CS 4504
Parallel and Distributed Computing

Project 2 Lab

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Parallel and Distributed Computing

Assignment

- Given two emoji strings s1 and s2. Write a Pthread program to find out the number of sub-emoji-strings, in string s1, that is exactly the same as s2.

S1 = 🍈🍉🦉🦉❤️🦉🥭🍒🥭🫐❤️🦉

S2 = ❤️🦉

Assignment Examples

- number_subEmojiStrings("🥎🍈🍉🍊🎊🍈🍉🥭🍈", "🍈🍉") = 2

- number_subEmojiStrings("🥎🥎🥎", "🥎") = 3
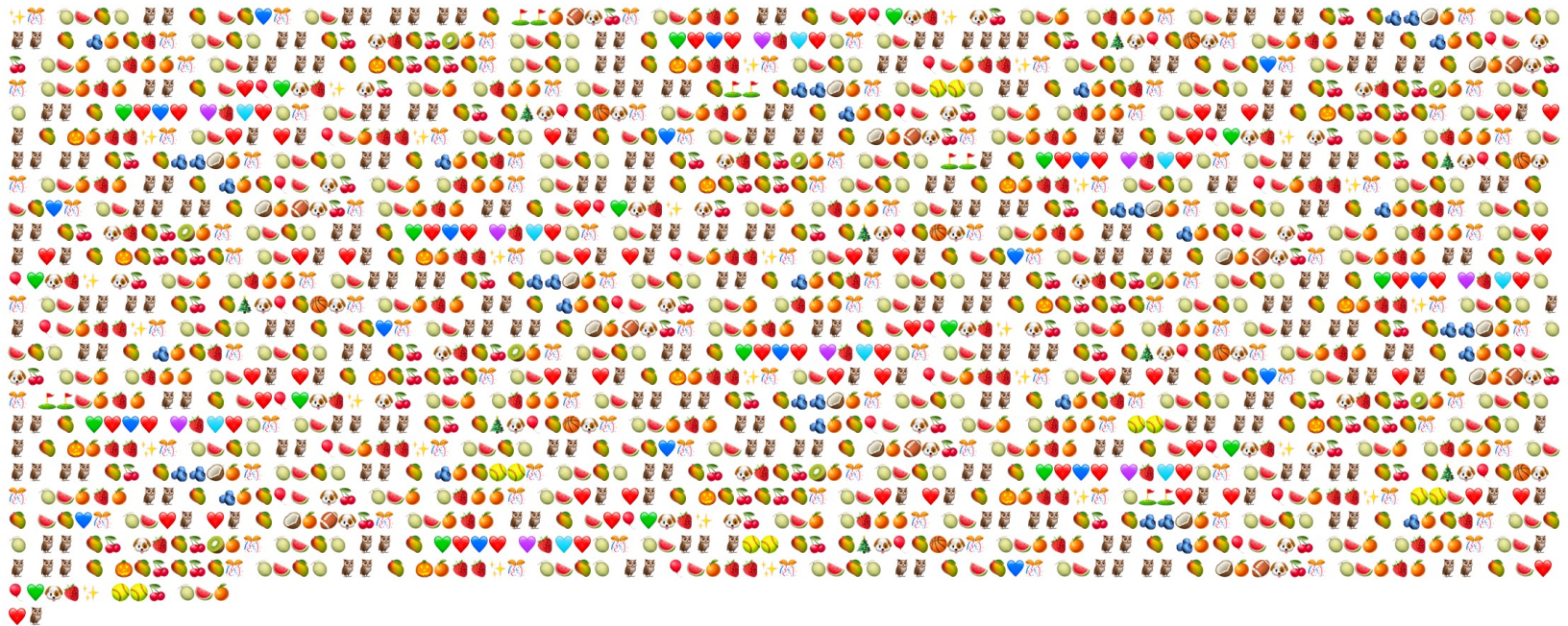
- number_subEmojiStrings("💚❤️💙❤️", "💚💙") = 0

