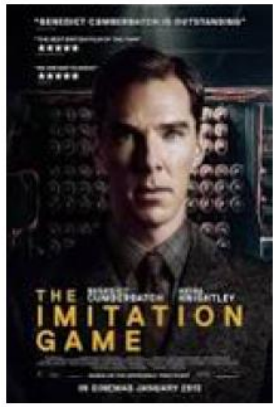
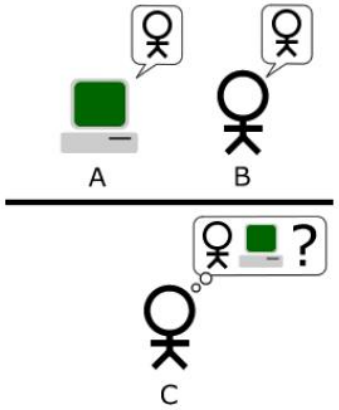
Arduino實作二：猜數字AI

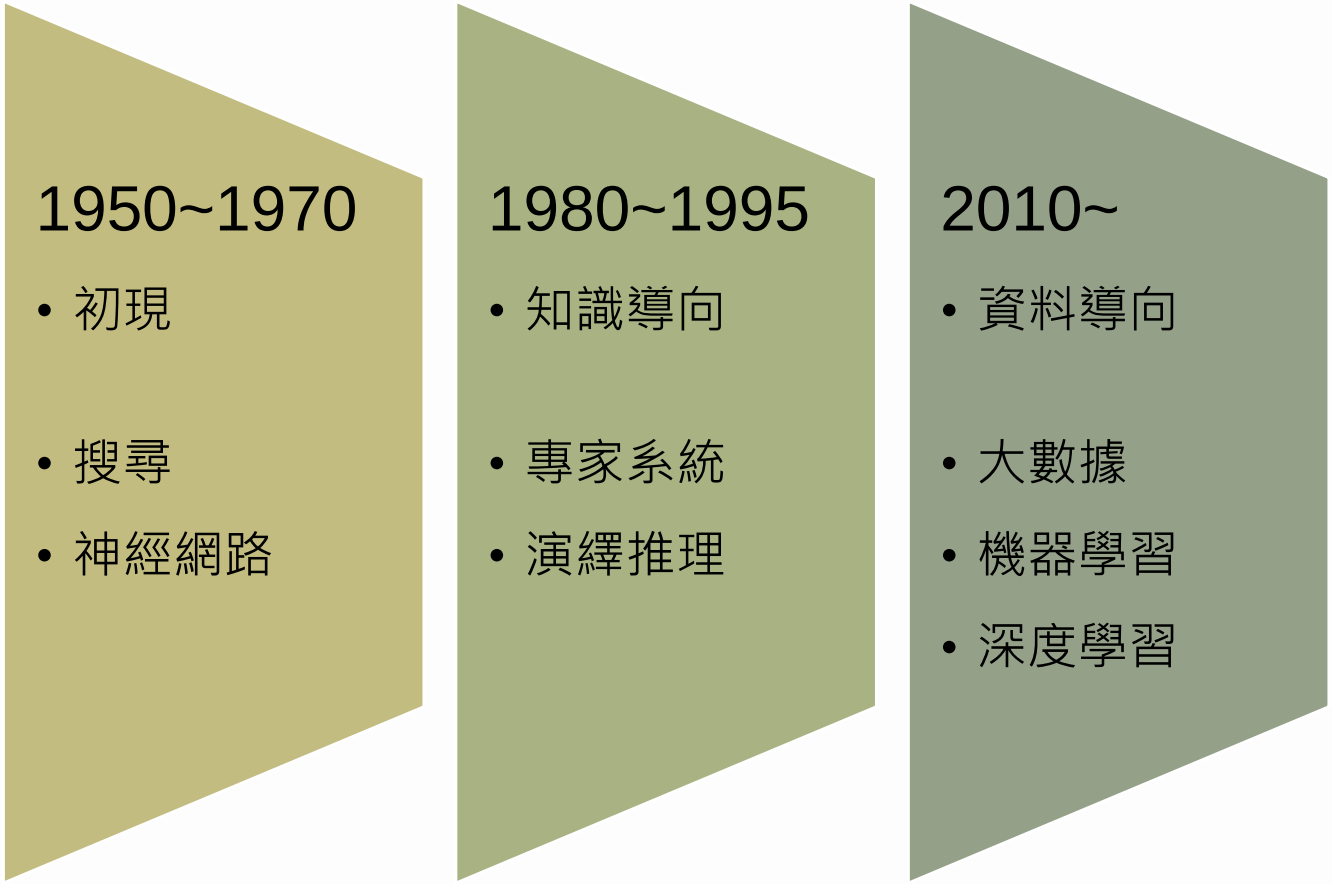
AI是什麼？

- 強AI與弱AI
- 圖靈測試(1950)
- 視覺圖靈測試



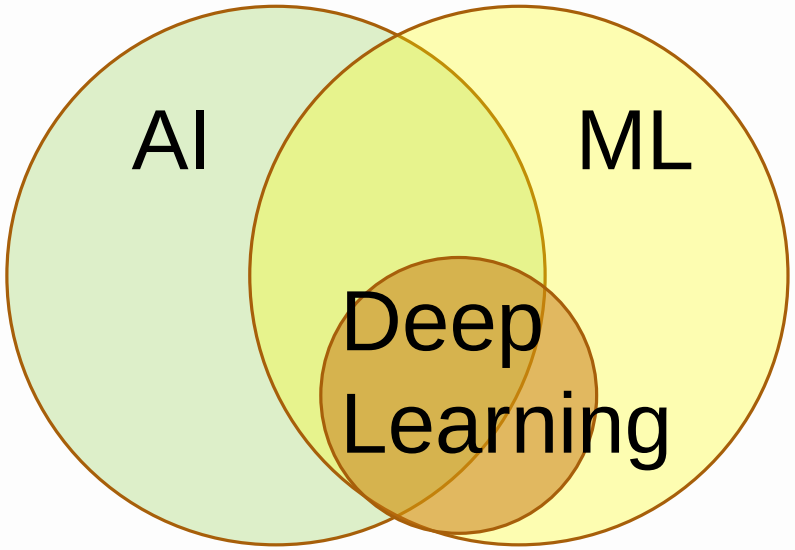


AI的三波熱潮


1950~1970
• 初現
• 搜尋
• 神經網路
1980~1995
• 知識導向
• 專家系統
• 演繹推理
2010~
• 資料導向
• 大數據
• 機器學習
• 深度學習

AI各種面相

- Level 0: 條件反射，市場營銷
- Level 1: 依據知識庫搜尋、計劃
- Level 2: 學習、探索
- Level 3: 自動特徵產生、高度抽象化



