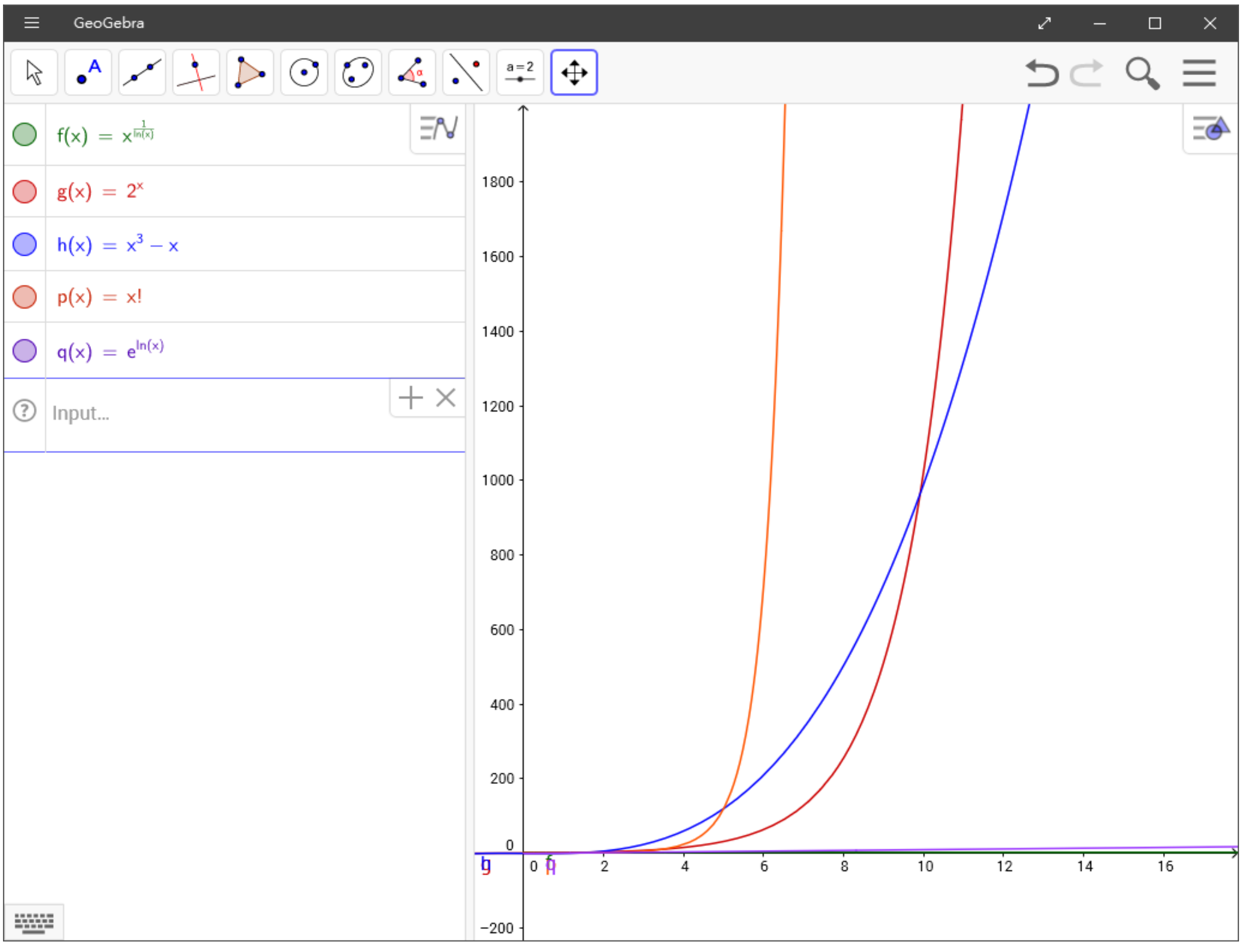
DSA hw1 b05902086

1.1.



$g_1 = n!$

$g_2 = 2^n$

$g_3 = n^3 - n$

$g_4 = \mathrm{e}^{\ln n}$

$g_5 = n^{\frac{1}{\ln n}}$

1.2.

$$n = 4 \Rightarrow 4! > 2^4,\ \forall n > 4[n > 2] \Rightarrow \forall n \geq 4[n! > 2^n],\ \lim_{\mathrm{h}\to\infty} \frac{2^n}{n!} = 0$$

$$\Rightarrow \exists C > 0, N \in \mathbb{N}[\forall n > N \Rightarrow 0 \leq C2^n < n!] \Rightarrow n! = \omega(2^n)$$

$$n = 2 \Rightarrow 2! < 2^2, \forall n > 2\left[n \leq \frac{n^{n+1}}{n^n} < \frac{(n+1)^{n+1}}{n^n}\right] \Rightarrow \forall n \geq 2[n! < n^n],\ \lim_{n\to\infty} \frac{n!}{n^n} = 0$$

$$\Rightarrow \exists C > 0, N \in \mathbb{N}[\forall n > N \Rightarrow 0 \leq n! < Cn^n] \Rightarrow n! = o(n^n)$$

1.3.