> Subsequence not substring

Assignment

- Input file:

  https://raw.githubusercontent.com/kevinsuo/CS4504/main/emoji.txt

```c
int total = 0;
int n1,n2;
char *s1,*s2;
FILE *fp;

int readf(FILE *fp)
{
    if((fp=fopen("strings.txt", "r"))==NULL){
        printf("ERROR: can't open string.txt!\n");
        return 0;
    }
    s1=(char *)malloc(sizeof(char)*MAX);
    if(s1==NULL){
        printf("ERROR: Out of memory!\n");
        return -1;
    }
    s2=(char *)malloc(sizeof(char)*MAX);
    if(s2==NULL){
        printf("ERROR: Out of memory\n");
        return -1;
    }
    /*read s1 s2 from the file*/
    s1=fgets(s1, MAX, fp);
    s2=fgets(s2, MAX, fp);
    n1=strlen(s1);  /*length of s1*/
    n2=strlen(s2);   /*length of s2*/

    if(s1==NULL || s2==NULL || n1<n2)  /*when error exit*/
        return -1;
    return 0;
}
```

```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

S1= 🍈🍉🦉🦉 … … … 🥭🫐❤️🦉

S2= ❤️🦉
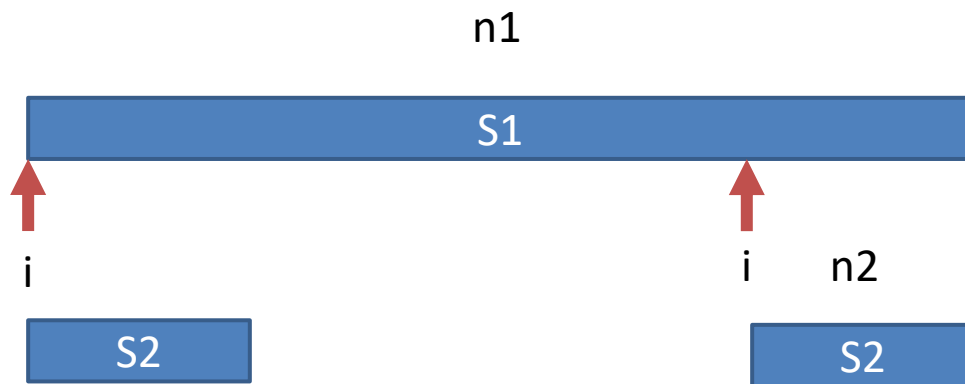
```c
int num_subEmojiString(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){   /*search for the next string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;            /*find a substring in this step*/
            }
        }
    }
    return total;
}
```

```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_subEmojiString();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```

n1

S1

i   i   n2

S2   S2

```c
int num_subEmojiString(void)
{
        int i,j,k;
        int count;


        for (i = 0; i <= (n1-n2); i++){
                count=0;
                for(j = i,k = 0; k < n2; j++,k++){      /*search for the next string of size of n2*/
                        if (*(s1+j)!=*(s2+k)){
                                break;
                        }else{
                                count++;
                        }

                        if(count==n2){
                                total++;                 /*find a substring in this step*/
                        }
                }
        }
        return total;
}
```
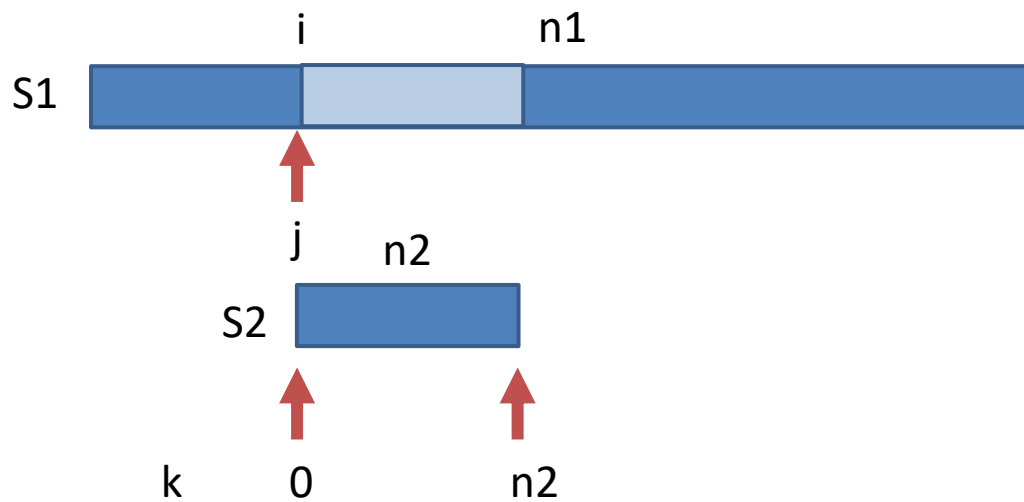
```c
int main(int argc, char *argv[])
{
        int count;

        readf(fp);
        count = num_subEmojiString();
        printf("The number of substrings is: %d\n", count);
        return 1;
}
```
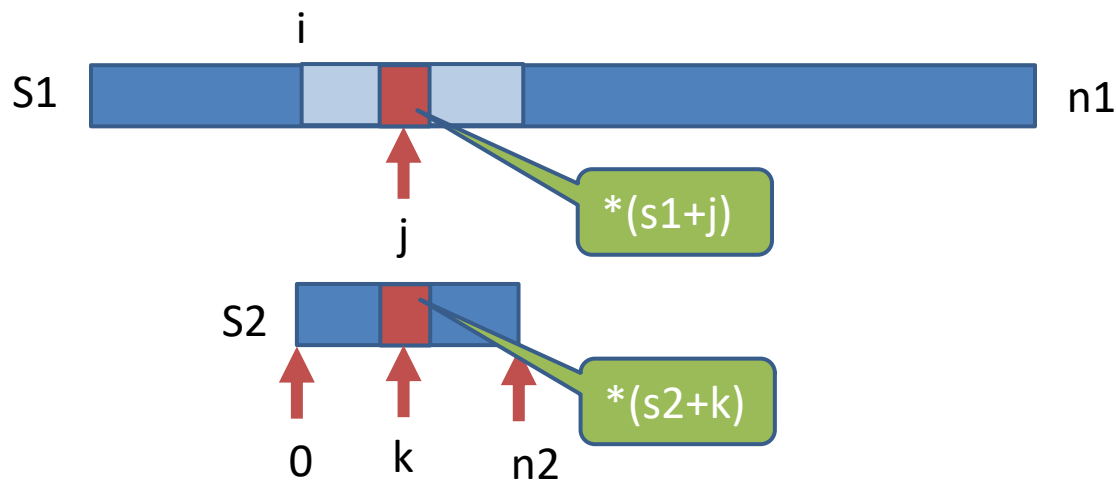

