Sudo code:

S=’’ #input string

n=|input| #length of string

win=5 #window size

tup=3 #tuple size

vocabulary={“A”,”C”,”G”,”T”}

nvocab=4

nvocab\_tab=0

for i=1 to nvocab

for j=1 to nvocab

for k=1 to nvocab

vocab\_tab[nvocab\_tab]=vocabulary[i].vocabulary[j].vocabulary[$k]; #all the 3 letters combination

nvocab\_tab++

end

end

end

for i=1 to nvocab\_tab

hash\_tab[vocab\_tab[i]]=i #label each 3 letters with a integer ranging from 1 to 64

end

for i =1to n-win

index[i]=0

ntup=1

for j= i+1 to i+win

for k= j+1 to i+win

index[i]=index[i]+hash\_tab[S[i]S[j]S[k]]\*(nvocab\_tab\*\*ntup); # generate the index for each position. The index indicates a unique segment of 5 bp start from i. “\*\*” indicates power

ntup++

end

end

end

# for each position i, get ff[i] which is the latest position and less than i have the same index with i. If this index appears for the first time, label it as “x”.

for i=1 to n-win { # indicates whether an index has appear or not.

g\_hash[index[i]]=0;

}

for i=1 to n-win {

if(g\_hash[index[i]]==0){

f\_hash[index[i]]=i; # if an index appears for the first time, label the corresponding position as “x” and record the current position of this index as i

g\_hash[index[i]]=1;

ff[i]="x";

}

else{

ff[i]=f\_hash[index[i]]; #if an index not appears for the first time, label the latest position with the same index for i as the old position of this index

f\_hash[index[i]]=i; #update the current position of this index as i

}

}

For i=1 to n-win {

vote1[i]=0; #number of the same 5 bp segment as position i appears after i

pos1[i]=0; #the last position has the same index with i

ppll[i]=0; #length of segment in current tandem repeat

st[i]=0; #start position of current tandem repeat

}

pl=0; #segment length

str=0; #start position of a tandem repeat

for i=1 to n-win {

u=i;

br=1; #flag to break a tandem repeat when new kind of segment length emerging.

while(ff[u] ne "x" && br==1 && u>=str){ # for current position u, trace back until “x” all the position have the same index, add the count and record the furthest position; br is the flag to start a new tandem when the length of segment is changing u>=str make sure for the position in the same tandem the tracing back procedure will not pass starting position.

if(pl==0){ #if segment length hasn’t been defined, define the start position for a tandem repeat; segment length and trace back one step, add the count of repeat on the position by 1 and record the support position as current position.

str=ff[u];

pl=u-ff[u];

u=ff[u];

vote1[u]++;

pos1[u]=i;

ppll[u]=pl;

st[u]=str;

}

elsif((u-ff[u])==pl){ #if the segment length is not changing then the position is still in the same tandem repeat.

u=ff[u];

vote1[u]++;

pos1[u]=i;

ppll[u]=pl;

st[u]=str;

}

else{ #if the segment length is changing, start a new tandem repeat, define new start position and segment length.

if(ff[u]>str){ #the new start position can’t be earlier than the original start postion.

pl=u-ff[u];

str=ff[u];

u=ff[u];

vote1[u]++;

pos1[u]=i;

ppll[u]=pl;

st[u]=str;

}

br=0; #indicate a new tandem repeat starts

}

}

}

#Collect all the position with tandem repeat more than 3

u=0; #repeat position

v=0; #number of repeats

w=0; #length of a segment

for i=1 to n-win{

l=0;

if(ppll[i]>4){ #make sure the segment is longer than 4

for j=0 to ppll[i]-5 {

If(vote1[i+j]<3){ #make sure at least 4 repeat

l=1;

}

}

if(l==0){

if(st[i]>u){ #if the a new position pass the condition, print it out and update

if(u>0){

print u,"\t",v+1,"\t",w,"\n";

}

u=st[i];

v=vote1[i];

w=ppll[i];

}

}

else{

if(vote1[i]>v && u==st[i] && w==ppll[i]){

w=vote1[i];

}

}

}

}

}

print u,"\t",v+1,"\t",w,"\n";