Name: Yu Qin

Results Screenshots

==========start of the write-up==========

Lab 1: S-W gene align

```
$ cd lab1_SWalign/
yuq8@andromeda-30 00:10:49 ~/253P/hw_lab/5/lab1_SWalign
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 00:10:54 ~/253P/hw_lab/5/lab1_SWalign
$ ls
main*  main.cpp  main.dSYM/  Makefile
yuq8@andromeda-30 00:10:56 ~/253P/hw_lab/5/lab1_SWalign
$
```

test case 1:

```
 string S1 = "ACACACTA";
 string S2 = "AGCACACA";
```

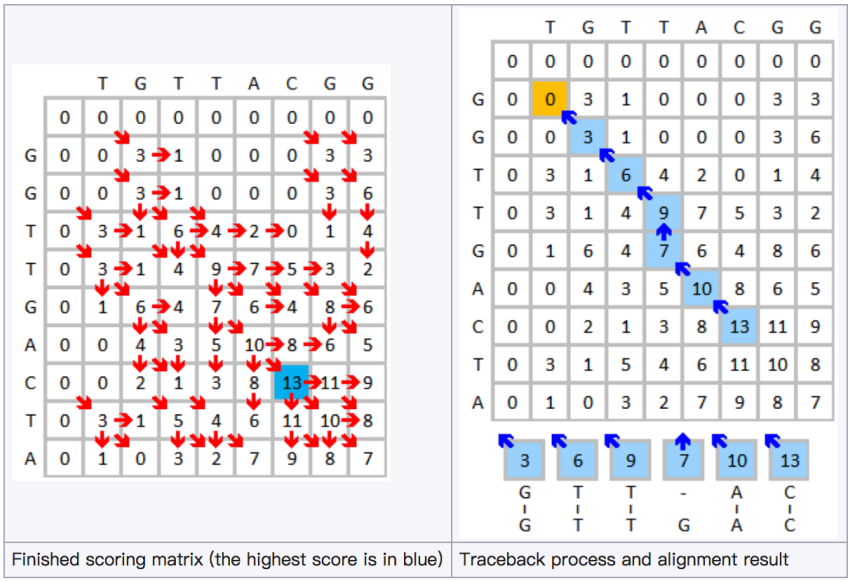
```
yuq8@andromeda-30 16:28:14 ~/253P/hw_lab/5/lab1_SWalign
$ valgrind ./main
==28975== Memcheck, a memory error detector
==28975== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==28975== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==28975== Command: ./main
==28975==
maxscore: 17
                A       G       C       A       C       A       C       A
        0       0       0       0       0       0       0       0       0
A       0       3       1       0       3       1       3       1       3
C       0       1       0       4       2       6       4       6       4
A       0       3       1       2       7       5       9       7       9
C       0       1       0       4       5       10      8       12      10
A       0       3       1       2       7       8       13      11      15
C       0       1       0       4       5       10      11      16      14
T       0       0       0       2       3       8       9       14      13
A       0       3       1       0       5       6       11      12      17
the back path found:
17
14
16
13
10
7
4
1
3
S1 = A-CACACTA
S2 = AGCACAC-A
==28975==
==28975== HEAP SUMMARY:
==28975==     in use at exit: 72,704 bytes in 1 blocks
==28975==   total heap usage: 122 allocs, 121 frees, 79,220 bytes allocated
==28975==
==28975== LEAK SUMMARY:
==28975==    definitely lost: 0 bytes in 0 blocks
==28975==    indirectly lost: 0 bytes in 0 blocks
==28975==      possibly lost: 0 bytes in 0 blocks
==28975==    still reachable: 72,704 bytes in 1 blocks
==28975==         suppressed: 0 bytes in 0 blocks
==28975== Rerun with --leak-check=full to see details of leaked memory
==28975==
==28975== For counts of detected and suppressed errors, rerun with: -v
==28975== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 16:28:26 ~/253P/hw_lab/5/lab1_SWalign
$
```

