

Pràctica 5

Objectius:

1. Resolució d'un exercici aplicant la tècnica del Backtracking.
2. Resolució d'un exercici aplicant la tècnica de Voraç.

En aquesta pràctica es resoldrà un mateix exercici aplicant les dues tècniques. Dues solucions alternatives pel mateix enunciat.

Durada: Dues sessions

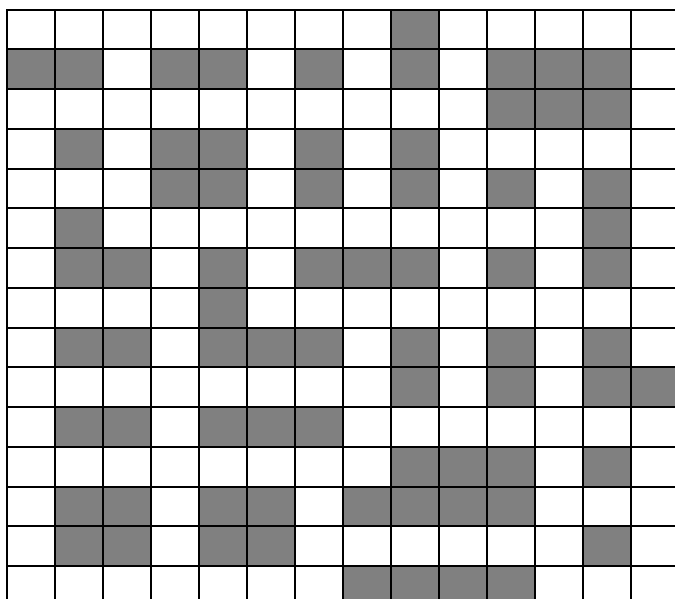
Llenguatge: Java utilitzant IntelliJ o Eclipse

Lliurament: penjar el projecte comprimit en format ZIP a l'aula virtual

Data Lliurament: diumenge 8 de desembre

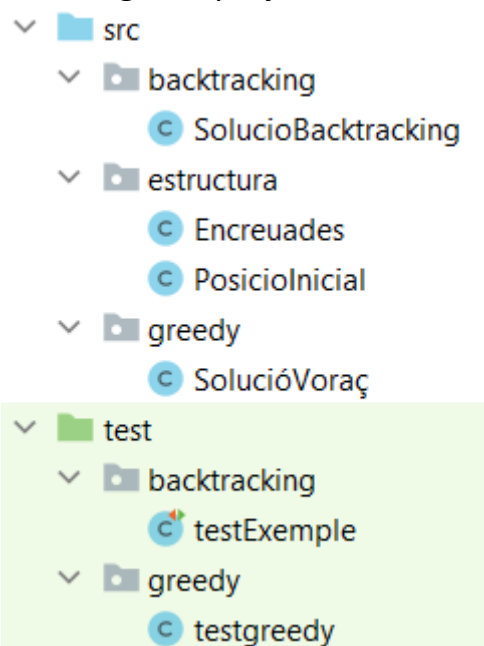
Enunciat

Resoldrem un encreuat que consisteix en col·locar uns certs elements (paraules o números) en una reixa.



CONTEMPLAREIS, ORDENANCISMO,
DESVINCULAR, APELTONAD,
COMPETÍAIS, PANTÓGRAFO,
AUTÓCTONO, MEZQUINEN,
LINOLEUM, PATAPLUM, RETRANCA,
ZOLLIPAS, PERECÍA, SACONEA,
CUORUM, LIMOSA, STABAT, ACLLA,
DALLA, VIOLO, ZAMPA, CUCA, MEAN,
NOS, ODA, SOL.

A l'aula virtual trobareu el següent projecte:



Al Package estructura trobareu dues classes ja implementades:

- Encreuades és una classe amb dos atributs finals que són arrays bidimensionals del tipus char que defineixen el problema a resoldre: l'atribut puzzle és la reixa a omplir i l'atribut items són els elements (paraules o números). Al constructor s'inicialitzarà un l·listat ordenat amb totes les ubicacions disponibles que són objectes del tipus PosicioInicial.
- PosicioInicial és una classe que conté la informació de les ubicacions on podem posar els elements, exactament coneixem la posició inicial (row,col), la longitud i la direcció (horitzontal o vertical).

Per exemple, la primera posició horitzontal que trobem és:

```
initRow=0
initCol=0
length=8
direcció=H
```

Un segon exemple, la primera posició vertical que trobem és:

```
initRow=2
initCol=0
length=13
direcció=V
```

Nota molt important: com podeu veure aquest problema té una única solució, per poder aplicar tècniques d'optimització hi haurà més elements inicials que ubicacions disponibles. Per escollir entre dos elements utilitzarem el valor decimal segons el codi ASCII:

Valor Decimal	char	Valor Decimal	char	Valor Decimal	char	Valor Decimal	char
48	0	65	A	75	K	85	U
49	1	66	B	76	L	86	V
50	2	67	C	77	M	87	W
51	3	68	D	78	N	88	X
52	4	69	E	79	O	89	Y
53	5	70	F	80	P	90	Z
54	6	71	G	81	Q		
55	7	72	H	82	R		
56	8	73	I	83	S		
57	9	74	J	84	T		

Per utilitzar-ho només cal sumar els caràcters en una variable entera:

```
char lletraA = 'A';
```

```
int valor = lletraA;
```

Anàlisi del problema

Prèviament a la codificació, obligatòriament s'ha de fer l'anàlisi del problema.