猜數字

1. 出題者想一組四個不一樣的數字給猜題者猜。

2. 猜題者提出一組數字，出題者回答幾A幾B。A代表猜題者答出數字正確、位置也正確的數量，B代表猜題者答出數字正確、但位置不正確的數量。

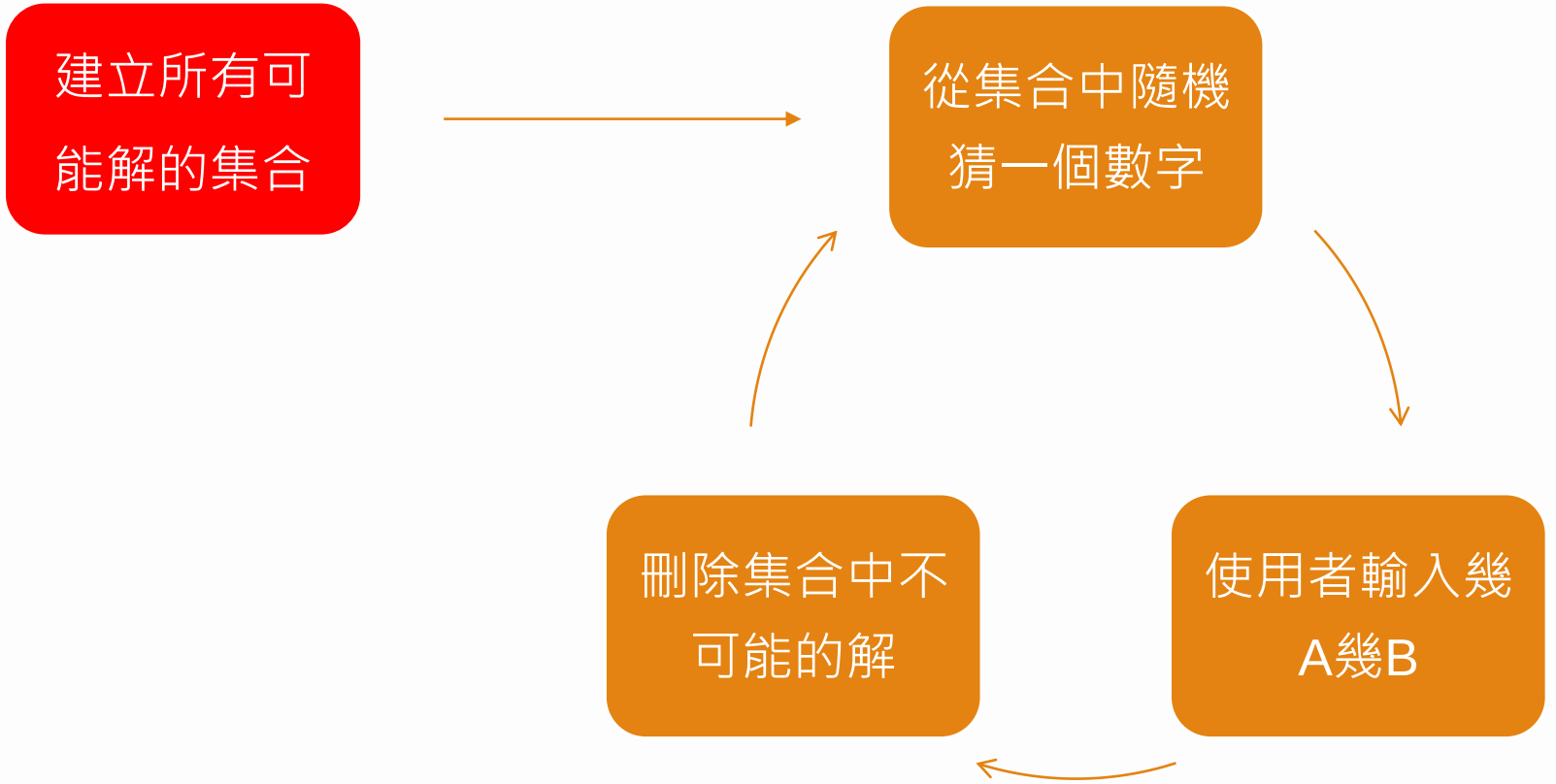
猜數字最佳解

- 目前已知最佳的演算法能在七次以內猜到正確答案。
- 已知不可能存在保證六次內解完的演算法。
- 最少平均猜題數為5.213次，有數種解法可以達成，例如：http://slovesnov.users.sourceforge.net/index.php?bullscows_tree,english,avgBullsCows

窮舉法

- 隨機挑選任一組數字，依據已知資訊刪除不可能的解，再從剩餘可能解隨機猜測。
- 平均表現不差，大約落在5~6次之間。
- 最差的情況可能猜到9次。
- 程式寫起來相對簡單。

程式流程



建立所有可能解的集合

- 從0123到9876，總共有10x9x8x7=5040個解
  - P^{10}_4=10*9*8*7=5040

- Arduino的變數上限為2048 bytes，一個int容量為2 bytes，若寫成int array會放不下

  →參考下頁範例

範例一(不須操作)

- 執行practice2-1，雖然可以執行編譯
- 但實際上已當機...。


```
Serial.begin(9600);
int a[100];
for(int j=0;j<100;j++){
a[j]=j;
}
Serial.print("a[41]=");
Serial.println(a[41]);
Serial.print("a[50]=");
Serial.println(a[50]);

int b[10000];
for(int j=0;j<10000;j++){
b[j]=j;
}
Serial.print("b[4124]=");
Serial.println(b[4124]);
Serial.print("b[5350]=");
Serial.println(b[5350]);
