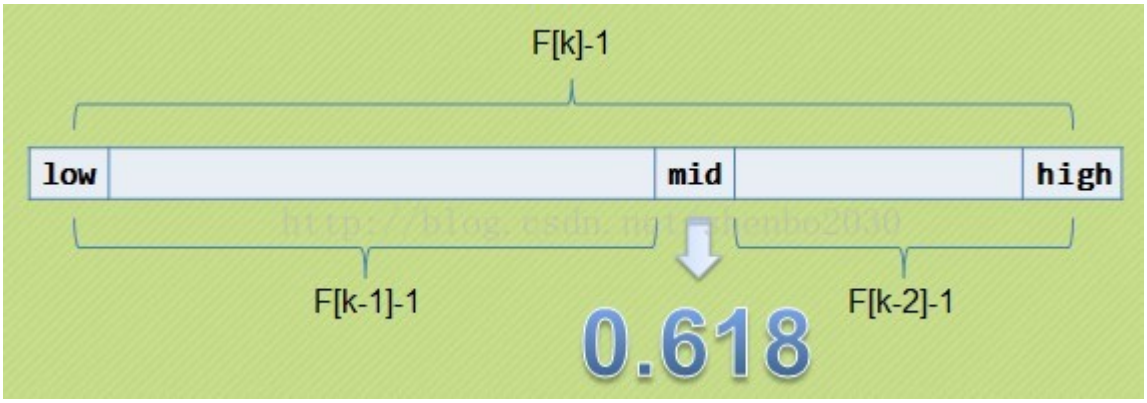
斐波那契查找

【思想】

斐波那契数列从第三个数开始，后边每个数都是前两个数的和。随着斐波那契数列的递增，前后两个数的比值会越来越接近0.618，利用这个特性，可以将黄金比例运用到查找技术中。



相对于折半查找，它是根据斐波那契序列的特点对有序表进行分割。它要求开始表中记录的个数为某个斐波那契数小1， 即 n = F(k) - 1

开始将k值与第F(k-1)位置的记录进行比较(及mid=low+F(k-1)-1),比较结果也分为三种：

1、相等，mid位置的元素即为所求。

2、>， low = mid +1， k-=2

说明：low=mid+1说明待查找的元素在[mid+1,high]范围内，k-=2 说明范围[mid+1,high]内的元素个数为n-(F(k-1))= Fk-1-F(k-1)=Fk-F(k-1)-1=F(k-2)-1个，所以可以递归的应用斐波那契查找。

3、<，high = mid -1, k-=1.

说明：low=mid+1说明待查找的元素在[low,mid-1]范围内，k-=1 说明范围[low,mid-1]内的元素个数为F(k-1)-1个，所以可以递归 的应用斐波那契查找。

【复杂度】

最坏情况下，时间复杂度为O(log2n)，且其期望复杂度也为O(log2n）

【代码】

```
#include <memory>
#include <iostream>

using namespace std;

const int max_size=20; //斐波那契数组长度

/*构造一个斐波那契数组*/
void Fibonacci(int * F){
    F[0] = 0;
    F[1] = 1;
    for(int i = 2; i < max_size; ++i){
        F[i] = F[i-1] + F[i-2];
    }
}

/*定义斐波那契查找法*/
int FibonacciSearch(int *a, int n, int key){
    int low = 0;
    int high = n - 1;

    int F[max_size];
    Fibonacci(F); //构造一个斐波那契数组F

    int * temp; //将数组a扩展到F[k] - 1的长度
    temp = new int [F[k]-1];
    memcpy(tem, a, n*sizeof(int));

    for(int i = n; i < F[k]-1; ++i)
        temp[i] = a[n-1];

    while(low <= high){
        int mid = low + F[k-1] - 1;
        if(key < temp[mid]){
            high = mid - 1;
            k-=1;
        }
        else if(key > temp[mid]){
            low = mid + 1;
            k-=2;
        }
        else{
            if(mid < n){
                return mid;    //若相等则说明mid即为查找到的位置
            }
            else{
                return n-1;
            }
        }
    }
    delete [] temp;
    return -1;

}
```