MIDTERM EXERCISE

TA TIME

THE NUMBER OF OCCURRENCES

- **Statement**
  - **Find the maximum number of occurrences**
- **Sample Input**
  - **12**
  - **4**
  - **12112212**
  - **12121212**
  - **111222**
  - **12112**
- **Sample Output**
  - **4**

THE NUMBER OF OCCURRENCES

- **Hint**
  - 可以用 **strlen(A) strlen(B)** 取得**A**與**B**的長度
  - 跑過每個 **B** 的 **index,** 並從每個**index**出發**,** 看是否成功找到 **A**
  - 計數總共找到幾組**A**
  - 找出最大值

PROGRESSION

- **Statement**
  - **Judge whether input is Arithmetic Progression or a Geometric Progression**
  - **Print out the first number and the 5th number**

- **Sample Input**
  - **2 -4 8**

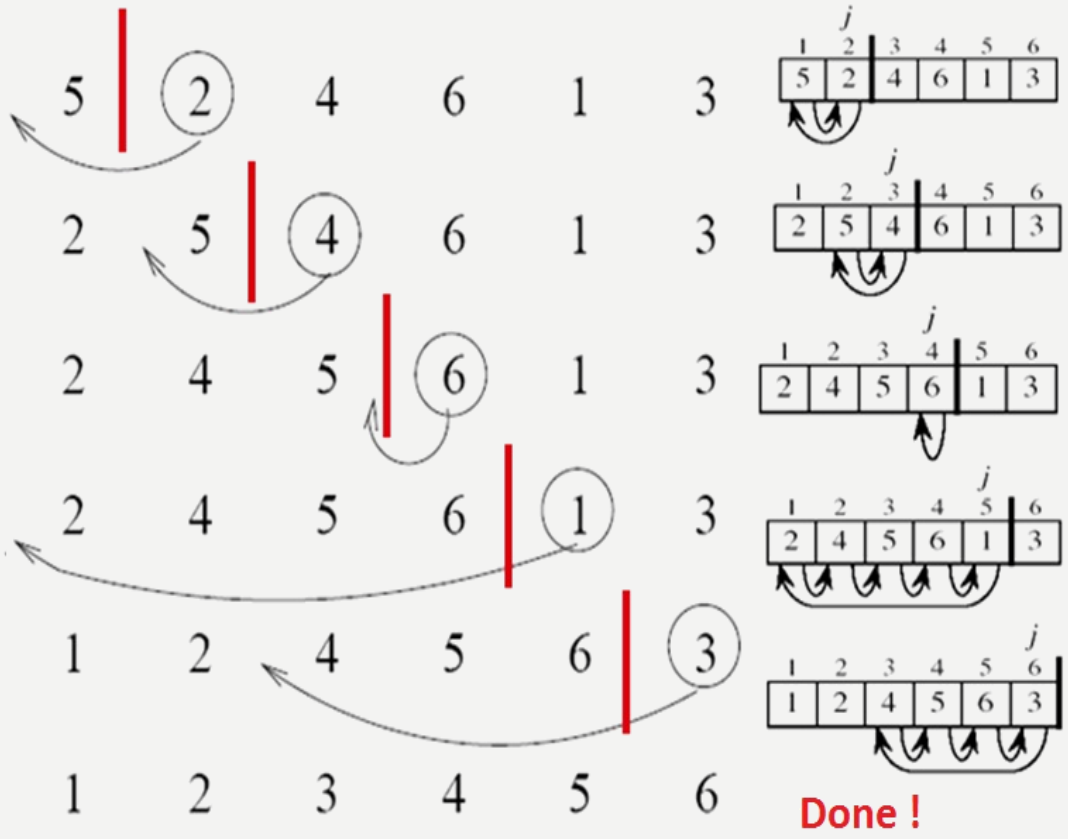
- **Sample Output**
  - **-1 -16**

SIMPLE INTEGER SORTING

- **Statement**
  - **List items in increasing order**
- **Sample Input**
  - **5**
  - **4 5 3 1 2**
- **Sample Output**
  - **1 2 3 4 5**

