Project 2

Project 2
1. Algorithm description
2. Asymptotic worst-case time complexity
3. How to compile and use
4. Sample input and output
5. Identify integers *x* and *y* that yield the largest possible number of distinct LCS's when *n* = 14, when *n* = 15, and when *n* = 16
    (1) n=14
    (2) n=15
    (3) n=16

1. Algorithm description

The main idea of the algorithm is using dynamic programming.

- dp[i][j] in the dp array denotes the length of longest commom subsequence between $X_i = x_1x_2\ldots\ldots x_i$ and $Y_i = y_1y_2\ldots\ldots y_i$
- So we can have the following formula:
    - when $x_i = y_j$, if i = 0 or j = 0, dp[i][j] = 0, else dp[i][j] = dp[i-1][j-1]+1
    - when $x_i \neq y_j$, if i = 0 or j = 0, dp[i][j] = 0, else dp[i][j] = max(dp[i-1][j], dp[i][j-1])
- At first, we change int x and int y to binary strings X_n and Y_n
- Iterating from $i = 0, j = 0$ to $i = n, j = n$, we get the length of longest commom subsequence between X_n and Y_n in dp[n][n].

```
void LCS(string X, string Y, int n)
```

- Then we track back to get the set of distinct lcs using the dp array we got from LCS function.

```
vector<string> Traceback_LCS(string &X, string &Y, int i, int j)
```

We can do the following things: (i and j are initially set to n)

- if dp[i][j] = 0, we just return "". Which means we already got the first chr of the lcs
- when $x_i = y_j$, it means that we build the lcs from the lcs of X_{i-1} and Y_{i-1}

  So we call Traceback_LCS(X, Y, i - 1, j - 1), and append x_i to the end of all lcs of (X_{i-1}, Y_{j-1})
- when $x_i \neq y_j$, it means that we can build the lcs from (X_{i-1}, Y_j) or (X_i, Y_{j-1})
  1. the end of the lcs can only be x_i or y_j (It's binary string and $x_i \neq y_j$)
  2. if the end of the lcs is y_j, we first find the index p of last chr of X_i that is y_j. Then if dp[p][j] = dp[i][j], we call Traceback_LCS(X, Y, p, j)
  3. if the end of the lcs is x_i, we first find the index q of last chr of Y_j that is x_i. Then if dp[i][q] = dp[i][j], we call Traceback_LCS(X, Y, i, q)
  4. we add this two parts together to get all lcs.

2. Asymptotic worst-case time complexity

The worst-case time complexity is $\theta(n^2 + 2nk)$

(k is the determined number of distinct LCS's)

Analysis:

For change int x and int y to binary strings X_n and Y_n, it will take $T_1(n) = \theta(n)$

For the bottom-up dp process of the LCS function, it will iterate $(n+1)^2$ times, so the time complexity is $T_2(n) = \theta(n^2)$

For the trace back process of the Traceback_LCS function:

- for each unique LCS, we will at worst need to go through both i, j from n to 0
- let k be the determined number of distinct LCS's
- we at worst need to do $2n$ subtraction to all k distinct LCS
- so the worst-case time complexity for Traceback_LCS function will be $\theta(2nk)$

Therefore, the total time complexity $T(n) = T_1(n) + T_2(n) + T_3(n) = \theta(n^2 + 2nk)$.

3. How to compile and use

Using the following command to compile:

```
g++ LCS.cpp -o LCS -O3
```

Using the following command to use, among which the first param is name of Lcs program, the second is n, the third is x, and the last is y:

```
./LCS 14 12642 5735
```

Using the following command to measure the time:

```
time ./LCS 14 12642 5735
```

4. Sample input and output

Input 1: $n = 14, x = 12642, y = 5735$

Output 1:

```
n = 14; x = 12642; y = 5735
binstring(n,x) = 11000101100010
binstring(n,y) = 01011001100111
the determined number of distinct LCS's = 12
the list of those LCS's:
110000111
110010111
100010111
110011001
100011001
000011001
111011001
101011001
001011001
010110001
010110000
010110010
./LCS_ad 14 12642 5735  0.00s user 0.00s system 52% cpu 0.010 total
```

Input2: $n = 20, x = 1048475, y = 524288$

Output 2:

```
n = 20; x = 1048475; y = 524288
binstring(n,x) = 11111111111110011011
binstring(n,y) = 10000000000000000000
the determined number of distinct LCS's = 1
the list of those LCS's:
1000
./LCS_ad 20 1048475 524288  0.00s user 0.00s system 63% cpu 0.006 total
```

5. Identify integers *x* and *y* that yield the largest possible number of distinct LCS's when *n* = 14, when *n* = 15, and when *n* = 16

(1) n=14

I wrote another two for loop to test all x from 0 to 2^n-1 and y from $x+1$ to 2^n-1 to get k_{max}, the largest possible number of distinct LCS's when n=14, when n=15, and when n=16. I also record x_{min}, the corresponding minimum x to achieve k_{max}.