```

因此，我們這樣做

- 一個byte有八個bits，可以儲存0or1，猜數字時，我們將
  - 不可能的解→ 0 可能的解→1
- 從guess[0]看起
  - 最右邊代表數字0
  - 右邊第二個代表1
  - 右邊第三個代表2
  - 以此類推
- guess[1]:
  - 最右邊代表數字8...

char guess[1250]		實際代表
guess[0]	00000000	7 6 5 4 3 2 1 0
guess[1]	00000000	15 14 13 12 11 10 9 8
guess[2]	00000000	23 22 21 20 19 18 17 16
…	…	
guess[15]	11111000	127~120
…	…	
guess[1249]	00000000	9999~9992

建立所有可能解的集合

- 將容量精簡化，改成以一個bit記錄該解是否可用，如此只需要10000 bits = 1250 bytes就能記錄所有的解。
- 以往，char guess[10000];
- 現在char guess[1250];

char guess[1250]		實際代表
guess[0]	00000000	7 6 5 4 3 2 1 0
guess[1]	00000000	15 14 13 12 11 10 9 8
guess[2]	00000000	23 22 21 20 19 18 17 16
…	…	
guess[15]	11111000	127~120
…	…	
guess[1249]	00000000	9999~9992

0000	0001	0002	……	0122	0123	0124	……	9997	9998	9999
0	0	0		0	1	1		0	0	0

範例二、觀察

1. 請先將前兩個函式複製過去

   定義char answerSet[1250] = {0};

2. for (int i = 0; i < 1000; ++i)

   if (isValid(i)){

   bitWrite(answerSet[i/8], i%8, 1);

