



POLITECNICO
DI TORINO



e-Lite



Recursion Exercise – Magic Square

Tecniche di Programmazione – A.A. 2019/2020



Magic Square

- ▶ A magic square is $n * n$ grid (where n is the number of cells on each side) filled with distinct positive integers in the range $1, 2, \dots, n^2$ such that each cell contains a different integer and the sum of the integers in each row, diagonal and column is equal
- ▶ The sum is called *magic constant*.

| | | | | |
|------|------|------|------|------|
| 2 | 7 | 6 | → 15 | |
| 9 | 5 | 1 | → 15 | |
| 4 | 3 | 8 | → 15 | |
| ↙ 15 | ↓ 15 | ↓ 15 | ↓ 15 | ↘ 15 |

Magic Square

- ▶ *magic constant* $M = \frac{n(n^2+1)}{2}$
 - ▶ For normal magic squares of orders $n = 3, 4, 5, 6, 7,$ and $8,$ the magic constants are, respectively: 15, 34, 65, 111, 175, and 260
- ▶ *There exist exact methods for constructing magic squares of even or odd order...*
- ▶ *... while we will use recursion*



Design tips

Recursion

Analizzare il problema

| | | |
|---|--|--|
| 1 | | |
| | | |
| | | |

| | | |
|---|---|--|
| 1 | 2 | |
| | | |
| | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| | | |
| | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | | |
| | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | |
| | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| | | |

Analizzare il problema

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

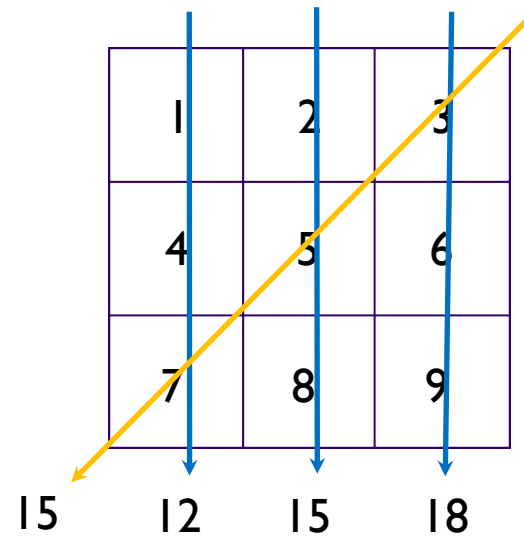
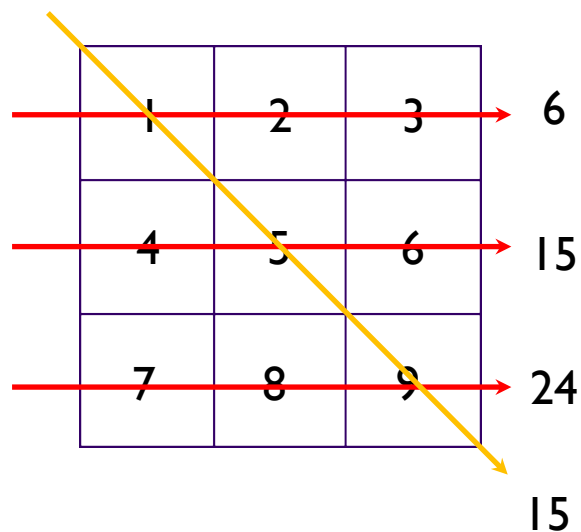
| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Analizzare il problema

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

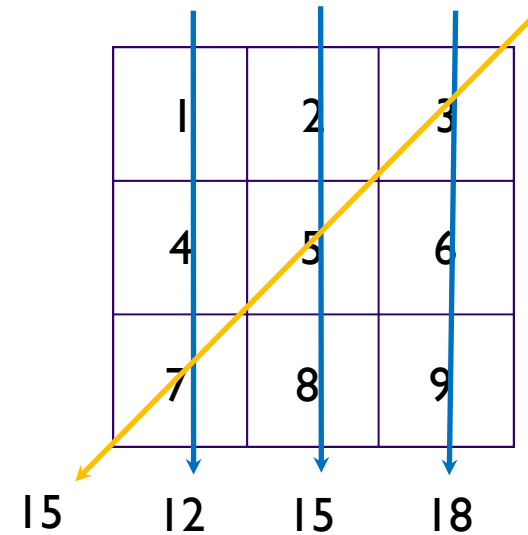
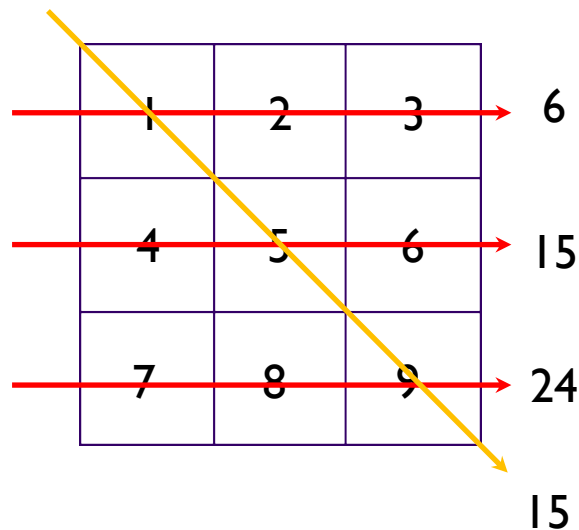


Analizzare il problema

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |



FAILED

Analizzare il problema

- ▶ Come imposto in generale la ricorsione?
- ▶ Che cosa mi rappresenta il "livello"?
- ▶ Qual è il livello massimo?
- ▶ Com'è fatta una soluzione parziale?
- ▶ Com'è fatta una soluzione completa?
- ▶ Come viene avviata la ricorsione (livello 0)?
- ▶ Come genero Parziale($i+1$) partendo da Parziale(i)?