test case 2:

```
//string S1 = "GGTTGACTA";
//string S2 = "TGTTACGG";
```

```
$ valgrind ./main
==29474== Memcheck, a memory error detector
==29474== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==29474== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==29474== Command: ./main
==29474==
maxscore: 13
                T       G       T       T       A       C       G       G
        0       0       0       0       0       0       0       0       0
G       0       0       3       1       0       0       0       3       3
G       0       0       3       1       0       0       0       3       6
T       0       3       1       6       4       2       0       1       4
T       0       3       1       4       9       7       5       3       2
G       0       1       6       4       7       6       4       8       6
A       0       0       4       3       5       10      8       6       5
C       0       0       2       1       3       8       13      11      9
T       0       3       1       5       4       6       11      10      8
A       0       1       0       3       2       7       9       8       7
the back path found:
13
10
7
9
6
3
S1 = GTTGAC
S2 = GTT-AC
==29474==
==29474== HEAP SUMMARY:
==29474==     in use at exit: 72,704 bytes in 1 blocks
==29474==   total heap usage: 101 allocs, 100 frees, 78,140 bytes allocated
==29474==
==29474== LEAK SUMMARY:
==29474==    definitely lost: 0 bytes in 0 blocks
==29474==    indirectly lost: 0 bytes in 0 blocks
==29474==      possibly lost: 0 bytes in 0 blocks
==29474==    still reachable: 72,704 bytes in 1 blocks
==29474==         suppressed: 0 bytes in 0 blocks
==29474== Rerun with --leak-check=full to see details of leaked memory
==29474==
==29474== For counts of detected and suppressed errors, rerun with: -v
==29474== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 16:30:54 ~/253P/hw_lab/5/lab1_SWalign
$
```

compare with the right answer:



Finished scoring matrix (the highest score is in blue) | Traceback process and alignment result

The alignment result is:

```
G T T - A C
| | |   | |
G T T G A C
```

Conclusion: right result, 0 error in memory leak.
Little Notes:

1) Interesting and useful of DP.
2) The trace back method should be based on the feedforward, but in future work it may be simplified using memory since there are some common things.
3) S-W algorithm is different with N-W algorithm. I think future hw could be put them together as 2 problems.

Lab 2: Levenshtein distance

Test cases 1, 2, 3:

```
int main(){
    string S1 = "lawn";
    string S2 = "flaw";
    //string S1 = "Sunday";
    //string S2 = "Saturday";
    //string S1 = "sitting";
    //string S2 = "kitten";
    OptimalLocalAlign(S1, S2);
    return 0;
}
```

Test case 1:

```
$ pwd
/home/yuq8/253P/hw_lab/5/lab2_Levenshtein
yuq8@andromeda-30 15:55:42 ~/253P/hw_lab/5/lab2_Levenshtein
$ make
------------compiling main.cpp to create executable program main--------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 15:55:47 ~/253P/hw_lab/5/lab2_Levenshtein
```

```
$ valgrind ./main
==14413== Memcheck, a memory error detector
==14413== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==14413== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==14413== Command: ./main
==14413==
                f       l       a       w
        0       1       2       3       4
l       1       1       1       2       3
a       2       2       2       1       2
w       3       3       3       2       1
n       4       4       4       3       2
Levenshtein distance: 2
==14413==
==14413== HEAP SUMMARY:
==14413==     in use at exit: 72,704 bytes in 1 blocks
==14413==   total heap usage: 14 allocs, 13 frees, 73,164 bytes allocated
==14413==
==14413== LEAK SUMMARY:
==14413==    definitely lost: 0 bytes in 0 blocks
==14413==    indirectly lost: 0 bytes in 0 blocks
==14413==      possibly lost: 0 bytes in 0 blocks
==14413==    still reachable: 72,704 bytes in 1 blocks
==14413==         suppressed: 0 bytes in 0 blocks
==14413== Rerun with --leak-check=full to see details of leaked memory
==14413==
==14413== For counts of detected and suppressed errors, rerun with: -v
==14413== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 15:57:44 ~/253P/hw_lab/5/lab2_Levenshtein
$
```

