Laboratorio Tarea B
Reformas al congreso

Canario García
Nelson Olveira
Arquero de Cacho Bochinche

Contenido

1 Cambios

1.1 Parametros de funcions

disponible, compatibles, predicado, esMejor

En estas funciones en lugar de pasar el puntero se pasa el valor entero, no se si se gana mucho o capáz nada, como son punteros...

```
//AHORA
bool disponible (int chini, int chfin, int sini, int sfin);
bool compatibles (int chiini, int chifin, int chjini, int chjfin);
bool esMejor(int tn, int sn);
bool predicado(int tn, int tn1, int sn);

//ANTES
bool disponible (int idCharla, int idSala, charla *charlas,
sala *salas);
bool disponible (int idCharla, int idSala, charla *charlas,
sala *salas);
bool esMejor(int *tupla, int *solucion, charla* charlas, int n);
bool predicado(int *tupla, int *solucion, int n);
```

1.2 Función verificar rest

verificarRest

Ahora hay solo un while que se ejecuta buscando compatibilidad entre las charlas de $charlas[0...i-2]$ y la charla de la posición *i-1*. La compatibilidad de vuelve a poner en el for del backtracking y se evalúa solo con la charla *i-ésima*.

verificarRest

```
bool verificaRest (int *t, int i, charla *charlas)
{
        int j = 0;

        bool boolean = true;

                while ((j<i-1) && boolean)
                {

                        if ( t[j]!=-1 && (t[i-1]==t[j]) &&
                        !compatibles( charlas[j].inicio, charlas[j].fin
                        charlas[i-1].inicio, charlas[i-1].fin ))

                                boolean = false;

                        j++;

                }

        return boolean;
}
```

disponibilidad en backtracking

Antes de avanzar con una sala para asignarsela a la charla de turno se pregunta si es compatible.

```
for(int j=i; j<n; j++){
        for(int xi=0; xi < m; xi++)
        {
                if ( (xi < 0) || disponible(charlas[j].inicio,
charlas[j].fin, salas[xi].inicio, salas[xi].fin )){
                        if(j<n){
                                tupla[j]=xi;

                                if(xi>=0){
                                tupla[n]+=charlas[j].asistentes;
                                tupla[n+1]-=charlas[j].asistentes;
                                }

                                backtracking(charlas, n, salas, m,
                                solucion, j + 1, tupla);

                                if(xi>=0){
                                tupla[n]-=charlas[j].asistentes;
                                tupla[n+1]+=charlas[j].asistentes;
                                }

                                if(tupla[j]=m)
                                        tupla[j]=-1;
                                }
                }

        }
}
```