   }

3. 觀察:

```
Serial.println(answerSet[14],BIN);
Serial.println(answerSet[15],BIN);
Serial.println(answerSet[16],BIN);
```

char guess[1250]		實際代表
guess[0]	00000000	7 6 5 4 3 2 1 0
guess[1]	00000000	15 14 13 12 11 10 9 8
guess[2]	00000000	23 22 21 20 19 18 17 16
…	…	
guess[15]	11111000	127~120
…	…	
guess[1249]	00000000	9999~9992

練習一、實作函式

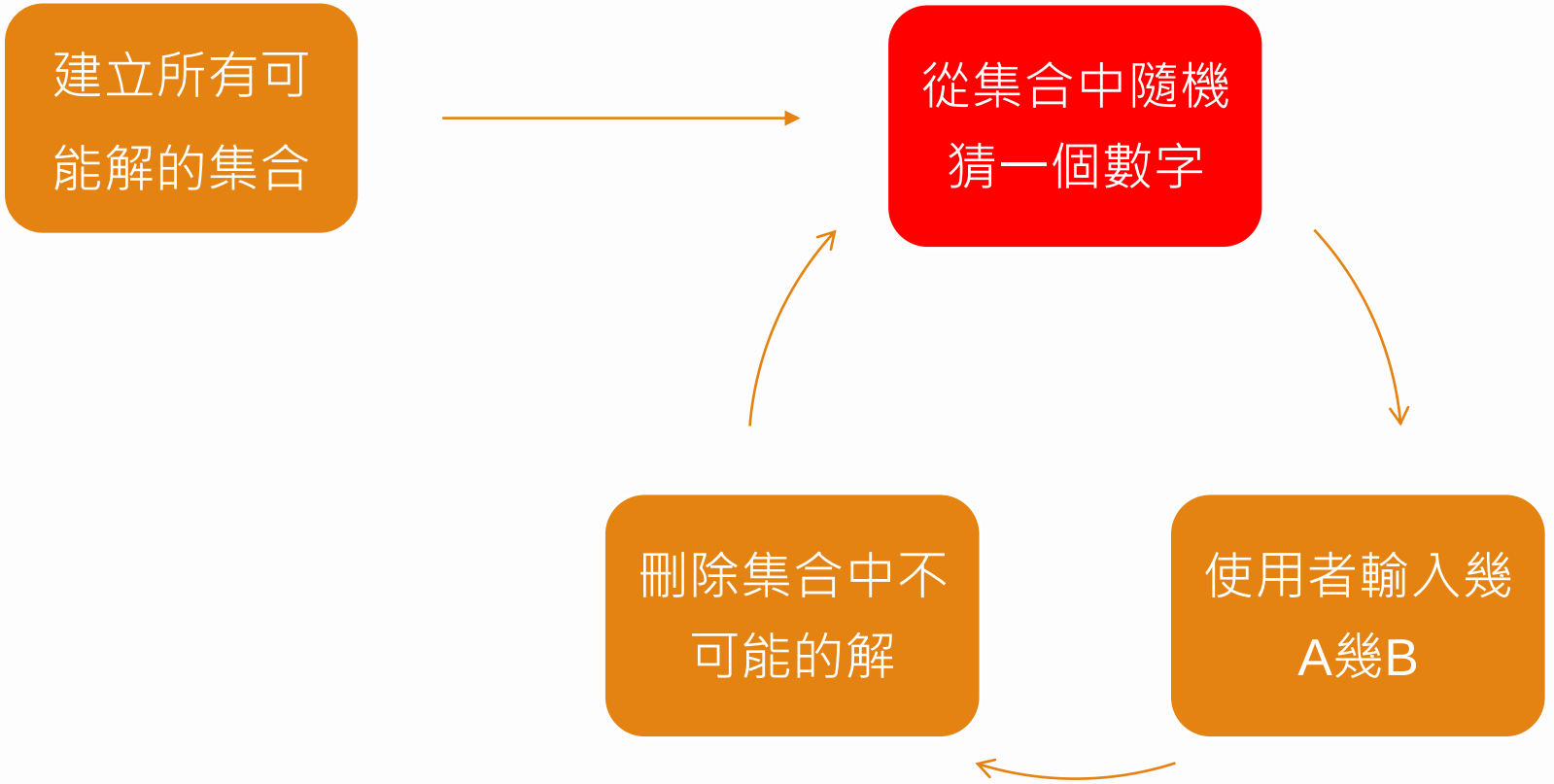
- 為了方便判斷四位數字的每個值，我們先寫一個函式，方便之後利用~

- int getDigit(int number, int position)，輸出number中某個位置的值。position由右至左，從0開始。例：getDigit(1234, 1)==3

練習二、建立所有可行集合

1. 請將練習一: int getDigit(int number, int position)完成的答案複製到練習二
2. 完成bool isValid(int number)，使得可行集合回傳為True，其他為False
3. 舉例: isValid(0000)→False : isValid(0123)→True
4. 執行程式看看輸出結果

