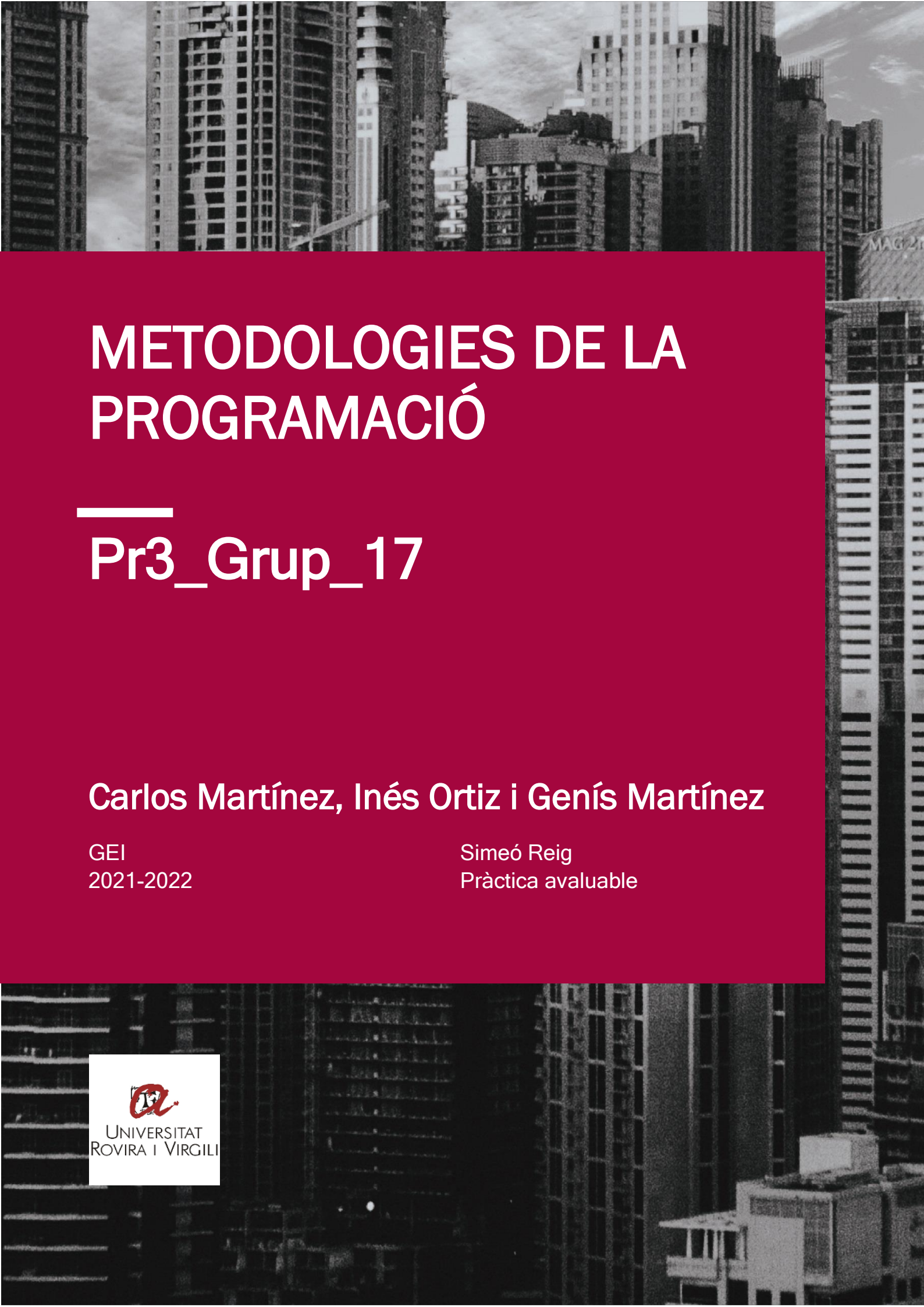


METODOLOGIES DE LA PROGRAMACIÓ

Pr3_Grup_17

Carlos Martínez, Inés Ortiz i Genís Martínez

GEI
2021-2022

Simeó Reig
Pràctica avaluable

UNIVERSITAT ROVIRA I VIRGILI

Índex

1.- Disseny de l'algorisme i implementació

Per a realitzar aquesta pràctica hem dissenyat dos tipus d'algoritmes com a solució. El primer d'ells proposa una solució de formà àvida i el segon proposa una solució utilitzant el backtracking. El primer algoritme que hem dissenyat, l'àvid, ens permet trobar una solució al problema plantejat, però no té perquè ser la solució més òptima. Les passes que hem seguit per a trobar aquesta solució han sigut les següents:

