// 5.1

int findGap (int \* A, int N, int M){

int i, gap = 0;

if(N <= 1) return gap ;

if(N == 2) return A[0] <A [1] ? A[1] -A [0] : A[0] -A [1];

int \* array = (int \*) malloc (M \* sizeof ( int ));

for (i =0;i<M;i ++) array [i] = 0;

int j;

int B[N];

int dif;

for(j=0;j<N;j++)

array[A[j]]=C[A[j]]+1;

for(i=1;i<M;i++)

array[i]=array[i]+array[i-1];

for(j=N-1;j>=0;j--){

B[array[A[j]]]=A[j];

array[A[j]]--;

}

for(j=0;j<N-1;j++){

dif=B[j+1]-B[j];

if (gap < dif) gap = dif;

}

return gap ;

}

//5.2

double findGap ( double \* A, int N){

double gap = 0.0;

if (N <= 1) return gap;

if (N == 2) return A[0] <A [1] ? A[1] -A [0] : A[0] -A [1];

double max, min;

int i;

max = min = A[0];

for (i=1; i < N; ++i) {

if (A[i] > max) max = A[i];

if (A[i] < min) min = A[i];

}

double bar;

int pos;

bar = (max-min)/(N-1);

double bucket[N][2];

for (i=0; i<N; ++i) {

pos = (int)(A[i]-min)/bar;

if (bucket[pos][0] == 0)

bucket[pos][0] = bucket[pos][1] = A[i];

else {

if (A[i] > bucket[pos][1])

bucket[pos][1] = A[i];

if (A[i] < bucket[pos][0])

bucket[pos][0] = A[i];

}

}

double dif;

for (i=1; i<N; ++i) {

if (bucket[i][0] != 0) {

dif = bucket[i][0] - bucket[i-1][1];

if (gap < dif) gap = dif;

}

}

return gap;

}