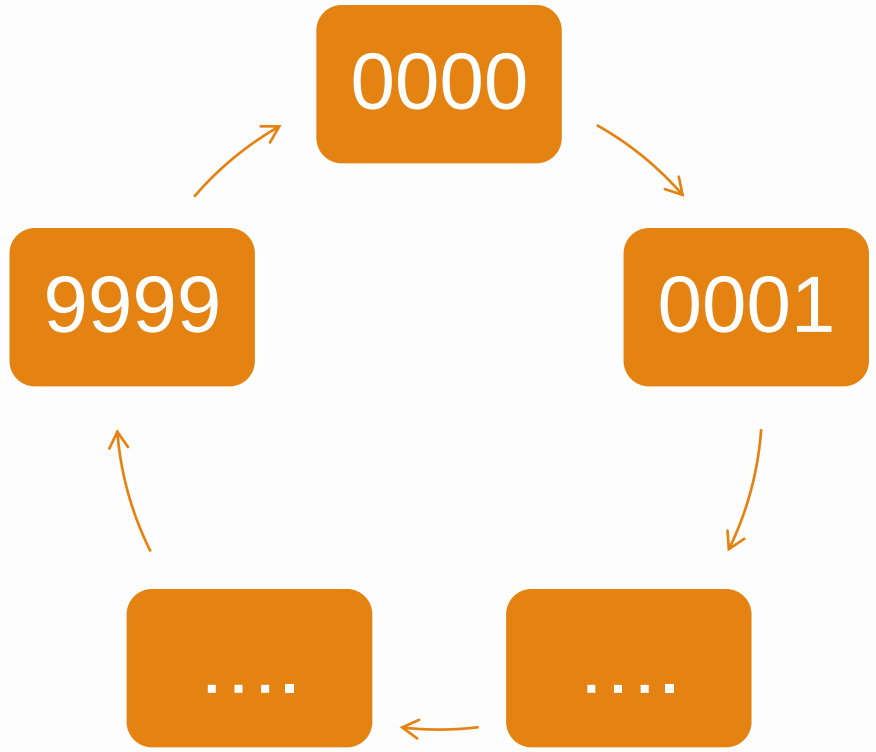
程式流程



隨機猜一個數字

- 在總集合中取亂數，若該位元標記為可使用就猜，不行就從該數字往後數至可用為止。

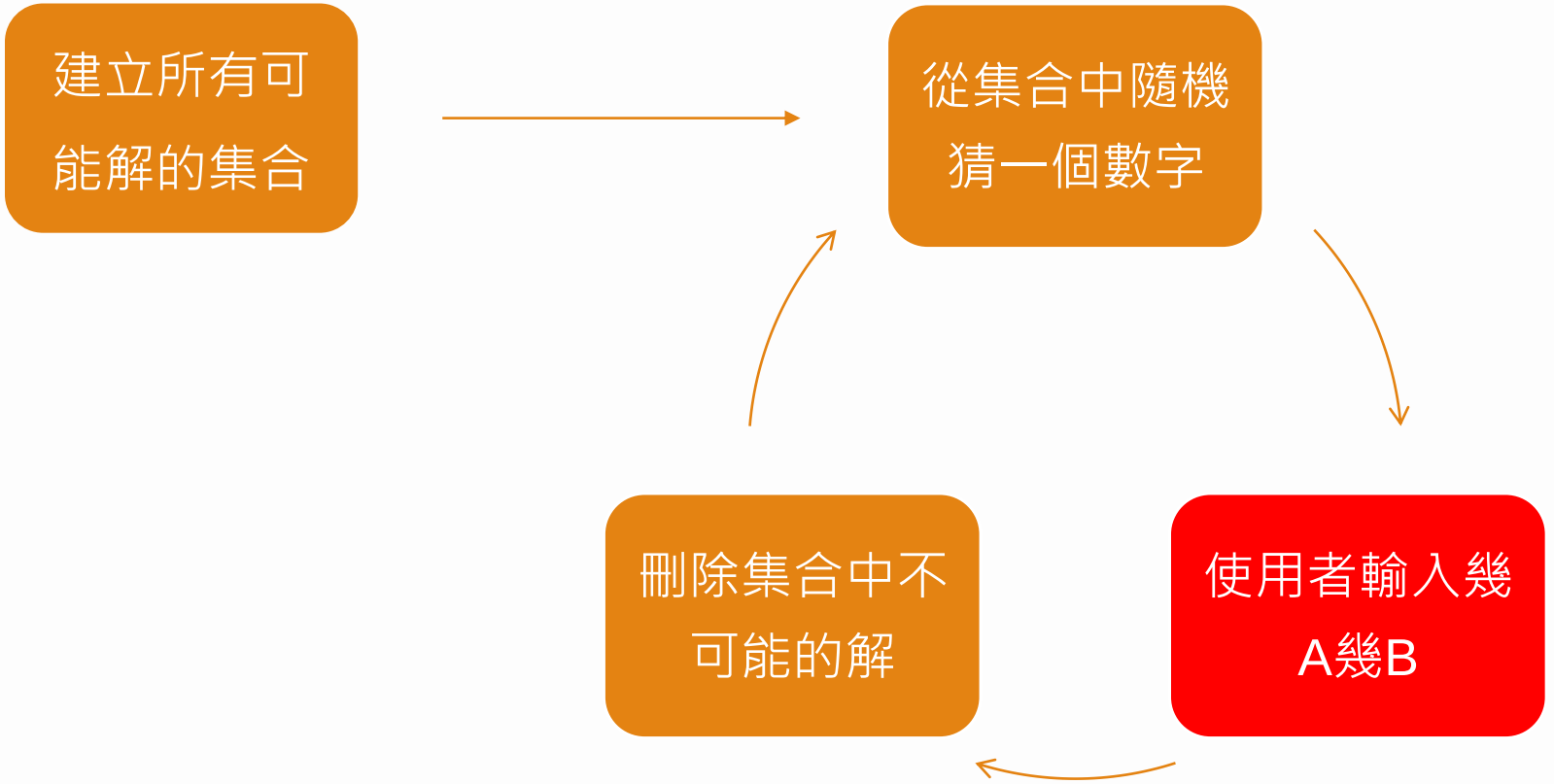
- 超過10000取餘數



隨機猜一個數字


```
void guessNext() {
  int index = random() % totalGuess;
  int counter = 0;
  for (int i=0; i<10000; ++i) {
      if (bitRead(answerSet[i/8], i%8) != 0)
          counter++;
      if (counter > index) {
          guess = i;
          break;
      }
  }
}
```

程式流程



使用者輸入回答

- 本實作沿用前面幾堂課的按鍵板、七段顯示器、蜂鳴器架構。
- 在左半邊的七段顯示器顯示AI猜的數字，右半邊顯示幾A幾B。
- 使用者用按鍵板回答幾A幾B、按*重新開始，用蜂鳴器發出按鍵音。