1. Partim d'una solució buida
2. Hem identificat els possibles candidats a considerar i anem triant el que maximitza els punts
3. Continuem fins que arribem a la solució

Per altre banda, el segon tipus d'algorisme que hem dissenyat utilitzant backtracking té un altre finalitat. Aquest tipus algorisme s'aplica quan es vol explorar possibles solucions a un problema i volem quedar-nos amb la millor o alguna de factible. El que caracteritza a aquesta estratègia és el retrocés. En a cerca exhaustiva valorem les possibles solucions que es poden donar. Al dissenyar un algoritme mitjançant backtracking hem de considerar els següents elements:

1. Representació de la solució en una llista ordenada
2. Una funció objectiu per determinar si la llista a analitzar és una solució
3. Utilitzar restriccions als candidats per omplir la llista ("NA")
4. Una funció de poda per eliminar parts de l'arbre de cerca
5. Organització del problema en un arbre
6. Construir la solució del problema en diferents etapes
7. En cada pas s'escull un candidat i s'afegeix a la solució i s'avança en la solució parcial
8. Si no es possible continuar en la construcció cap a una solució completa, abandonem aquesta i l'última component es canvia per un altre valor
9. Si no queden més valors per provar, es retrocedeix al candidat anterior, s'elimina i es selecciona un altre candidat

Estructura genèrica per a un algoritme backtracking:

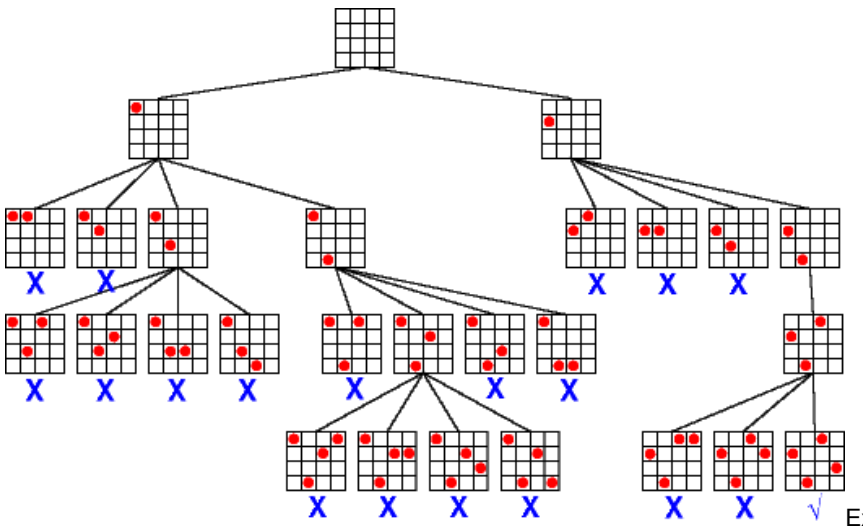
```
Solució [i] ∈ Si per i = 1, 2,..., n
funció BACKTRACKING_REC (k, solució [n])
per j ∈ Si
si (PODA (k, j, solució) == cert) fer
        sol [k] = j
        si (TEST_SOL (soluci) == cert) fer
                retorna solució
        si (k < n)
                BACKTRACKING_REC (k+1, solució [n])
```

L'eficiència d'un algoritme backtracking sol ser de tipus exponencial a^n.
L'eficiència depèn de:

- La ramificació de l'arbre
- Del temps d'execució de la funció sol·lució
- Del temps d'execució de la funció poda
- Del estalvi d'utilitzar la poda

Hem pogut comprovar que les funcions de poda no són molt eficients.

Entre els problemes típics que es poden resoldre fàcilment amb les tècniques de backtracking tenim: les voltes del caball, sudoku, laberint, sopa de lletres, problema de les parelles, formació d'una paraula amb "n" cubs, problema del dominó, problema del repartiment de les monedes, coloració de grafs, problema del viatjant sobre grafs dirigits, etc...