(a)

$f(n)=O(g(n)) \Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le Cf(n)\le g(n)]$

$\Rightarrow g(n)=\Omega(f(n)) \Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le Cf(n)\le g(n)]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)] \Rightarrow f(n)=O(g(n))$

(b)

$f(n)=\Theta(g(n)) \Rightarrow \exists C_1>0, C_2>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le C_1g(n)\le f(n)\le C_2g(n)]$

$\Rightarrow \exists C_1>0, C_2>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le C_2g(n), 0\le C_1g(n)\le f(n)]$

$\Rightarrow f(n)=O(g(n)), f(n)=\Omega(g(n))$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)],$

$\exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le Cg(n)\le f(n)]$

$\Rightarrow \exists C_1>0, C_2>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le C_1g(n)\le f(n)\le C_2g(n)]$

$\Rightarrow f(n)=\Theta(g(n))$

(c)

$f(n)=O(g(n)) \Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\cdot g(n)\le Cg(n)\cdot g(n)]$

$\Rightarrow f(n)\cdot g(n)=O(g(n)^2)$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\cdot g(n)\le Cg(n)^2] \Rightarrow f(n)=O(g(n))$

(d)

$f(n)=O(g(n)) \Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)^2\le C^2g(n)^2]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)^2\le Cg(n)^2]$

$\Rightarrow f(n)^2=O(g(n)^2)$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)^2\le Cg(n)^2]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)^2\le C^2g(n)^2]$

$\Rightarrow \exists C>0, N\in\mathbb{N}[\forall n>N \Rightarrow 0\le f(n)\le Cg(n)]$

$\Rightarrow f(n)=O(g(n))$

2.1.

time complexities of Binary_Search: $O(N\log_2 N)$

time complexities of Count_Search: $O(N)$

space complexities of Count_Search: $O(K)$

當 K 在記憶體可開起來的範圍時，使用 Count_Search，否則使用 Binary_Search

2.2.

(a)

```
Calculate_M(A[],N, K, k){
    M=0;
    for(i=0;i<N;i++)
        for(j=i+1;j<N;j++)
            M+=(A[i]+A[j]==k);
    return M;
}
```

(b)

```
Binary_Search_2(A[],N,begin,val,op){
    left=begin;
    right=N-1;
    while(left<=right){
        size_t mid=(left+right)/2;
        if(op(A[mid],val))left=mid+1;
        elsr left=mid-1;
    }
    return right;
}

Calculate_M_2(A[],N,K,k){
    M=0;
    sort(A);
    for(i=0;i<N;i++)
        M+=Binary_Search_2(A,N,i+1,k-A[i],less_equal())-
Binary_Search_2(A,N,i+1,k-A[i],less());
    return M;
}
```

2.3.

```
Calculate(A[],N,K,m){
    for(i=0;i<N;i++){
        k=N-1;
        for(j=i+1;j<k;j++){
            while(k>j+1&&A[i]+A[j]+A[k]>m)
                k--;
            if(A[i]+A[j]+A[k]==m)
                return true;
        }
    }
    return false;
}
```

因為 k 在每次第二層 for 中最多只能減N次($0 \le k < N$)，因此 while 是均攤$O(1)$的，而再加上外面的兩層 for 迴圈，總複雜度為$O(N^2)$

3.1.

PUSH 1

PUSH 2

PUSH 3

POP

POP

PUSH 4

POP

POP

PUSH 5

POP

3.2.

```
Valid2(A[],N){
    for(i=0;i<N;i++)
        if(A[i]!=i+1)
            return false;
    return true;
}
```

最多用 if 判斷N次，因此複雜度為$O(N)$

3.3.

```
Valid3(A[],N){
    S=malloc(N);
    size=0;
    now=1;
    for(i=0;i<N;i++){
        if(size&&S[size-1]==A[i]){
            size--;
            continue;
        }
        while(now<=N&&(!size||S[size-1]!=A[i]))
            S[size++]=now++;
        if(size&&S[size-1]==A[i])
            size--;
        else
            return false;
    }
    return true;
}
```

for 迴圈中的 while 內的語句最多執行N次，而 for 迴圈內的其他東西也最多執行N次，因此總複雜度為$O(N)$

3.4.

```
Valid4(A[],N){
    for(i=0;i<N;i++)
        if(A[i]!=i+1)
            return false;
    return true;
}
```

最多用 if 判斷N次，因此複雜度為$O(N)$

3.5.

```
Valid5(A[],N){
    now=N;
    sp=N;
    for(i=N;i>0;i--){
        if(sp!=N&&A[sp]==i){
            sp++;
            continue;
        }
        while(now&&(sp==N||A[sp]!=i))
            A[--sp]=A[--now];
        if(sp!=N&&A[sp]==i)
            sp++;
        else
            return false;
    }
    return true;
}
```

此作法相當於 3.3 的做法只是把輸入和目標兩邊都翻轉過來後交換，因此可以發現做法是等價的。而 for 迴圈中的 while 內的語句最多執行N次，而 for 迴圈內的其他東西也最多執行N次，因此總複雜度為$O(N)$