Analizzare il problema

- ▶ **Come imposto in generale la ricorsione?**
 - ▶ Ad ogni passo inserisco un numero $1, 2, \dots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle

Analizzare il problema

- ▶ **Come imposto in generale la ricorsione?**
 - ▶ Ad ogni passo inserisco un numero $1, 2, \dots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle
- ▶ **Che cosa mi rappresenta il "livello" i-esimo?**
 - ▶ Il livello rappresenta la casella i-esima del quadrato che devo riempire

Analizzare il problema

- ▶ **Come imposto in generale la ricorsione?**
 - ▶ Ad ogni passo inserisco un numero $1, 2, \dots, n^2$ nella prima casella libera del quadrato. Delego al passo successivo il riempimento delle successive caselle
- ▶ **Che cosa mi rappresenta il "livello" i -esimo?**
 - ▶ Il livello rappresenta la casella i -esima del quadrato che devo riempire
- ▶ **Qual è il livello massimo?**
 - ▶ n^2 , l'ultima casella

Analizzare il problema

- ▶ Com'è fatta una soluzione parziale i -esima?
 - ▶ Quadrato riempito fino alla casella i -esima

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | |
| | | |

Analizzare il problema

- ▶ Com'è fatta una soluzione parziale i -esima?

- ▶ Quadrato riempito fino alla casella i -esima

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | |
| | | |

- ▶ Com'è fatta una soluzione completa?

- ▶ Il quadrato completo di tutti numeri $1, 2, \dots, n^2$

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Analizzare il problema

- ▶ **Come viene avviata la ricorsione (livello 0)?**
 - ▶ La ricorsione inizia al livello 0 con il quadrato vuoto

Analizzare il problema

- ▶ Come viene avviata la ricorsione (livello 0)?
 - ▶ La ricorsione inizia al livello 0 con il quadrato vuoto
- ▶ Come genero Parziale(i+1) partendo da Parziale(i)?

```
for (i = 0; i < n*n; i++)  
    if (!parziale.contains(i))  
        parziale.add(i)
```


Identificare le soluzioni valide

- ▶ Data una soluzione **parziale**, come sapere se è valida?
 - ▶ Non esistono soluzioni parziali valide

Identificare le soluzioni valide

- ▶ Data una soluzione **parziale**, come sapere se è valida?
 - ▶ Non esistono soluzioni parziali valide
- ▶ Data una soluzione **completa**, come sapere se è valida?
 - ▶ Calcolo la somma del numero di righe, di colonne e delle diagonali. Confronto con magic number

Identificare le soluzioni valide

- ▶ Data una soluzione **parziale**, come sapere se è valida?
 - ▶ Non esistono soluzioni parziali valide
- ▶ Data una soluzione **completa**, come sapere se è valida?
 - ▶ Calcolo la somma del numero di righe, di colonne e delle diagonali. Confronto con magic number
- ▶ Cosa devo fare con le soluzioni complete valide?
 - ▶ Fermarmi alla prima?
 - ▶ Generarle e memorizzarle tutte?
 - ▶ Contarle?

Progettare le strutture dati

- ▶ Qual è la struttura dati per memorizzare una soluzione (parziale o completa)?
 - ▶ `ArrayList<Integer>`

Progettare le strutture dati

- ▶ Qual è la struttura dati per memorizzare una soluzione (parziale o completa)?
 - ▶ `ArrayList<Integer>`
- ▶ Qual è la struttura dati per memorizzare lo stato della ricerca (della ricorsione)?
 - ▶ Variabile intera `livello`.

Scheletro del codice

```
// Struttura di un algoritmo ricorsivo generico

void recursive (... , level) {

    // E -- sequenza di istruzioni che vengono eseguite sempre
    // Da usare solo in casi rari (es. Ruzzle)
    doAlways();

    // A
    if (condizione di terminazione) {
        doSomething;
        return;
    }

    // Potrebbe essere anche un while ()
    for () {

        // B
        generaNuovaSoluzioneParziale;

        if (filtro) { // C
            recursive (... , level + 1);
        }

        // D
        backtracking;
    }
}
```

Riempire lo scheletro (del codice)

| Blocco | Frammento di codice |
|--------|---------------------|
| A | |
| B | |
| C | |
| D | |
| E | |

```
// Struttura di un algoritmo ricorsivo
void recursive (... , level) {
    // E -- sequenza di istruzioni che ve
    // Da usare solo in casi rari (es. Ru
    doAlways();

    // A
    if (condizione di terminazione) {
        doSomething;
        return;
    }

    // Potrebbe essere anche un while ()
    for () {






        // B
        generaNuovaSoluzioneParziale;

        if (filtro) { // C
            recursive (... , level + 1);
        }

        // D
        backtracking;
    }
}
```

Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)”
- ▶ Sei libero:
 - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
 - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
 - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
 - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
 - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>