el primer representante es mayor o igual a s".
- el problema completo sería $S_{1,}$""
- Para resolver $S_{i,s}$ elegimos un representante r_i para g_i y pasamos a resolver S_{i+1,r_i}.

2. Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?

2. Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).

2. Elección greedy

- Podemos elegir un representante en cada paso de tal forma que después no nos arrepintamos?
- Sí! Tomamos el más chico posible (alfabéticamente).
- $r_i = min\{x \in g_i : r_{i-1} < x\}$ ($r_0 = ""$)
- Y podemos llamar $R_i = r_1, ..., r_i$ a nuestras soluciones parciales.
- Existen siempre?

3. Demostración de correctitud

Teorema

La estrategia greedy *Rn* va a estar bien definida (siempre encuentra un representante) si y solo si existe una solución para los grupos dados.

$\Rightarrow)$

Queremos ver que si *Rn* está bien definida, existe una solución. Por hipótesis sabemos que existen $r_1, \ldots, r_n$, que por definición son $r_1 < \ldots < r_n$. Listo, esa es una solución.
($r_i = min\{x \in g_i : r_{i-1} < x\}$)

$\Leftarrow)$

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto. s_{i+1} está porque $s_{i+1} > s_i$ y $s_i \geq r_i$ por HI.

$\Leftarrow$)

Sabemos que existe una solución $s_1, \ldots, s_n$ y queremos ver que R_n está bien definida. Veamos por inducción que las R_i van a estar bien definidas. Formalmente:

$P(i) =$ Si existe solución e $i \leq n$, entonces R_i está bien definida y $r_i \leq s_i$

Caso base: $r_1 = min\{x \in g_1 : "" < x\}$.
s_1 está en el conjunto así que R_1 está bien definida. Y además $r_1 \leq s_1$ por ser el mínimo.
Paso inductivo: $(P(i) \implies P(i+1))$. Si $i \geq n$ ya está. Si no:
Sabemos que $R_i = r_1, \ldots, r_i$ está bien definida. Queremos ver que $r_{i+1} = min\{x \in g_{i+1} : r_i < x\}$ está bien definido.
Afirmo que s_{i+1} está en ese conjunto. s_{i+1} está porque $s_{i+1} > s_i$ y $s_i \geq r_i$ por HI.
Además $r_{i+1} \leq s_{i+1}$ por ser el mínimo del conjunto.

4. Implementación

```
bool representantes(vector<vector<string> >& grupos){
    string anterior = "";
    for(vector<string>& g : grupos){
        string r = "";
        for(string& e : g){
            if(e > anterior) if(r == "") r = e; else r = min(r, e);
        }
        if(r == "") return false;
        anterior = r;
    }
    return true;
}
```

- Complejidad?

4. Implementación

```
bool representantes(vector<vector<string> >& grupos){
    string anterior = "";
    for(vector<string>& g : grupos){
        string r = "";
        for(string& e : g){
            if(e > anterior) if(r == "") r = e; else r = min(r, e);
        }
        if(r == "") return false;
        anterior = r;
    }
    return true;
}
```

- Complejidad? $O(n)$ con n la cantidad de estudiantes.

4. Changüí: versión recursiva

```
bool representantesBT(int i, const string& s, vector<vector<string> >& grupos){
    if(i == grupos.size()) return true;
    for(string& e : grupos[i]){
        if(e > s && representantesBT(i+1, e, grupos)) return true;
    }
    return false;
}

bool representantesGrRec(int i, const string& s, vector<vector<string> >& grupos){
    if(i == grupos.size()) return true;
    string r = "";
    for(string& e : grupos[i]){
        if(e > s) if(r == "") r = e; else r = min(r, e);
    }
    if(r == "") return false;
    return representantesGrRec(i+1, r, grupos);
}
```