Algorítmica avançada
Arcade

Departament d'Enginyeria
La Salle - Universitat Ramon Llull
3 desembre de 2021

Índex

1 Introducció

Tots tenim les nostres aficions, però els programadors solem coincidir en el fet que ens agraden els videojocs. I és que, a banda d'ajudar-nos a passar l'estona, també poden ser una font d'inspiració per projectes personals.

En aquesta ocasió, i moguts per la nostàlgia, ens hem proposat crear-ne un inspirat per les màquines recreatives clàssiques (també conegudes com a *Arcade*). Identificant que molts jocs de l'època es basaven a tenir un mapa en forma de graella, se'ns ha ocorregut barrejar dues mecàniques en aquesta línia: Escapar d'un laberint, i resoldre una sopa de lletres.

Ara mateix tenim implementat un sistema que és capaç de generar i visualitzar nivells aleatòriament, però volem complementar-lo amb diferents algorismes que siguin capaços de solucionar-los per més endavant ser capaços de valorar-ne la dificultat.

Així doncs, se us encarrega explorar, analitzar i comparar diferents enfocs basats en optimització combinatòria per solucionar els dos minijocs del sistema.

2 Minijocs

En aquest apartat es descriu la lògica associada a cadascun dels dos minijocs.

2.1 Laberint

El primer minijoc a resoldre és el laberint. En aquest, el jugador parteix d'una posició d'origen i ha de trobar el camí més curt per arribar a la posició marcada com a destí. Cal tenir present que no podrem situar-nos en posicions on hi hagi parets i que només ens podem moure en les quatre direccions cardinals.

La Figura 1 mostra un exemple de laberint, podent-se'n observar a la Figura 2 dues solucions: una de costosa en vermell i la més eficient en verd.

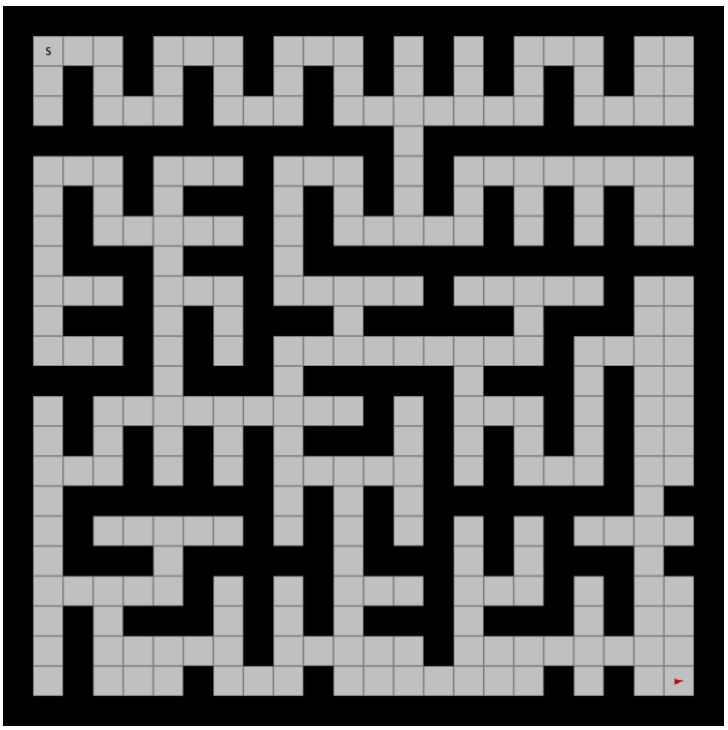

Figura 1. Representació visual d'un laberint, amb l'origen a la part superior esquerra i el destí a la inferior dreta.