S1

i   n1

j
S2   n2

k   0   n2

```c
int num_subEmojiString(void)
{
    int i,j,k;
    int count;

    for (i = 0; i <= (n1-n2); i++){
        count=0;
        for(j = i,k = 0; k < n2; j++,k++){   /*search for the next string of size of n2*/
            if (*(s1+j)!=*(s2+k)){
                break;
            }else{
                count++;
            }

            if(count==n2){
                total++;              /*find a substring in this step*/
            }
        }
    }
    return total;
}
```

```c
int main(int argc, char *argv[])
{
    int count;

    readf(fp);
    count = num_subEmojiString();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```


S1 n1

*(s1+j)

i

j

S2

*(s2+k)

0   k   n2
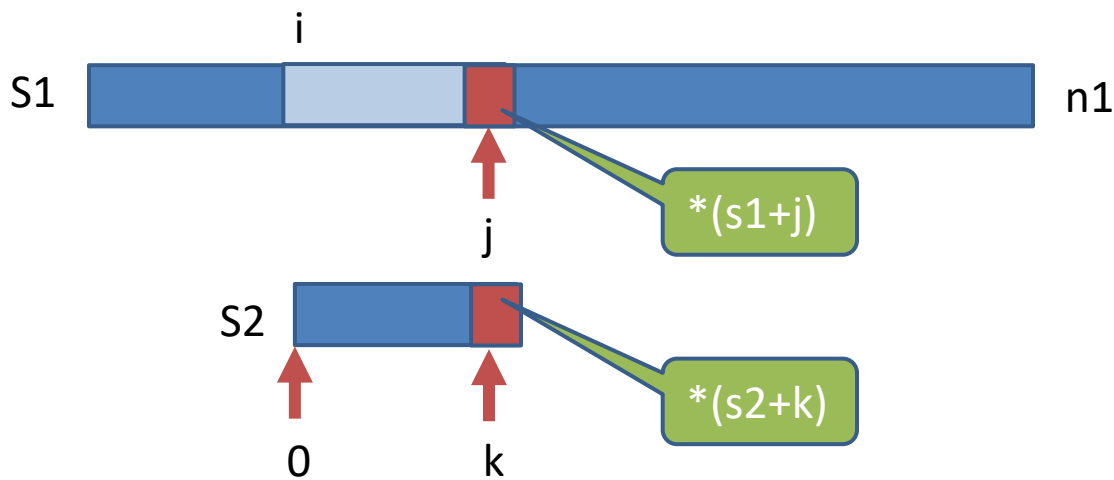
```c
int num_subEmojiString(void)
{
        int i,j,k;
        int count;


        for (i = 0; i <= (n1-n2); i++){
                count=0;
                for(j = i,k = 0; k < n2; j++,k++){  /*search for the next string of size of n2*/
                        if (*(s1+j)!=*(s2+k)){
                                break;
                        }else{
                                count++;
                        }