- 1.- Per què l'esquema de backtracking és aplicable per a resoldre aquest enunciat?
- 2.- Indica quina pregunta et fas en cada nivell de l'arbre. Quines són les possibles respostes d'aquesta pregunta (domini)?
- 3.- Quin és el criteri per determinar si una decisió és o no acceptable? (pregunta relacionada amb el mètode acceptable)
- 4.- Quin és el criteri per determinar si un conjunt de decisions és o no completable?

5.- Quin és el criteri per determinar si un conjunt de decisions són o no solució? (pregunta relacionada amb el mètode esSolució)

6.- Dibuixeu l'espai de cerca del problema, és a dir l'arbre que ha de recórrer la tècnica del backtracking, indica quina és l'alçada i l'amplada i si són valors exactes o valors màxims. Usaràs marcatge?

7.- Quin és el criteri per determinar si una solució és o no millor a una altra ja trobada prèviament?

Solució Backtracking

Al projecte trobaràs la classe SolucioBacktracking, dins d'aquesta classe ja disposes del esquema recursiu backUnaSolucio que troba una solució.

Atributs de classe:

- Repte: és un objecte de la classe Encreuades que s'instancia al constructor.
- A partir del teu anàlisi del problema defineix els atributs que siguin necessaris (*).

Mètodes públics:

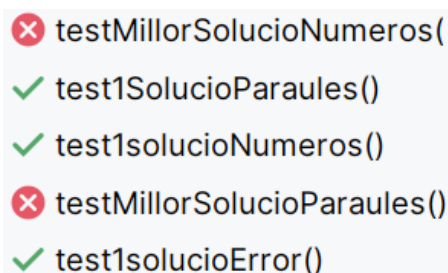
- getMillorSolucio: retorna un array bidimensional amb la millor solució trobada.
- Start: mètode instància els atributs necessaris (*) per executar el backtracking, i que segons un paràmetre d'entrada executarà el mètode backUnaSolucio o backMillorSolucio.

1. Mètodes privats per trobar una solució al problema:

Ja teniu implementat el mètode `backUnaSolucio`, que és un mètode recursiu amb un esquema de backtracking per trobar una solució. No modifiqueu aquest mètode. Per tal que funcioni correctament, cal implementar els següents mètodes auxiliars:

- `acceptable`: retornar un booleà indicant si un element es pot assignar a una ubicació (a partir d'una posició inicial).
- `anotarASolucio`: modifica la reixa per afegir l'element.
- `esSolucio`: retornar un booleà indicant si la reixa és una solució al problema.
- `desanotarDeSolucio`: modifica la reixa per esborrar l'element (no podeu esborrar caràcters d'altres elements, podeu crear un mètode privat per consultar si un caràcter es pot eliminar).

Podeu comprovar aquests mètodes amb `testExemple`:



A screenshot of a test runner interface showing the results of five test cases. Each line consists of a status icon (a red 'X' for failure or a green checkmark for success) followed by the test method name. The results are: `testMillorSolucioNumeros()` failed (red X), `test1SolucioParaules()` passed (green checkmark), `test1solucioNumeros()` passed (green checkmark), `testMillorSolucioParaules()` failed (red X), and `test1solucioError()` passed (green checkmark).

```
✗ testMillorSolucioNumeros()  
✓ test1SolucioParaules()  
✓ test1solucioNumeros()  
✗ testMillorSolucioParaules()  
✓ test1solucioError()
```

2. Mètodes privats per trobar la millor solució:

Ara cal implementar el mètode `backMillorSolucio`: mètode recursiu amb el esquema de backtracking per trobar la millor solució, observant el mètode `backUnaSolució` i els esquemes estudiants a teoria, crea un mètode recursiu que busqui totes les solucions i guardi la millor. A més d'utilitzar els mètodes privats anteriors, caldrà crear-ne de nous:

- `calcularFuncioObjectiu`: retorna un int amb el valor de una solució.
- `guardarMillorSolució`: si la solució actual és millor, guardem una còpia.

Podeu comprovar aquesta classe amb `testExemple`:

✓ testMillorSolucioNumeros()	60 ms
✓ test1SolucioParaules()	6 ms
✓ test1solucioNumeros()	2 ms
✓ testMillorSolucioParaules()	17 ms
✓ test1solucioError()	4 ms

Com pots veure aquelles execucions dels mètodes backMillorSolucio necessiten més temps, donat que recorren tot l'arbre de manera exhaustiva mirant totes les solucions per determinar l'optima.

Solució Voraç

Implementa la classe utilitzant la tècnica greedy per intentar trobar una solució.

Podeu comprovar aquesta classe amb testgreedy, no passa res si el vostre mètode no pot resoldre algun o tots dos exemples:

✗ testSolucioParaules()	43 ms
✓ testsolucioNumeros()	3 ms