Test case 2:

```
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 15:58:40 ~/253P/hw_lab/5/lab2_Levenshtein
$ valgrind ./main
==14637== Memcheck, a memory error detector
==14637== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==14637== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==14637== Command: ./main
==14637==
                S       a       t       u       r       d       a       y
        0       1       2       3       4       5       6       7       8
S       1       0       1       2       3       4       5       6       7
u       2       1       1       2       2       3       4       5       6
n       3       2       2       2       3       3       4       5       6
d       4       3       3       3       3       4       3       4       5
a       5       4       3       4       4       4       4       3       4
y       6       5       4       4       5       5       5       4       3
Levenshtein distance: 3
==14637==
==14637== HEAP SUMMARY:
==14637==     in use at exit: 72,704 bytes in 1 blocks
==14637==   total heap usage: 18 allocs, 17 frees, 73,580 bytes allocated
==14637==
==14637== LEAK SUMMARY:
==14637==    definitely lost: 0 bytes in 0 blocks
==14637==    indirectly lost: 0 bytes in 0 blocks
==14637==      possibly lost: 0 bytes in 0 blocks
==14637==    still reachable: 72,704 bytes in 1 blocks
==14637==         suppressed: 0 bytes in 0 blocks
==14637== Rerun with --leak-check=full to see details of leaked memory
==14637==
==14637== For counts of detected and suppressed errors, rerun with: -v
==14637== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 15:58:43 ~/253P/hw_lab/5/lab2_Levenshtein
```

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Test case 3:

```
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main
yuq8@andromeda-30 15:59:21 ~/253P/hw_lab/5/lab2_Levenshtein
$ valgrind ./main
==14818== Memcheck, a memory error detector
==14818== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==14818== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==14818== Command: ./main
==14818==
                k       i       t       t       e       n
        0       1       2       3       4       5       6
s       1       1       2       3       4       5       6
i       2       2       1       2       3       4       5
t       3       3       2       1       2       3       4
t       4       4       3       2       1       2       3
i       5       5       4       3       2       2       3
n       6       6       5       4       3       3       2
g       7       7       6       5       4       4       3
Levenshtein distance: 3
==14818==
==14818== HEAP SUMMARY:
==14818==     in use at exit: 72,704 bytes in 1 blocks
==14818==   total heap usage: 20 allocs, 19 frees, 73,564 bytes allocated
==14818==
==14818== LEAK SUMMARY:
==14818==    definitely lost: 0 bytes in 0 blocks
==14818==    indirectly lost: 0 bytes in 0 blocks
==14818==      possibly lost: 0 bytes in 0 blocks
==14818==    still reachable: 72,704 bytes in 1 blocks
==14818==         suppressed: 0 bytes in 0 blocks
==14818== Rerun with --leak-check=full to see details of leaked memory
==14818==
==14818== For counts of detected and suppressed errors, rerun with: -v
==14818== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 15:59:29 ~/253P/hw_lab/5/lab2_Levenshtein
$
```

Conclusion: right result, 0 error in memory leak.

Little Notes:

1) Levenshtein distance allows for insertions, deletions or substitutions. With different allowable edit operation rules, other edit distance are :
   a) Hamming Distance: only substitution, hence, it only applies to strings of the same length.
   b) Damerau–Levenshtein distance: allows insertion, deletion, substitution, and the transposition of two adjacent characters;
   c) longest common subsequence (LCS) distance: allows only insertion and deletion, not substitution;
   d) Jaro distance: allows only transposition
2) Levenshtein distance is mainly for short strings in many longer texts, for instance, spell checkers, correction systems for optical character recognition, and software to assist natural language translation based on translation memory. For long comparison, the compared strings are usually shortened to help improve speed of comparisons, eg. fuzzy string searching in applications such as record linkage.

