Exercises for lab 6 of CS2101a

Instructor: Marc Moreno Maza, TA: Xiaohui Chen

November 28, 2012

1 Exercise 1

The file listed below contains a `C` program for the `Longest Common Subsequence` problem:

`http://en.wikipedia.org/wiki/Longest_common_subsequence_problem`

1. Modify (if necessary) this program such that you can compile with the `cilk++` compiler. Then Parallelize it! **Alternatively**, you can write your own `Cilk++` program from scratch.
2. Compare the running times of your parallel program and its serial counterpart on sufficiently large examples.
3. Analyze the work and the span of your program. To this end, you can conduct complexity analysis and/or use `cilview`.

```
/*
 * File:   main.c
 * Author: umel
 *
 * Created on May 13, 2010, 1:23 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
/*
 * This file compute the LCS of two strings
 */

void scan(int i_start,int  i_end, int j_start, int j_end, char *st1, char *st2, int **c, int **b){
    int i, j;
    for(i = i_start; i <= i_end; i++){
        for(j = j_start; j<= j_end; j++){
            if(st1[i-1] == st2[j-1]){
                c[i][j] = c[i-1][j-1] + 1;
                b[i][j] = 1;
            }
            else if(c[i-1][j] >= c[i][j-1]){
                c[i][j] = c[i-1][j];
                b[i][j] = 2;
            }
            else{
                c[i][j] = c[i][j-1];
```

```
                b[i][j] = 3;
            }
        }
    }
}

void scan_1(int st1_start, int st1_end, int st2_start, int st2_end, char *st1,char  *st2, int **c, int **b, int threshold){
    int part1 = (st1_end-st1_start + 1) / 3;
    int part2 = (st2_end-st2_start + 1) / 3;
    if((st1_end-st1_start + 1) == threshold)
        scan(st1_start, st1_end, st2_start, st2_end, st1, st2, c, b);
    else{
        scan_1( st1_start, st1_start + (part1 - 1), st2_start, st2_start + (part2 - 1), st1, st2, c, b, threshold);
        scan_1( st1_start + part1 , st1_start+(2*part1)-1, st2_start, st2_start + (part2 - 1), st1, st2, c, b, threshold);
        scan_1( st1_start, st1_start + (part1 -1), st2_start + part2, st2_start + (2*part2)-1, st1, st2, c, b, threshold);
        scan_1( st1_start + 2*part1, st1_end, st2_start, st2_start + (part2 - 1), st1, st2, c, b, threshold);
        scan_1( st1_start + part1, st1_start+(2*part1)-1, st2_start + part2, st2_start+(2*part2)-1, st1, st2, c, b, threshold
        scan_1( st1_start, st1_start + (part1 - 1), st2_start + 2*part2, st2_end, st1, st2, c, b, threshold);
        scan_1( st1_start + 2*part1, st1_end, st2_start + part2, st2_start + (2*part2) - 1, st1, st2, c, b, threshold);
        scan_1( st1_start + part1, st1_start + (2*part1) - 1, st2_start + 2*part2, st2_end, st1, st2, c, b, threshold);
        scan_1( st1_start + 2*part1, st1_end, st2_start + 2*part2, st2_end, st1, st2, c, b, threshold);
    }
}

void LCS(char *st1, char *st2, int **c,int **b){
    int m = strlen(st1)+1;
    int n = strlen(st2)+1;
    int i;
    //int th = (n-1)/3;
    int th = 81;
    for(i = 0; i < m; i++)
        c[i][0] = 0;
    for(i = 0; i < n; i++)
        c[0][i] = 0;
    //scan(1, m-1, 1, n-1, st1, st2, c, b);

    scan_1(1, m-1, 1, n-1, st1, st2, c, b, th);
}

void print_lcs(int **b, char *st1, int i, int j){
    if(i == 0 || j == 0)
        return;
    if(b[i][j] == 1){
        print_lcs(b, st1, i-1, j-1);
        printf("%c",st1[i-1]);
    }
    else if(b[i][j] == 2)
        print_lcs(b, st1, i-1, j);
    else
        print_lcs(b, st1, i, j-1);
}

int main(int argc, char** argv) {
    char *st1, *st2,st_t1[19684],st_t2[19684];
    int **c, **b;
    int m,n;
    int i;
    m = pow(3,8);
    n = m;
    st1 = (char *)malloc((m+1)*sizeof(char));
    st2 = (char *)malloc((n+1)*sizeof(char));
    //char st1[m+1], st2[m+1];
    for(i = 0; i < m; i++){
        st1[i] = rand()%27 + 65;
        st2[i] = rand()%27 + 65;
    }
    st1[i] = '\0';
    st2[i] = '\0';
```

```
    printf("\n Its ok\n");
    //st1 = "uvwxyzaabcdefghijklmnopqrst";
    //st2 = "abcdefghijklmnopqrstuvwxyza";
    //st1="aafdsfdfgfsdfstyuhjgfdretfg";
    //st2="fsdffghdfsdfxguikloijhuymkj";
    m = strlen(st1) + 1;
    n = strlen(st2) + 1;

    c = (int **)malloc(m*sizeof(int *));
    b = (int **)malloc(m*sizeof(int *));
    for(i = 0; i < m ; i++){
        c[i] = (int*)malloc(n*sizeof(int));
        b[i] = (int*)malloc(n*sizeof(int));
    }

    LCS(st1, st2, c, b);
    //print_lcs(b, st1, m-1, n-1);
    return (EXIT_SUCCESS);
}
```

2 Exercise 2

Remember the *merge-sort* algorithm:

`http://en.wikipedia.org/wiki/Merge_sort`

1. Write a **serial** `Cilk++` program (thus with no parallel constructs) that implement the *merge-sort* algorithm. To that end, you can modify the program of Exervise 1.
2. Then Parallelize it!
3. Compare the running times of your parallel program and its serial counterpart on sufficiently large examples.
4. Analyze the work and the span of your program. To this end, you can conduct complexity analysis and/or use `cilview`.
5. As you can see the ratio work to span, that is, the parallelism is quite low! This explains the poor speedup factor of your parallel program. Propose a strategy to resolve this situation.