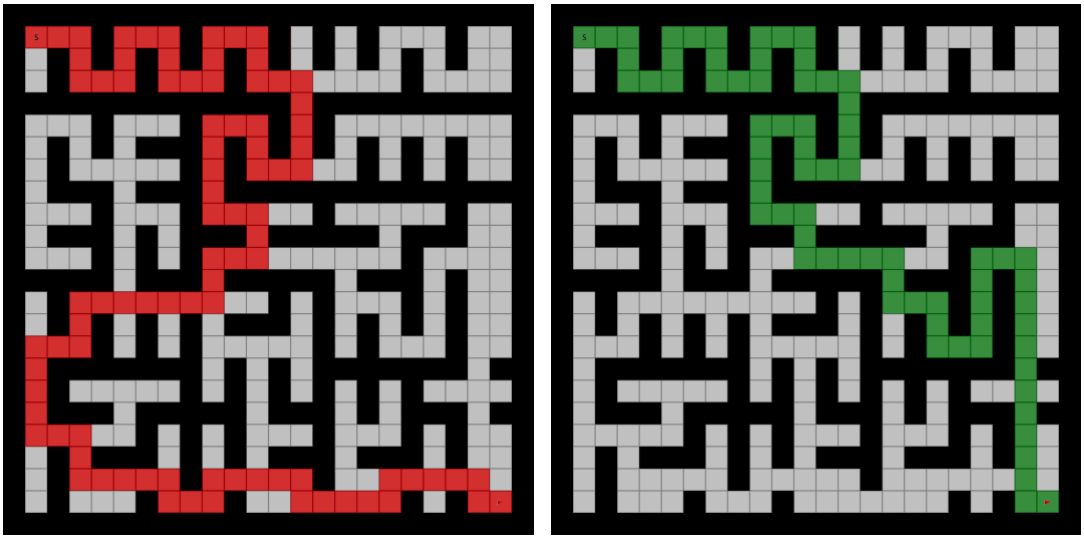

Figura 2. Representació visual de dues solucions: En vermell, una de poc eficient. En verd, la més òptima.

2.2 Sopa de lletres

El segon minijoc consisteix a resoldre una sopa de lletres simplificada. Concretament, cal trobar una paraula determinada a una graella de caràcters. Sabem que les paraules poden aparèixer en horitzontal, vertical i diagonal, però sempre en el sentit natural de lectura.

La Figura 1Figura 3 exemplifica un possible *input* on es demana trobar la paraula “BRANCH”, mentre que la Figura 4 en mostra la solució.

I	A	E	E	H	S	E	O	L	E	K	A
B	G	L	A	I	U	A	M	K	H	R	E
A	R	O	G	D	A	E	U	T	U	O	L
C	F	A	Z	E	A	L	R	S	K	W	P
K	A	J	N	I	T	S	O	A	R	N	A
T	F	K	O	C	P	I	D	E	A	U	B
R	N	A	E	N	H	O	C	O	S	T	O
A	H	R	X	Q	U	I	C	K	E	A	U
C	T	A	S	M	E	R	G	E	A	N	N
K	H	S	B	U	C	K	E	T	S	E	D
M	R	T	C	L	O	C	S	L	Y	R	O
W	K	S	O	R	T	G	N	A	P	N	V

Figura 3. Graella d'exemple pel problema de la sopa de lletres.

I	A	E	E	H	S	E	O	L	E	K	A
B	G	L	A	I	U	A	M	K	H	R	E
A	R	O	G	D	A	E	U	T	U	O	L
C	F	A	Z	E	A	L	R	S	K	W	P
K	A	J	N	I	T	S	O	A	R	N	A
T	F	K	O	C	P	I	D	E	A	U	B
R	N	A	E	N	H	O	C	O	S	T	O
A	H	R	X	Q	U	I	C	K	E	A	U
C	T	A	S	M	E	R	G	E	A	N	N
K	H	S	B	U	C	K	E	T	S	E	D
M	R	T	C	L	O	C	S	L	Y	R	O
W	K	S	O	R	T	G	N	A	P	N	V

Figura 4. Solució per la graella d'exemple, mostrant la posició (en diagonal) de la paraula "BRANCH".