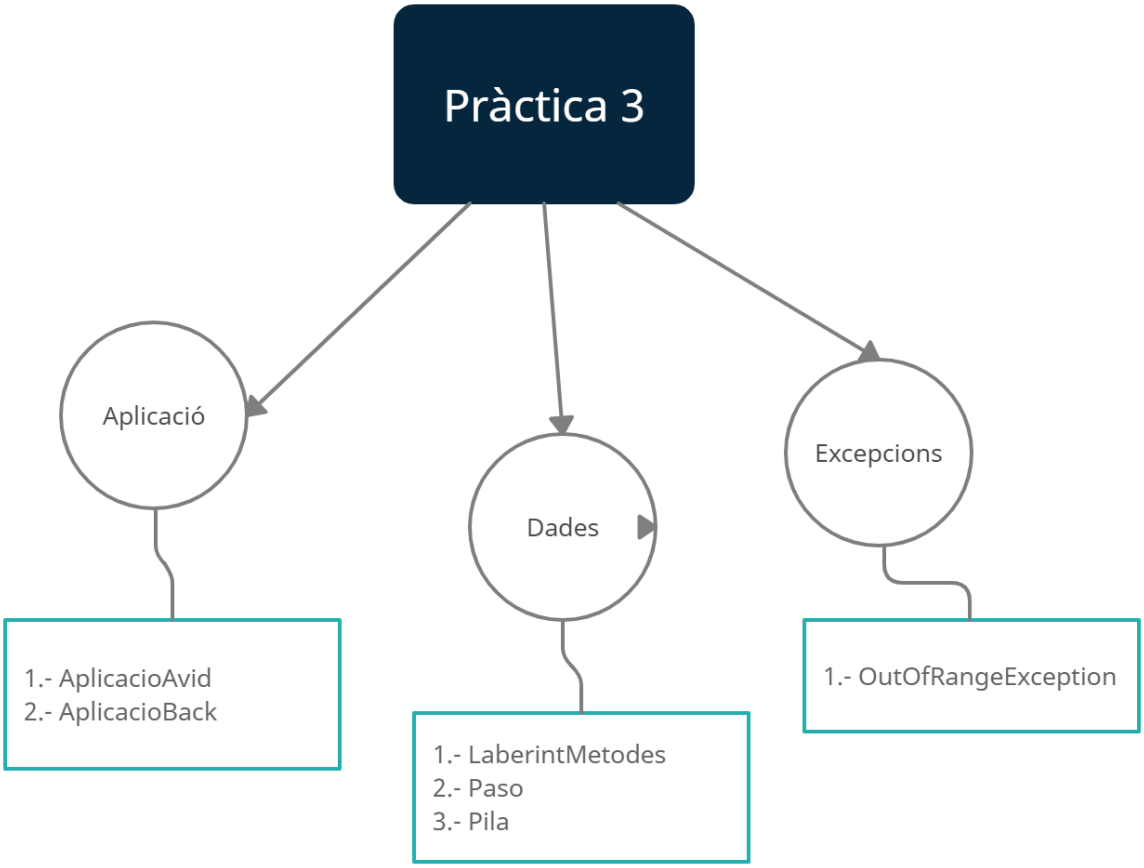


Exemple del que podria ser un arbre

2.- Implementació

Una vegada amb els dos algoritmes dissenyats arriba l'hora d'implementar-los.

Diagrama de classes:



La pràctica està distribuïda en 3 packages diferents:

1. Aplicació
2. Dades
3. Excepcions

Dintre del package "Aplicació" hem guardat els dos programes principals per a executar els dos algoritmes. La classe "AplicacioAvid" executa el codi de l'algorisme àvid i la classe "AplicacioBack" executa el codi de l'algorisme que utilitza la tècnica del backtracking.

Dins del package “Dades” hem guardat les classes que utilitzem per gestionar la pila, els passos del backtracking i per últim la classe on desenvolupem tots els procediments (“LaberintMetodes”). Ademés, dins el package “Exceptions” hem tractat les excepcions.

Alguns dels procediments que hem implementat són els següents:

- afegirLab (String[][] valors)
- operacio (int valorActual, int i, int j)
- finalJoc (int valorActual, int i, int j)
- comprobar (int valorRestant, int casellaF, int casellaC)
- pasoBackT (int valorActual, int i, int j)
- Entre d’altres...