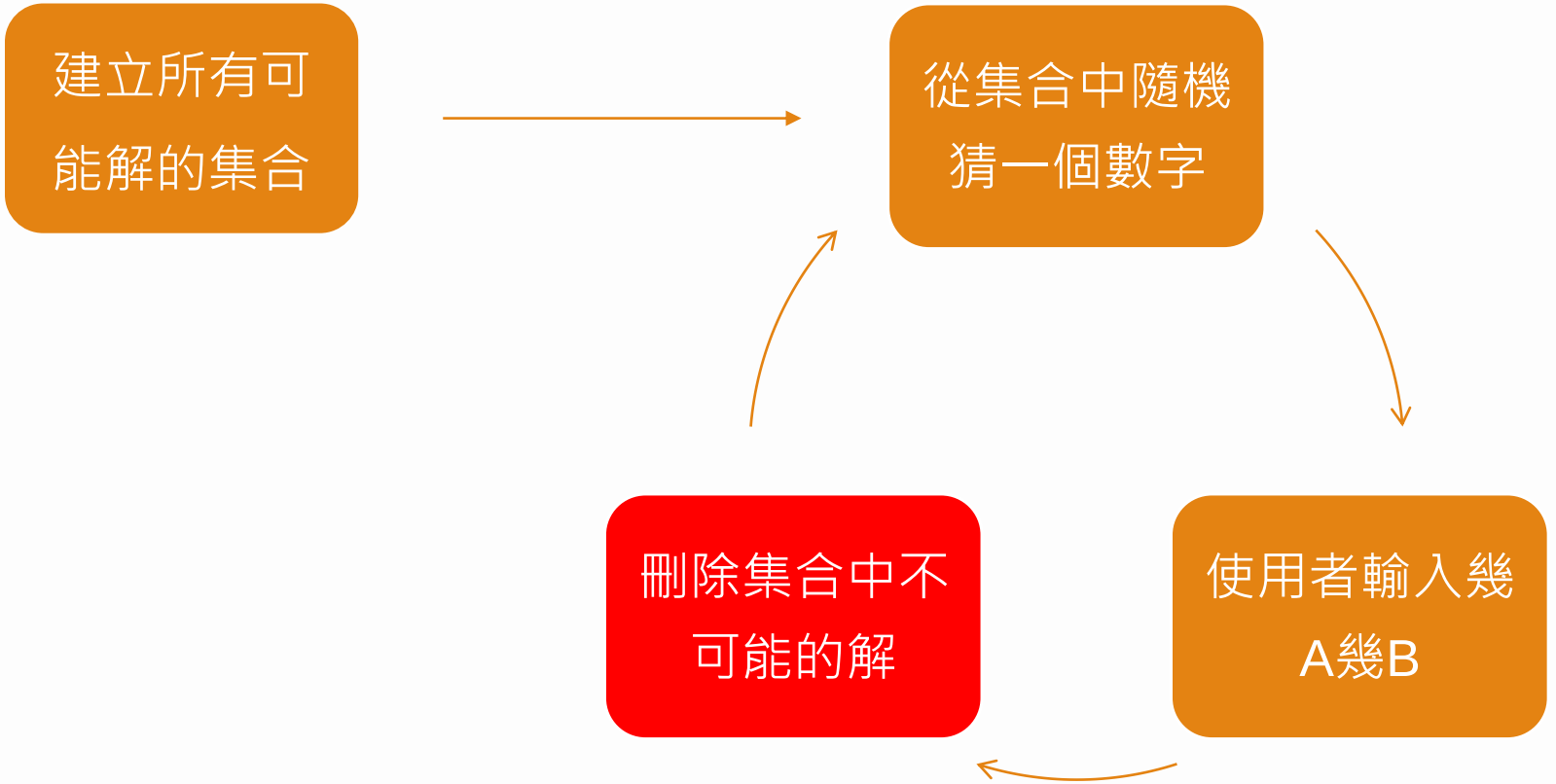
Think…

如何輸入兩個數字分別代表幾A幾B???

在loop迴圈中…

```
 char key = kpd.getKey();
  if (key >= '0' && key <= '4') {
    if (secondArguement) {
    b=key;
    secondArguement = false;
}
else {
      a = key;
secondArguement = true;
    }
  }
```

程式流程



刪去不可能的解/猜下一組

- 一一檢查各個解是否符合使用者輸入的幾A幾B，若不符合則標記為不可用。
- 猜下一組解同前面猜第一組解的做法。
- 若總集合都不可用，代表使用者回答有錯誤，顯示錯誤訊息。

Void deleteAns(int guess, int ab)

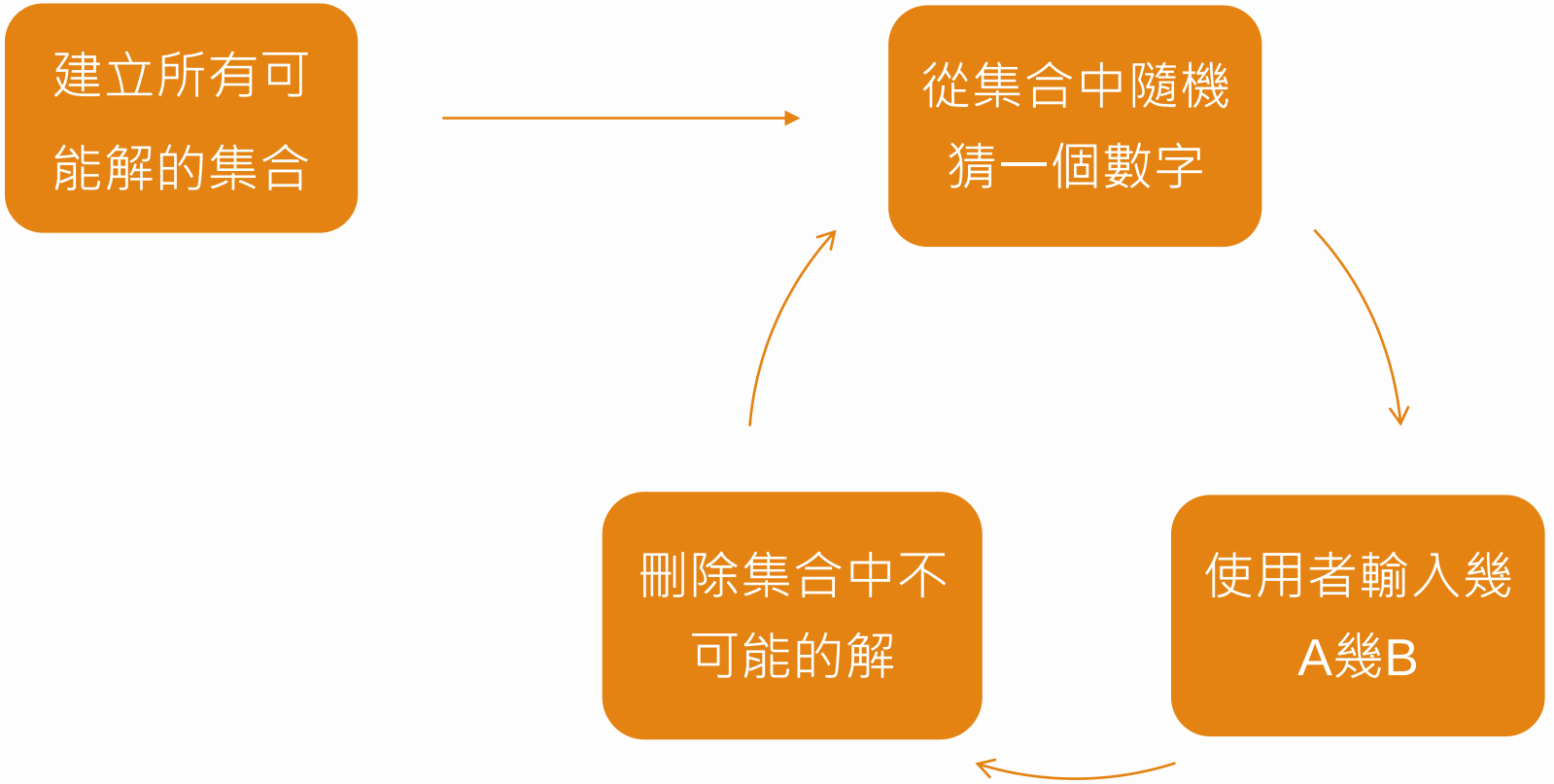
```
void deleteAns(int guess, int ab) {


  for (int i = 0; i < 10000; ++i) {
    if (bitRead(answerSet[i/8], i%8) == 0) continue;


    if (getAB(guess, i) != ab) {
       bitWrite(answerSet[i/8], i%8, 0);
      --totalGuess;
    }
  }
```

```
if (totalGuess <= 0)  {
    Serial.println("error");
    lc.setChar(0,7,'E',false);
    lc.setChar(0,6,'A',false);
    lc.setChar(0,5,'A',false);
    lc.setChar(0,4,'0',false);
    tone(13,330,100);
    delay(200);
    tone(13,330,600);
    delay(1000);
    restart();
    Display();
  }
}
```

