Longest Palindromic Subsequence: An optimisation comparison

The following programs both run in O(n^2) time and use O(1) space to find the longest palindromic subsequence of a string.

The python solution looks at all pairs of characters which comply with the palindromic requirement, in the order of “odd” then “even” length sequences.

The C solution does the exact same thing, except analyses the sequences in parallel.

The following comparison analysis of each method.

Python Solution

```
import datetime

def lpss():

  start = datetime.datetime.now()

  seq = “aaasadsdafasdhjkagsdfjhasjbjflASHBFHJASKBHFKJASBFASSDHJGVAHDGVsakhjfdsakjfuadsfk”
  n = len(seq) - 1
  max = 0

  # odd
  for i in range(1, n):
    low = i - 1
    high = i + 1
    currMax = 1
    while low >= 0 and high <= n and seq[low] == seq[high]:
      low-=1
      high+=1
      currMax = currMax + 2
    if currMax > max:
      max = currMax

  # even
  for i in range(0, n):
    low = i
    high = i + 1
```

```
    currMax = 0
    while low >= 0 and high <= n and seq[low] == seq[high]:
      low-=1
      high+=1
      currMax = currMax + 2
    if currMax > max:
      max = currMax

  end = datetime.datetime.now()
  diff = end - start
  print(diff)

lpss()
```

- This code prints a values of around 1000 microseconds.

C Solution

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <time.h>

// main prog
struct str{
  char* seq;
  int len;
};

void *odd(void* arg){
  struct str index = *(struct str*)arg;
  int maxAns = 1;
  for(int i = 1; i < index.len; i++){
    int low = i - 1;
    int high = i + 1;
    int currMax = 1;
    while(low >= 0 && high < index.len && index.seq[low] ==
index.seq[high]){
      low--;
      high++;
      currMax=currMax+2;
    }
    if(currMax > maxAns){
      maxAns = currMax;
    }
  }
  int* res = malloc(sizeof(int));
  *res = maxAns;
  free(arg);
  return (void*)res;
```

```
}

void *even(void* arg){
  struct str index = *(struct str*)arg;
  int maxAns = 0;
  for(int i = 0; i < index.len; i++){
    int low = i;
    int high = i + 1;
    int currMax = 0;
    while(low >= 0 && high < index.len && index.seq[low] ==
index.seq[high]){
      low--;
      high++;
      currMax=currMax+2;
    }
    if(currMax > maxAns){
      maxAns = currMax;
    }
  }
  int* res = malloc(sizeof(int));
  *res = maxAns;
  free(arg);
  return (void*)res;
}

// timer
long timediff(clock_t t1, clock_t t2) {
    long elapsed;
    elapsed = ((double)t2 - t1) / CLOCKS_PER_SEC * 1000000;
    return elapsed;
}

int main(void){
  // timer
  clock_t ti1, ti2;
  int i;
  float x = 2.7182;
  long elapsed;

  ti1 = clock();

  // main program
  char seq0[] =
“aaasadsdafasdhjkagsdfjhasjbjflASHBFHJASKBHFKJASBFASSDHJGVAHDGVsakhjfdsakjf
uadsfk”;
  int len = sizeof(seq0)/sizeof(seq0[0])-1;

  struct str* s0 = malloc(sizeof(struct str));
  struct str* s1 = malloc(sizeof(struct str));
  s0->seq = (char*)seq0;
  s1->seq = (char*)seq0;
  s0->len = len;
```

```
  s1->len = len;

  pthread_t t0;
  pthread_t t1;
  int* res0;
  int* res1;
  if (pthread_create(&t0, NULL, &odd, s0)!=0){
    return 0;
  }
  if (pthread_create(&t1, NULL, &even, s1)!=0){
    return 00;
  }
  if(pthread_join(t0, (void**)&res0)!=0){
    return 1;
  }
  if(pthread_join(t1, (void**)&res1)!=0){
    return 11;
  }

  if(*res0 > *res1){
    printf("%d\n", *res0);
  }else{
    printf("%d\n", *res1);
  }

  free(s0);
  free(s1);

  // end timer
  ti2 = clock();

  elapsed = timediff(ti1, ti2);
  printf("elapsed: %ld microseconds\n", elapsed);
  return 0;
}
```

- This code prints a value of around 300 microseconds.