- n = 14, $k_{max}=70$, $x_{min}=3171$
- n = 15, $k_{max}=96$, $x_{min}=6342$
- n = 16, $k_{max}=141$, $x_{min}=14563$

Then I use the same loop to record all (x, y) pairs that achieve k_{max} from x_{min}

There are 32 different (x, y) pairs that yield the largest possible number of distinct LCS's when *n* = 14.

I also record the result in 14.txt

```
x = 3171; y = 10922
x = 3185; y = 10922
x = 3187; y = 10922
x = 3299; y = 10922
x = 3633; y = 10922
x = 3635; y = 10922
x = 3641; y = 10922
x = 3683; y = 10922
x = 3697; y = 10922
x = 3699; y = 10922
x = 5461; y = 8988
x = 5461; y = 9100
x = 5461; y = 9102
```

```
x = 5461; y = 9116
x = 5461; y = 9830
x = 5461; y = 10012
x = 5461; y = 12684
x = 5461; y = 12686
x = 5461; y = 12700
x = 5461; y = 12742
x = 5461; y = 12748
x = 5461; y = 12750
x = 5461; y = 13084
x = 5461; y = 13196
x = 5461; y = 13198
x = 5461; y = 13212
x = 6371; y = 10922
x = 6553; y = 10922
x = 7267; y = 10922
x = 7281; y = 10922
x = 7283; y = 10922
x = 7395; y = 10922
```

(2) n=15

There are 70 different (x, y) pairs that yield the largest possible number of distinct LCS's when *n* = 15.

I also record the result in 15.txt

```
x = 6342; y = 21845
x = 6348; y = 21845
x = 6350; y = 21845
x = 6374; y = 21845
x = 6540; y = 21845
x = 6542; y = 21845
x = 6556; y = 21845
x = 6598; y = 21845
x = 6604; y = 21845
x = 6606; y = 21845
x = 7270; y = 21845
x = 7366; y = 21845
x = 7372; y = 21845
```

```
x = 7374; y = 21845
x = 7398; y = 21845
x = 10922; y = 17969
x = 10922; y = 17971
x = 10922; y = 17977
x = 10922; y = 18019
x = 10922; y = 18033
x = 10922; y = 18035
x = 10922; y = 18201
x = 10922; y = 18225
x = 10922; y = 18227
x = 10922; y = 18233
x = 10922; y = 19555
x = 10922; y = 19569
x = 10922; y = 19571
x = 10922; y = 19683
x = 10922; y = 20017
x = 10922; y = 20019
x = 10922; y = 20025
x = 10922; y = 20067
x = 10922; y = 20081
x = 10922; y = 20083
x = 10922; y = 25369
x = 10922; y = 25393
x = 10922; y = 25395
x = 10922; y = 25401
x = 10922; y = 25497
x = 10922; y = 26161
x = 10922; y = 26163
x = 10922; y = 26169
x = 10922; y = 26211
x = 10922; y = 26225
x = 10922; y = 26227
x = 10922; y = 26393
x = 10922; y = 26417
x = 10922; y = 26419
x = 10922; y = 26425
x = 12684; y = 21845
x = 12686; y = 21845
x = 12700; y = 21845
```

```
x = 12742; y = 21845
x = 12748; y = 21845
x = 12750; y = 21845
x = 13084; y = 21845
x = 13196; y = 21845
x = 13198; y = 21845
x = 13212; y = 21845
x = 14534; y = 21845
x = 14540; y = 21845
x = 14542; y = 21845
x = 14566; y = 21845
x = 14732; y = 21845
x = 14734; y = 21845
x = 14748; y = 21845
x = 14790; y = 21845
x = 14796; y = 21845
x = 14798; y = 21845
```

(3) n=16

There are 2 different (x, y) pairs that yield the largest possible number of distinct LCS's when *n* = 16.

I also record the result in 16.txt

```
x = 14563; y = 43690
x = 21845; y = 50972
```