程式流程→組合



組合一、Void setup()

```
void setup()
{
  Serial.begin(9600);//每秒取樣9600次
  pinMode(13,OUTPUT);
  randomSeed(analogRead(A0));
  restart();
  lc.shutdown(0,false);
  lc.setIntensity(0,5);
  lc.clearDisplay(0);
  Display();
}
```

組合二、Void loop()

```
char key = kpd.getKey();
 if (key >= '0' && key <= '4') {
   if (secondArguement) {
     b = key;
     tone(13,523,100);
     Display();
     deleteAns(guess,  ((a-'0')*10+(b -'0')));
     guessNext();
     secondArguement = false;
     a = ' ';
     b = ' ';
     Display();
   }
   else {
      a = key;
      tone(13,523,100);
      Display();
      secondArguement = true;
    }
  }
  else if (key == 'A') {
    restart();
    Display();
  }
}
```

組合三、Void Display()

```
void Display() {
  lc.setChar(0,7,guess/1000,false);
  lc.setChar(0,6,guess/100%10,false);
  lc.setChar(0,5,guess/10%10,false);
  lc.setChar(0,4,guess%10,false);
  lc.setChar(0,3,a,false);
  lc.setChar(0,2,'A',false);
  lc.setChar(0,1,b,false);
  lc.setChar(0,0,'b',false);
}
```

組合四、Void restart()

```
void restart() {

  totalGuess = 5040;

  for (int i = 0; i < 10000; ++i)
    if (isValid(i))
      bitWrite(answerSet[i/8], i%8, 1);

  guessNext();
}
```

實作

- 請實作函式int getDigit(int number, int position)，輸出number中某個位置的值。position由右至左，從0開始。例：getDigit(1234, 1)==3
- 請實作函式 bool isValid(int number)，確認number為4位數 (0<=number<=9999)，以及任兩位數字不同。
- 請將所有可能解印出來，確認一下有5040組，以及看起來無明顯錯誤。
- 請實作函式 int getAB(int guess, int answer) 。回傳值為2位數，前A後B。例：getAB(1234, 1349) == 12
- 最後將整個程式串起來完成！

Bonus: 5040組解

If 4A0B →1

If 3A0B →C4取3*P6取1=24

If 2A2B →C4取2=72

If 2A1B →C4取2*C2取1*P6取1=72

…

加總=5040?

Bonus: 七次內猜對的演算法

- 如同前述，有多種演算法可以在七次內猜到正確答案。差別在於猜下一個數字時如何挑選，例如FanoutChooser、GiniIndexChooser、MinSetChooser等等都可以在七次內猜完。參考：

https://www.javaworld.com.tw/jute/post/view?bid=35&id=140228&tpg=1&ppg=1&sty=0&age=0