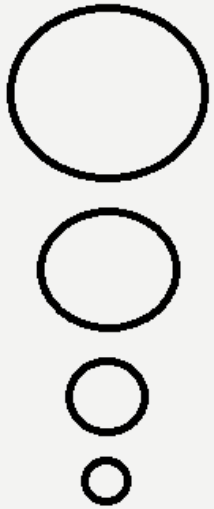
SIMPLE INTEGER SORTING

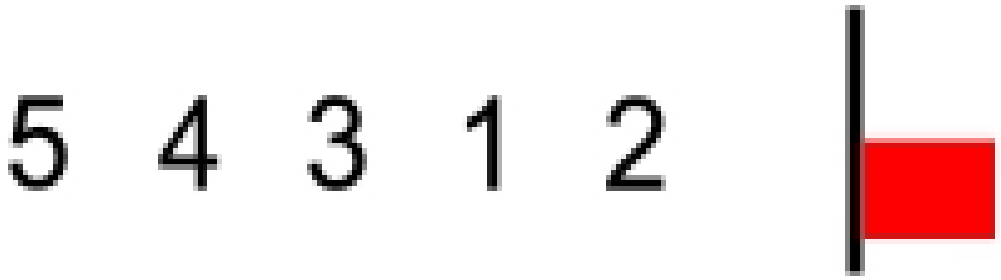
- **Insertion Sort**
  - goo.gl/nNwKCH



SIMPLE INTEGER SORTING

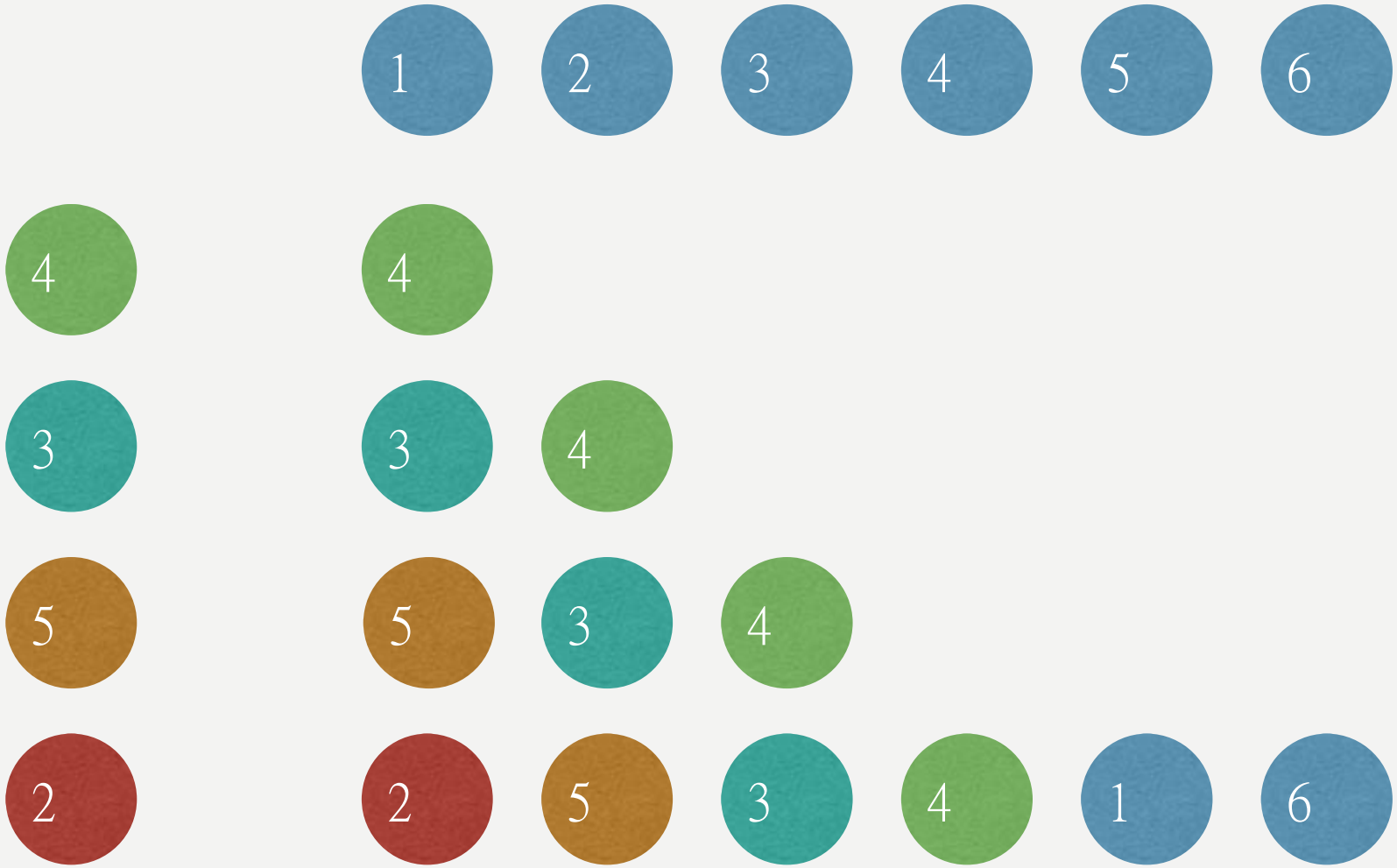
- **Bubble Sort**
  - **Flag**
  - goo.gl/WJZ7iL





PA – ARRANGING A SEQUENCE

- **Description**
  - **Maintain a sequence that when you input a number , the number will go to the first position of the sequence and you need to print out the sequence when the input end**
- **Concept**
  - **When taking the assigned number to the first place, others after it will forward one place.**
  - **The last input will be the first output number**


