Algorisme i Estructura de Dades

Algorisme

El nostre projecte crea l'horari mitjançant un algorisme que s'estructura en dues parts principals:

- Backtracking cronològic
- Forward checking

Backtracking cronològic

En aquesta part explicarem el backtracking cronològic que és la primera part del nostre algorisme.

Com a paràmetres de l'algorisme tenim l'horari, en un inici buit, que anem emplenant a mesura que va avançant l'algorisme, l'estructura de dades esmentada anteriorment on hi ha guardades totes les sessions que s'han d'assignar i les seves possibles aules en determinats dies i hores, i un iterador que es es va movent per aquesta estructura de dades.

L'algorisme funciona de manera que itera pels dies, hores i aules possibles que pot tenir una sessió, si el gap de l'horari està buit, comprova les restriccions i si les compleix l'assigna.

Llavors propaga les possibilitats per l'estructura de dades on tenim guardades les possibilitats. Llavors podem passar dues coses:

- Aquesta propagació provoca que hi hagi alguna sessió sense possibilitats: Esborrem l'assignació efectuada anteriorment, recuperem les possibilitats que hi havien abans de propagar i continuem el bucle.
- Es fa una poda de domini per totes les sessions, Esborrem de l'estructura de dades les possibilitats de la sessió ja assignada i continuem amb una crida recursiva per a la següent sessió.

En el cas que fem una crida recursiva per a la següent sessió també tenim dues possibilitats:

- No hi hagi cap error i l'algorisme continuï fins que estigui tot ple
- Falli en algun lloc: en aquest cas recuperem les possibilitats que teníem abans de propagar, esborrem totes les assignacions que havíem fet i contínuem el bucle.

Finalment si s'han realitzat tots els bucles, retornem que no s'ha pogut efectuar la sessió que estàvem comprovant. En el cas que sigui la primera sessió significarà que no es pot efectuar l'horari de cap manera, altrament, es farà backtracking.

Un cop l'iterador arribi a l'última sessió que s'ha d'assignar retorna true perquè vol dir que ha acabat de crear l'horari.

Pseudocodi del backtracking cronològic:

```
creaHorari(iterador per sessions, horari,HashMap<Sessió, ArrayList<possibilitats>> pos)
if(iterador == última sessió ) return true "S'HA FET L'HORARI"
for(d -> dies_possibles)
  for(h->d.hores_possibles)
     for(a->h.aules_possibles){
```

```
          if(no està assignat aquest slot al horari){
              if(es compleixen les restriccions){
                  Assignar a l'horari a l'slot a,d,h durant la duració de la sessió
                  propagarPossibilitats()
                  if(propagarPossibilitats() fa que alguna sessió no tingui opcions)
                      Esborrar assignació feta a a,d,h durant la duració de la sessió
                      continuarBucle()

                  else
                    Esborrar de possibilitats la sessió assignada i la seva informació
                    creaHorari(iterador+1, horari, pos)
                    if(creaHorari() acaba per a totes les sessions) return true "S'HA
FET L'HORARI"
                    else {
                      Recuperar la sessió i la seva informació, posar-la a possibilitats
                      Esborrar assignació feta a a,d,h durant la duració de la sessió
                      continuarBucle()
                      }
return false "NO ES POT FER AIXÍ"
```

Forward Checking

Tal i com hem comentat anteriorment en l'explicació del backtracking cronològic el nostre algorisme no itera envers a totes les possibles aules de la universitat, tots els dies i totes les hores; sinó que només ho fa a través de les que la sessió té com a possibles. Aquestes aules a hores a dies possibles s'inicialitzen en la primera poda de domini. Si aquesta primera ja ens crea una sessió sense possibilitats, sabem que l'horari no es pot efectuar. A més, després de cada assignació fa una poda de domini en el que es propaguen les possibilitats. Així a mesura que avanci l'algorisme farem menys comprovacions i menys iteracions.

L'algorisme funciona de la següent manera:

Després de fer una assignació hem en el backtracking cronològic cridem a la propagació de posibilitats amb l'aula, el dia, la hora de l'assignació acabada de realitzar, la sessió i l'estructura de dades on guardem totes les possibilitats de totes les sessions.

Llavors per a cada sessió efectuem les diferents restriccions i si alguna ens dóna que l'aula que ens ha entrat com a paràmetre no hauria d'estar com a possibilitat de la sessió la esborrem. Parem de mirar restriccions i continuem. Si en un moment arribem a tenir que una sessió no té cap possibilitat parem i retornem que no es pot seguir fent l'horari d'aquesta manera. Altrament acabem retornant les possibilitats amb la propagació efectuada.

Pseudocodi del forward checking:

```
propagarPossibilitats(aula,dia,hora,HashMap<Sessió, ArrayList<possibilitats>> pos)
for(i->totes les sessions)
   forAll restriccions:
        comprovarRestriccio()
        if(comprovarRestriccio() ens dóna que hem d'esborrar l'aula del dia i hora)
            La esborrem dintre les possibilitats i parem de comprovar restriccions
        if(la sessió acaba no tenint cap possibilitat)
            return false "no es pot continuar"
 return true "es pot continuar"
```

Estructura de Dades

Les dues estructures principals que fem servir en el nostre projecte són les següents:

- La d'horari
- La de sessions i possibilitats

El nostre horari és un una matriu 3D d'assignacions on cada eix representa els dies, les hores i les aules que ja s'han assignat.

```
  private Assignacio[][][] horari;
```

Per a guardar el domini de cada sessió de cada grup fem servir un HashMap on la clau és una sessió d'un grup que hem d'assignar i el valor és una llista d'aules possibles representades com a enters.

```
  HashMap<SessioGrup,ArrayList<ArrayList<ArrayList<Integer>>>> possibilitats
```

La llista està feta de manera que pot no haver-hi res, de manera que no és una via possible o haver-hi una aula on cada vegada que interioritzés més per l'estructura de dades cada índex representaria el dia i l'hora que és possible.