3 Requeriments

En aquest apartat es presenten les metes principals de la pràctica, i se'n descriuen els resultats esperats.

3.1 Objectius

Tot i que hi ha diverses alternatives, es demana resoldre cadascun dels dos minijocs **amb dues estratègies diferents com a mínim** basades en les tècniques de resolució de problemes combinatoris tractades a l'assignatura (és a dir, a escollir entre backtracking, branch & bound i greedy).

L'objectiu principal de la pràctica és comparar el rendiment de les diferents solucions que es proposin, pel que cal aplicar millores d'eficiència (marcatge, PBMSC...) allà on sigui possible.

En la mateixa línia, per facilitar l'anàlisi de resultats és altament recomanable mantenir diferents versions d'una solució per poder-les examinar més en detall. Alguns **exemples** de comparatives interessants inclouen:

- **Força bruta *vs* backtracking**, per veure l'efecte de la poda.
- **Abans *vs* després d'aplicar marcatge**, per apreciar la millora.
- **Backtracking amb PBMSC *vs* branch & bound**, per veure l'efecte de canviar l'ordre en que es recorre l'espai de solucions.
- **Ús de diferents heurístiques** en aquells algorismes que en facin servir, per veure'n la utilitat.

Per descomptat, n'hi ha moltes més, i sou lliures d'escollir les més rellevants segons les solucions que hàgiu decidit implementar per cada minijoc.

3.2 Codi

El projecte que entregueu ha de implementar les diferents estratègies escollides correctament, però no hi ha restriccions a nivell de format d'entrada ni de sortida de dades. Podeu deixar els mecanismes d'avaluació de rendiment que hàgiu escollit (per exemple, la mesura del temps d'execució).

Tot i així, és recomanable estructurar el codi de forma que sigui fàcil canviar-ne el funcionament (o escollir la funcionalitat que s'executa). Això es pot aconseguir de moltes maneres, com per exemple definint paràmetres d'entrada per consola de comandes, codificant un petit menú, fent servir constants, deixant comentaris al procediment principal...

3.3 Memòria

La memòria hauria de tenir els següents apartats, o un conjunt equivalent que cobreixi el mateix contingut:

- **Portada** (amb el número de grup i els noms complets i *login* dels membres).
- **Índex numerat**.
- Explicació del **llenguatge de programació escollit**, amb els avantatges que proporciona i, si es considera necessari, possibles inconvenients.
- Explicació del **disseny dels diferents algorismes** i estratègies (poda, heurístiques, millores...) segons les fases de la metodologia de resolució que correspongui.
- **Anàlisi de resultats**, tenint en compte *inputs* de diferents mides i explicant com s'han obtingut les mesures. Cal realitzar les comparacions que es considerin necessàries i generar gràfiques.
- **Problemes observats** i la seva solució.
- **Dedicació** total en hores. Inclogueu també un desglossament temàtic en les categories que considereu necessàries. Alguns exemples inclouen: Comprensió, investigació, disseny, implementació, anàlisi de resultats, documentació...
- **Conclusions**, també a nivell personal però sobretot a nivell tecnològic.
- **Bibliografia** segons la norma ISO 690 o APA 7th.

La memòria ha d'estar escrita en **llenguatge formal**. Aquesta és igual o més important que la implementació del treball, ja que reflexa el coneixement adquirit a la pràctica i l'assignatura. Així doncs, es tindrà en compte que el seu contingut **prioritzi la qualitat abans que la quantitat**.

L'apartat d'anàlisi de resultats és crític per aprovar la pràctica, i determinarà una part substancial de la nota. Heu de ser capaços de dur a terme mesures correctes de rendiment, així com de identificar comparacions rellevants a partir d'elles i d'aquestes extreure'n conclusions substancials.

4 Consideracions

En aquest apartat es realitzen una sèrie de puntualitzacions respecte la pràctica i el seu desenvolupament.