                        if(count==n2){
                                total++;               /*find a substring in this step*/
                        }
                }
        }
        return total;
}
```

```c
int main(int argc, char *argv[])
{
        int count;

        readf(fp);
        count = num_subEmojiString();
        printf("The number of substrings is: %d\n", count);
        return 1;
}
```


i

S1   n1

*(s1+j)

j

S2

*(s2+k)

0   k

Assignment

```c
int main(int argc, char *argv[])
{
        int count;

        readf(fp);
        count = num_subEmojiString();
        printf("The number of substrings is: %d\n", count);
        return 1;
}
```
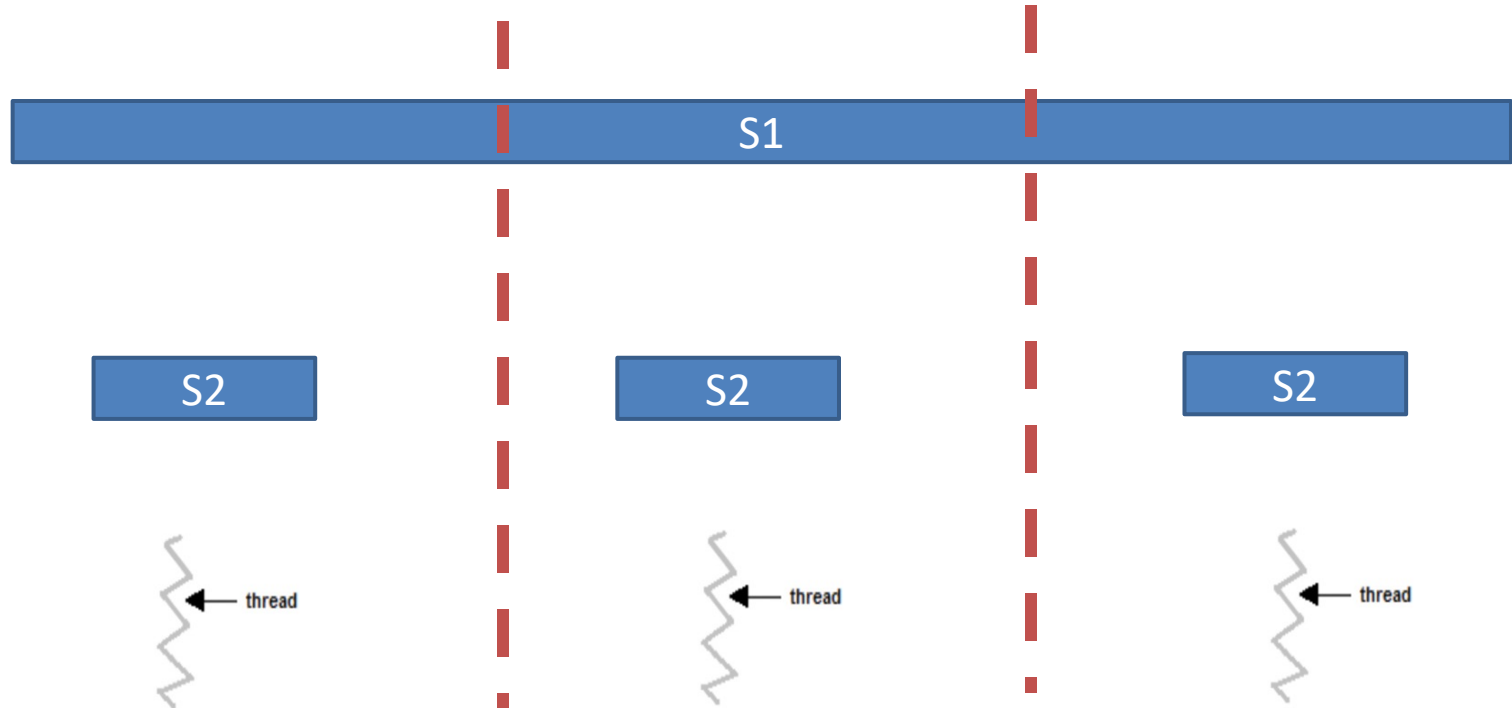
(Different text files output is also different. For the emoji.txt, the output is 55)