1 2 3 4 5 6
4 4
3 3 4
5 5 3 4
2 2 5 3 4 1 6

PA – ARRANGING A SEQUENCE

- **Implementation**
  - **Maintain an array, store the number which we want to pick it to the first**
  - **From the last to the first , maintain an array if the number doesn't appear print it**
  - **From the first to the last , if the number didn't change the position , print it out**

```c
for(i= 0 ; i < change ; i++){
    scanf("%d" , &a[i]) ;
}
```

```c
for(i = change -1 ; i >=0 ; i--){
    if(!is[a[i]]){
        printf("%d\n" , a[i]);
        is[a[i]]  = 1 ;
    }
}
```

```c
for(i = 1 ; i<=num ; i++){
    if(!is[i])printf("%d\n" , i);
}
```

PB – BIRTHDAY PARTY

- long long int
- LCM(GCD between x, y, z and ponies)

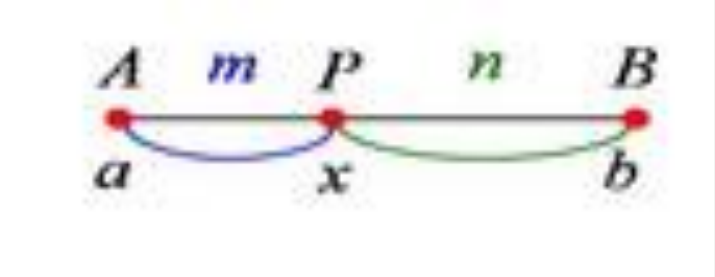
```
LCM_4_3(GCD(red, num), GCD(blue, num), GCD(yellow, num))
```

```
long long int LCM(long long int a, long long int b){
    return a / GCD(a, b) * b;
}

long long int LCM_4_3(long long int a, long long int b, long long int c){
    return LCM(LCM(a, b), c);
}
```

PC - COLLINEAR

- **3 finite loops to compare any 2 points with the rest points**




```
for(i = 0 ; i < n ; i++){
    for(j = i+1 ; j < n ; j++){
        // find dots on the same vector between dots[i] and dots[j]
        for(k = 0 ; k < n ; k++){
            if(k == i || k == j)    continue;
            else{
                // (a.x - b.x) * (a.y - b.y) == (c.x - b.x) * (c.y - b.y)
            }
        }
    }
```

PD – DISTRAIT

- **Description**
  - **Judge that if you can bingo or not , if you can , print which step we will bingo , if not , print Not yet \(^o^)/**
- **Concept**
  - **Every time you can search whether you can bingo or not**
  - **Remember the row and column of the input number and only test this row and column**

23	20	14	12	17
1	3	5	9	19
2	16	18	25	21
4	6	7	8	22
10	11	13	15	24

PD – DISTRAIT

- **Implementation**
  - **Recall the index of the input data**
  - **Judge that if the number we give will make the bingo condition satisfied**

```c
int x; scanf("%d", &x);
pos[x][0] = i;
pos[x][1] = j;
int rs[5]={0}, cs[5]={0}, ul=0, ur=0;
rs[pos[x][0]] ++;
cs[pos[x][1]] ++;
if(pos[x][0] == pos[x][1]) ul ++;
if((pos[x][0]+pos[x][1]) == 4) ur ++;
```

PE – EXQUISITE SUBSTRINGS

- String S - "PlayerUnknowns BattleGrounds"
- A Substring of a string S is a string S' that occurs in S.
- For example, "BattleGrounds" is a substring (S').

- The list of all substrings of the string "Dude" would be
- "Dude", "Dud", "ude", "Du", "ud", "de", "D", "u", "d", "e".

PE – EXQUISITE SUBSTRINGS

- **A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as *AAAA* or *DudeduD*.**

- **In problem 11621 :**

  HT discovered that for some special string *s*, he could find two distinct indices *l* and *r* such that if he reverses the substring *s[l, r]*

PE – EXQUISITE SUBSTRINGS

- Input :
  - abcd
  - abab
  - aaaaa
- Output:
  - 0
  - 2
  - 10

PE – EXQUISITE SUBSTRINGS

- **Solution**

```
for(i = 0; i < length of String ; i++){
        for(j = i+1; j<length of String ; j++){
            if String(i, j) is Palindrome :
                Number of Exquisite Substrings ++;
        }
    }
int isPalindrome(int l, int r)
{
    Decide whether s[l,r] is palindrome
}
```

PE – EXQUISITE SUBSTRINGS

- **Solution**

```
// middle is char, “o0乂A乂0o”, “QAQ”
for( the index in string ){
    l =  left , r = right;
    while( l and r do not exceed the boundry ){
        if( left char == right char ) {
            Number ++,  then keep checking
        }
        else break;
    }
}
// middle is empty “7777”
```