4.1 Llenguatge de programació

La pràctica es pot implementar en qualsevol llenguatge de programació, a escollir pel grup. Si es programa en C, aquesta haurà de compilar, executar i funcionar correctament al servidor de la universitat (matagalls).

Si es desenvolupa en Java podreu fer ús d'un software auxiliar que es detalla al final del document, en l'Annex - Material disponible. En cas contrari, caldrà que feu servir fitxers d'entrada i que n'expliqueu el format en un README.

4.2 Grups

La pràctica es pot realitzar de forma individual o en parelles. Notifiqueu la conformació dels grups de treball per correu electrònic (Xavi Solé -xavier.sole@salle.url.edu) amb data límit 10 de desembre.

4.3 Eines Atlassian

Un cop presentat aquest enunciat, i després d'un petit marge de temps per la conformació de grups de treball, se us assignarà un repositori al servidor Bitbucket de la universitat pel control de versions amb git. Independentment del llenguatge escollit i de la mida del grup **cal fer-lo servir de forma regular durant el desenvolupament de la pràctica**.

Si desitgeu fer ús d'altres eines d'Atlassian com Jira o Confluence, podeu demanar-ho per correu al professor de l'assignatura.

4.4 Detecció de còpia

Una còpia s'interposa en l'aprenentatge dels alumnes, alhora que és una falta de respecte als companys que han dedicat temps i esforç a la realització del treball.

Una pràctica es classifica com a activitat **altament significativa.** La còpia, parcial o total, d'un company o d'internet es considerarà **acció premeditada**. Per aquests motius, aplicant la normativa de còpies de la universitat, es considerarà una **falta molt greu**.

5 Entrega

En aquest apartat es descriu tot allò relacionat amb l'entrega de la pràctica, des del format fins la data límit i mecanismes de recuperació.

5.1 Format

Cal entregar **dos fitxers distints** al pou corresponent de l'eStudy: El **PDF** de la memòria i un fitxer comprimit **ZIP** que contindrà:

- README: Fitxer **TXT** o **MD** on s'expliqui **TOT** allò relacionat amb el *testeig* del codi, de la forma més detallada possible (llenguatge, versió del llenguatge, IDE utilitzat, instruccions de compilació/execució, instruccions d'ús...)
- Carpeta amb el codi i/o estructura del projecte generat per l'IDE. **Ha de correspondre al que hi hagi al repositori de Bitbucket**.

No s'acceptaran entregues on la memòria estigui comprimida dins el ZIP.

5.2 Data límit

La pràctica es publicarà un cop establerta la base de coneixements necessària per començar a implementar-la, i s'entregarà com a molt el dia **16 de gener de 2022.**

No s'acceptarà cap entrega posterior.

5.3 Recuperació

Si l'entrega no es realitza a temps (NP) o se suspèn, la pràctica haurà de ser entregada de nou com a molt tard el **20 de març del 2022,** amb nota màxima de 7.

La darrera oportunitat serà durant la convocatòria extraordinària, amb el **3 de juliol de 2022** com a data límit i nota màxima de 5.

No s'acceptarà cap entrega posterior.

Annex - Material disponible

Tot i que no hi ha restriccions pel que fa al llenguatge utilitzat per implementar la pràctica, des de l'assignatura es proporciona una base de software auxiliar en Java. Concretament, aquesta base implementa les següents funcionalitats:

- Generació aleatòria (però configurable) de *inputs* pels minijocs.
- Visualització gràfica dels *inputs* i solucions dels minijocs. Inclou la possibilitat d'aturar l'execució durant uns mil·lisegons per observar millor el procés.
- Validació de la solució trobada (en cas de ser un problema d'optimització, **NO** es comprova si és la millor).

Per fer-la servir, disposeu dels següents fitxers:

- **Arcade.jar**: Llibreria Java que conté tot el codi auxiliar.
- **Arcade-Javadoc.zip**: Documentació de les classes, interfícies i enumeracions amb les que interaccionareu si feu ús de la llibreria, en format Javadoc estàndard.
- **Arcade-UML.mdj**: Diagrama de classes UML representant tot allò amb el que interaccionareu si feu ús de la llibreria. Per comoditat, també s'inclou a la Figura 5.