```
ksuo@LinuxKernel2 ~> ./project-pthread.o
The number of substrings is: 320
```
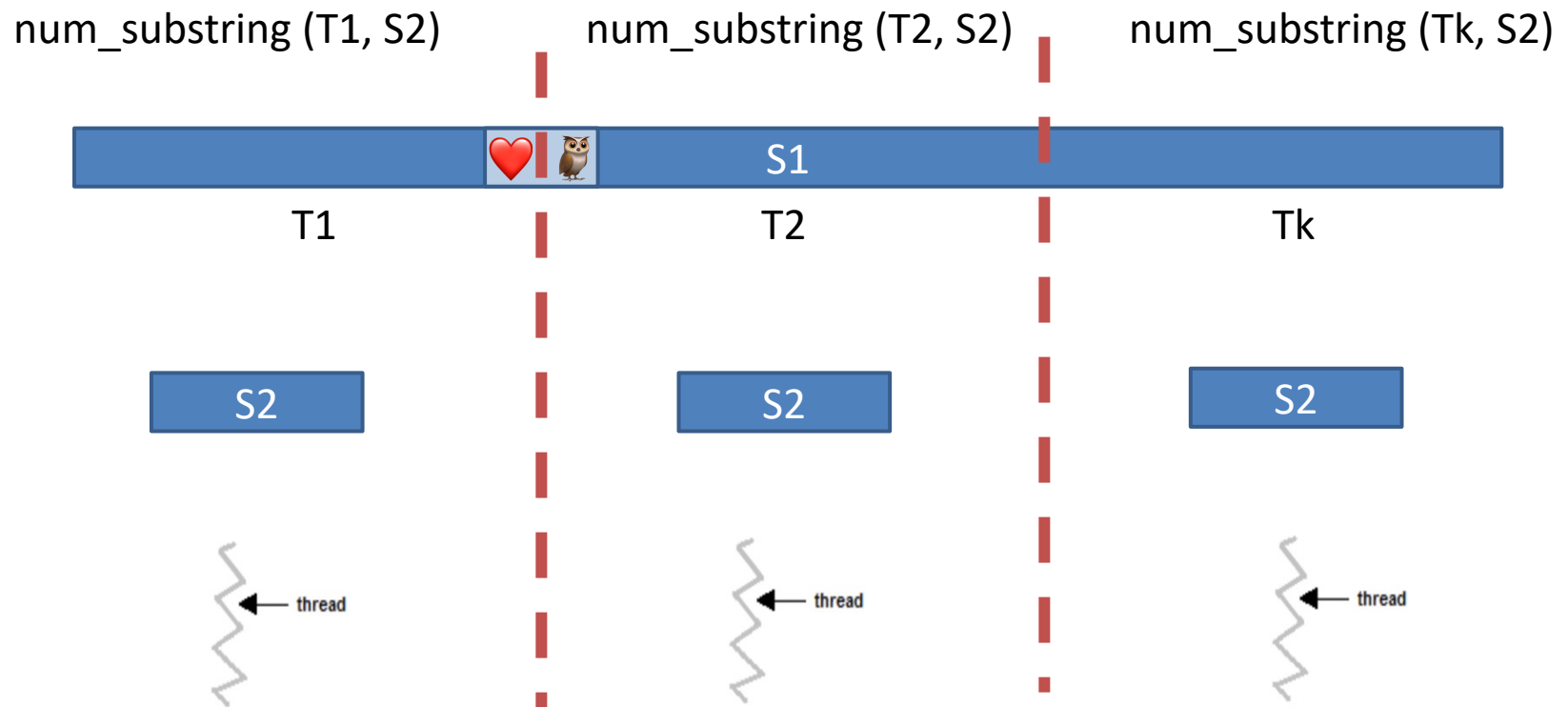
Idea

- Write a parallel program using Pthread based on this sequential solution.

Corner Case

num_substring (T1, S2)   num_substring (T2, S2)   num_substring (Tk, S2)

| ❤️ 🦉 | S1 | |

T1   T2   Tk

S2   S2   S2

thread   thread   thread

Verify whether your parallel thread is correct

- Modify the emoji.txt by yourself

- Compare the sequential and parallel program results that whether the total numbers are the same

Submission

1. source code

2. output screenshot of your parallel code

3. a report describe your code logic