Lab3:

```
void MaxCalculate(int* cn, int tv, int* cv){
    int size1 = 4 + 1;
    int size2 = tv + 1;
    int MAXCOIN = cn[0]+cn[1]+cn[2]+cn[3]+1;
    //cout<<size1<<endl;
    //cout<<size2<<endl;
    cout<<MAXCOIN<<endl;
    vector<vector<float>> H(size1, vector<float>(size2));//H[i][j],first i kinds of coins for j target value.
    //initialize
    pair<int, int> maxlocation;
    for (int i = 1; i < size1; i++){
        for (int j = 0; j < size2; j++){
            if (j == 0)
                H[i][0] = 0;
            else if (i == 1)
                if ( (j%cv[i-1] == 0) && (j%cv[i-1] <= cn[i-1]))
                    H[1][j] = j/cv[i-1];
                else
                    H[1][j] = 0;
            else
                H[i][j] = 0;
        }
    }
    //transfer
    cout<<"trans"<<endl;
    for (int i = 2; i < size1; i++){
        for (int j = 1; j< size2; j++){
            cout<<cn[i-1]<<endl;
            for (int k = 0; k <= cn[i-1]; k++){
                if( ((j - k * cv[i-1]) >= 0) && (H[i][j] < (H[i-1][j - k * cv[i-1]] + k)) )
                    H[i][j] = H[i-1][j - k * cv[i-1]] + k;
                //cout<<i<<","<<j<<","<<k<<endl;
            }
        }
    }
    //result
    cout<<"res"<<endl;
    int maxcoins = H[size1 -1 ][size2 -1];
```

```
$ cd lab3_coin/
yuq8@andromeda-30 15:10:34 ~/253P/hw_lab/5/lab3_coin
$ ls
input  main*  main.cpp  main_cppreg.cpp  main.dSYM/  Makefile
yuq8@andromeda-30 15:10:35 ~/253P/hw_lab/5/lab3_coin
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   main
------------Congratulation to you! Successfully compile.
------------Run manually by :
------------./main input
yuq8@andromeda-30 15:10:45 ~/253P/hw_lab/5/lab3_coin
```

```
$ valgrind ./main input
==32003== Memcheck, a memory error detector
==32003== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==32003== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==32003== Command: ./main input
==32003==
processing input line: 1
1 4 2 20 46
28
trans
res
maxcoins: 46
              0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18  19      20
21      22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39  40      41      42      43
44      45      46
        0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0   00      0
0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0   00      0       0       0
1       0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19  20      21
22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40  41      42      43      44
45      46
5       0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19  20      21
22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40  41      42      43      44
45      46
10      0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19  20      21
22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40  41      42      43      44
45      46
25      0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19  20      21
22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40  41      42      43      44
45      46
processing input line: 2
finished or wrong
==32003==
==32003== HEAP SUMMARY:
==32003==     in use at exit: 72,704 bytes in 1 blocks
==32003==   total heap usage: 15 allocs, 14 frees, 75,580 bytes allocated
==32003==
==32003== LEAK SUMMARY:
==32003==    definitely lost: 0 bytes in 0 blocks
==32003==    indirectly lost: 0 bytes in 0 blocks
==32003==      possibly lost: 0 bytes in 0 blocks
==32003==    still reachable: 72,704 bytes in 1 blocks
==32003==         suppressed: 0 bytes in 0 blocks
==32003== Rerun with --leak-check=full to see details of leaked memory
==32003==
==32003== For counts of detected and suppressed errors, rerun with: -v
==32003== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
yuq8@andromeda-30 15:11:02 ~/253P/hw_lab/5/lab3_coin
$
```

Conclusion: right result, 0 error in memory leak.

Little Notes:

1) Different parameters for min, max coin numbers.
2) There's something not finished, I'd like to finish it in some time.