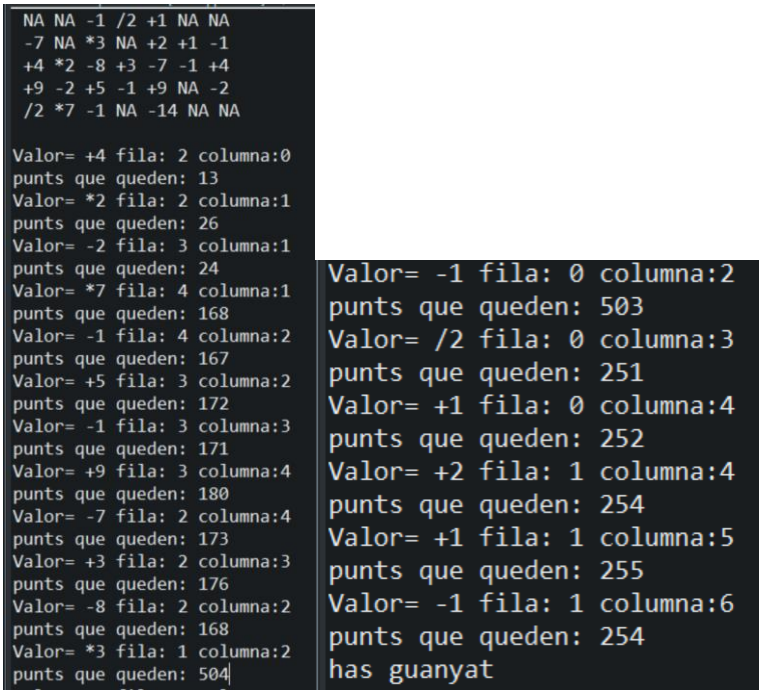
Cal afegir que finalment vam decidir fer un mètode backtraking nou perquè el que teniem implementat no acabava de funcionar correctament, el mètode de backtraking vell està comentat al final de la classe LaberintMetodes.

3.- Joc de proves

Joc de proves per el mètode àvid:

Primerament vam probar amb el laberint del fitxer “Laberint.txt”.

La solució que esperavem es que finalitzes en la posició (fila=1, columna=6) amb un valor final de 254 punts.



Joc de proves backtraking:

La segona prova la varém fer amb el laberint del fitxer “Laberint2.txt”.

La solució que esperavem es que finalitzes en la posició (fila=0, columna=0) amb un valor final de 2 punts.

```
 -2 /8 NA NA NA +1 /7
 -7 -8 NA +8 *2 /2 -1
 +1 *3 +2 /5 NA NA +2
 /6 *1 -3 NA *1 *2 +5
 *5 -2 +7 -1 +3 /2 -1

Valor: +3, fila= 4, col= 4
        Act: 3
Valor: *1, fila= 3, col= 4
        Act: 3
Valor: *1, fila= 3, col= 4
        Act: 3
Valor: *2, fila= 3, col= 5
        Act: 6
Valor: /2, fila= 4, col= 5
        Act: 3
Valor: /2, fila= 4, col= 5
        Act: 3
Valor: /2, fila= 4, col= 5
        Act: 3
Valor: -1, fila= 4, col= 6
        Act: 2
Valor: +5, fila= 3, col= 6
        Act: 7
Valor: +2, fila= 2, col= 6
        Act: 9
Valor: -1, fila= 1, col= 6
        Act: 8
Valor: /7, fila= 0, col= 6
        Act: 1
Valor: /7, fila= 0, col= 6
        Act: 1
Valor: +1, fila= 0, col= 5
        Act: 2
Valor: /2, fila= 1, col= 5
        Act: 1
Valor: *2, fila= 1, col= 4
        Act: 2
Valor: +8, fila= 1, col= 3
        Act: 10
Valor: /5, fila= 2, col= 3
        Act: 2
Valor: /5, fila= 2, col= 3
        Act: 2
Valor: +2, fila= 2, col= 2
        Act: 4
Valor: -3, fila= 3, col= 2
        Act: 1
Valor: -3, fila= 3, col= 2
        Act: 1
Valor: +7, fila= 4, col= 2
        Act: 8
Valor: +7, fila= 4, col= 2
        Act: 8
Valor: -2, fila= 4, col= 1
        Act: 6
Valor: *1, fila= 3, col= 1
        Act: 6
Valor: *3, fila= 2, col= 1
        Act: 18
Valor: -8, fila= 1, col= 1
        Act: 10
Valor: /8, fila= 0, col= 1
        Act: 1
Valor: /8, fila= 0, col= 1
        Act: 1
Valor: +1, fila= 0, col= 0
        Act: 2
Solution found
```