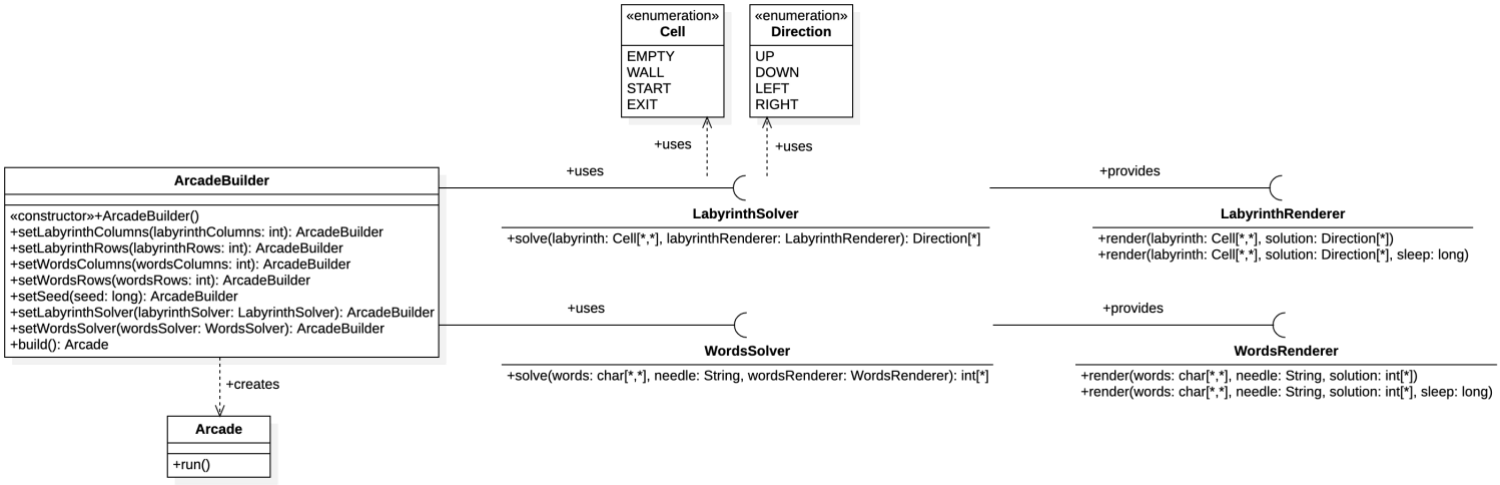

Figura 5. Diagrama UML de les classes, interfícies i enumeracions públiques de la llibreria auxiliar.

Tot i que disposeu de la documentació prèviament esmentada, a continuació es descriuen els passos a seguir per usar la llibreria, i s'inclou un exemple de codi:

1. Definir com a mínim una classe que implementi la interfície LabyrinthSolver i una altra que implementi la interfície WordsSolver. És recomanable tenir una implementació diferent per cada estratègia (backtracking, branch & bound...).
2. Fer ús de la classe ArcadeBuilder per configurar i obtenir una instància de la classe principal de la llibreria (Arcade).
3. Executar el mètode arcade.run().

```
Arcade arcade = new ArcadeBuilder()
        .setLabyrinthColumns(24)
        .setLabyrinthRows(24)
        .setWordsColumns(12)
        .setWordsRows(12)
// Opcional, per fixar un input en comptes d'obtenir-ne un d'aleatori
        .setSeed(42)
// DemoLabyrinthSolver implementa LabyrinthSolver
        .setLabyrinthSolver(new DemoLabyrinthSolver())
// DemoWordsSolver implementa WordsSolver
        .setWordsSolver(new DemoWordsSolver())
        .build();

 arcade.run();
```