作业13

KMP

运行下面的代码，结果是：

```
A B C D A A B C D A B C G
-1 0 0 0 0 1 1 2 3 4 5 2 3
```

```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

vector<int> buildNext(string p){
    int m = p.size(), j = 0;
    vector<int> N(m, 0);
    int t = N[0] = -1;
    while(j < m - 1){
        if(0 > t || p[j] == p[t]){
            j++; t++;

            N[j] = t;
        }else{
            t = N[t];
        }
    }
    return N;
}


int main(){
    string p = "ABCDAABCDABCG";
    vector<int> N = buildNext(p);
    for (int i = 0; i < p.size(); i++){
        cout << p[i] << " ";
    }
    cout << endl;
    for(int i = 0; i < N.size(); i++){
        cout << N[i] << " ";
    }
    cout << endl;
}
```

并发

计算的结果是832040。我基于的:F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)。

```
ziqianzhang@ziqiandeMac-Studio Parallel % g++ -o main ./main.cpp
ziqianzhang@ziqiandeMac-Studio Parallel % ./main
The 30th fibonacci number is 832040
ziqianzhang@ziqiandeMac-Studio Parallel %
```

计算的方法有一点tricky的地方，就是给线程分配ID的时候，如果发现超过了那就不能再分配了。

> 关于为什么是O（1），我的理解是线程的数量最大是固定的m，那么每个线程里面消耗的空间比如那些变量，我计为n。那么总空间就是个常数，跟输入的30没关系。

```
#include <iostream>
#include <thread>
#include <mutex>

using namespace std;

const int MAX_THREAD_NUM = 50;


// F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)
class Thread_Args
{
public:
    int n;
    int result;
    int threadID;
};

void *fibonacci(void *arg) {
    // 把参数转换为Thread_Args类型
    Thread_Args *args = (Thread_Args *)arg;


    if (args->n == 0 || args->n == 1){
        args->result = args->n;
        return NULL;
    }

    // 检查是否超过最大线程数
    if (args->threadID >= MAX_THREAD_NUM) {
        // 用循环代替递归
        int a = 0, b = 1;
        for (int i = 2; i <= args->n; i++) {
            int temp = a + b;
            a = b;
            b = temp;
        }
```

```
38            args->result = b;
39            return NULL;
40        }
41
42        // 创建两个线程
43        // 这里ID的计算方式是父线程ID * 2 和 父线程ID * 2 + 1 有一点tricky的地方
44        // 初始化的时候，父线程ID是1，所以两个子线程的ID分别是2和3
45
46        // 然后2的子线程ID是4和5，3的子线程ID是6和7
47        // 以此类推，每个线程的子线程ID都是父线程ID * 2 和 父线程ID * 2 + 1
48        Thread_Args args1, args2;
49        args1.n = args->n - 1;
50        args1.threadID = args->threadID * 2;
51        args2.n = args->n - 2;
52        args2.threadID = args->threadID * 2 + 1;
53
54        pthread_t thread1, thread2;
55        pthread_create(&thread1, NULL, fibonacci, &args1);
56        pthread_create(&thread2, NULL, fibonacci, &args2);
57
58        // 等待两个线程结束
59        pthread_join(thread1, NULL);
60        pthread_join(thread2, NULL);
61
62        // 计算结果
63        args->result = args1.result + args2.result;
64        return NULL;
65    }
66
67
68
69    int main() {
70        Thread_Args argInit;
71        argInit.n = 30;
72        argInit.threadID = 1;
73        fibonacci(&argInit);
74        cout << "The 30th fibonacci number is " << argInit.result << endl;
75        return 0;
76    }
77
```