DSA hw1 ****b05902086****

1.1.

G OG b 
5 ( : 〕 ! Q 三 
○ f(x) - 
○ h(x) = x3 一 x 
In(x) 
Input.. 
1800 
1500 
1200 
1000 
B00 

1.2.

1.3.

(a)

(b)

(c)

(d)

2.1.

time complexities of Binary\_Search:

time complexities of Count\_Search:

space complexities of Count\_Search:

當 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中最多只能減次()，因此while是均攤的，而再加上外面的兩層for迴圈，總複雜度為

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判斷次，因此複雜度為

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內的語句最多執行次，而for迴圈內的其他東西也最多執行次，因此總複雜度為

3.4.

Valid4(A[],N){

    for(i=0;i<N;i++)

        if(A[i]!=i+1)

            return false;

    return true;

}

最多用if判斷次，因此複雜度為

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內的語句最多執行次，而for迴圈內的其他東西也最多執行次，因此總複雜度為