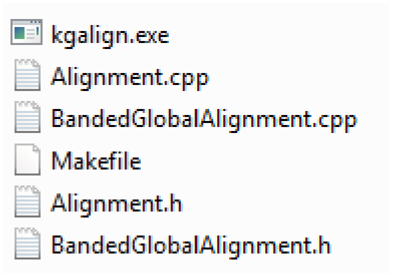
MATH 578A Programming Assignment #2

Haifeng Chen

- **Summary**

  As the Table 1 shown, kgalign.exe and galign.exe got the same optimal score on all datasets. However, results show kgalign.exe takes less space than galign.exe, and kgalign.exe takes more time than galign.exe. Figure 1 and 2 also show the same conclusion.

- **Code and executable file**



kgalign.exe
Alignment.cpp
BandedGlobalAlignment.cpp
Makefile
Alignment.h
BandedGlobalAlignment.h

-

- **Compiler**

  MinGW GCC 4.7.2

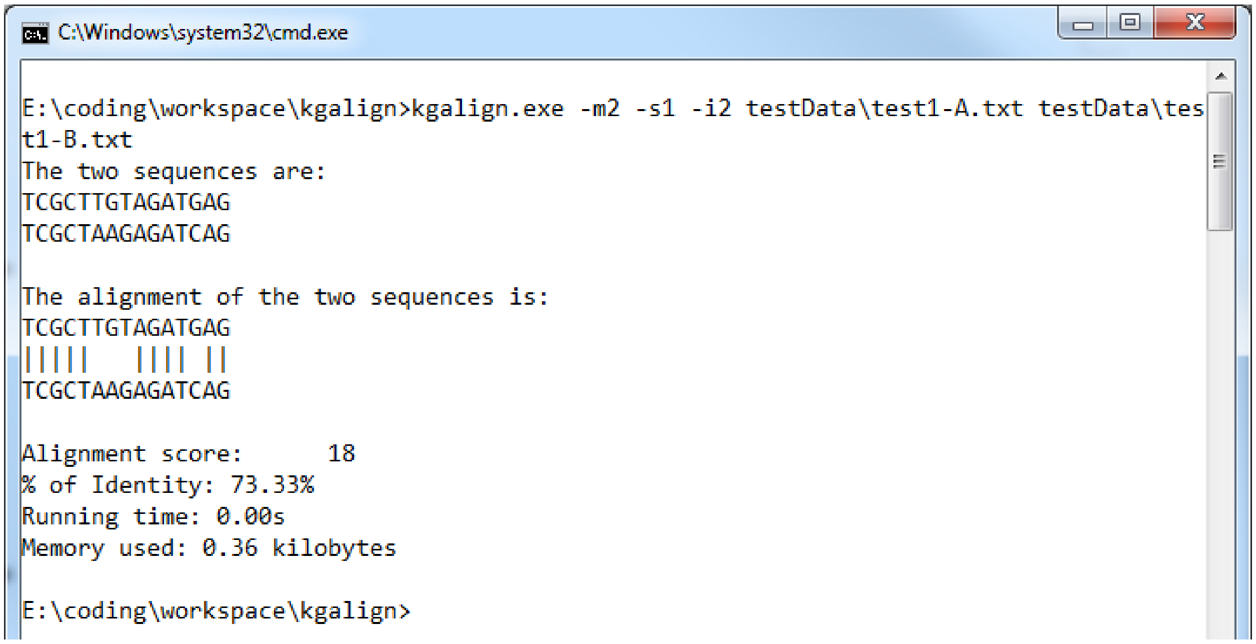
```
>g++ -o kgalign.exe Alignment.cpp BandedGlobalAlignment.cpp
```

- **Running Environment**

  Windows 7 Professional 32-bit
  Processor: Inter(R) Core(TM) 2 Duo CPU T6570 @2.10GHz 2.10GHz
  Installed memory (RAM): 2.00 GB

- **Results of test examples**

1) Result of the first example:

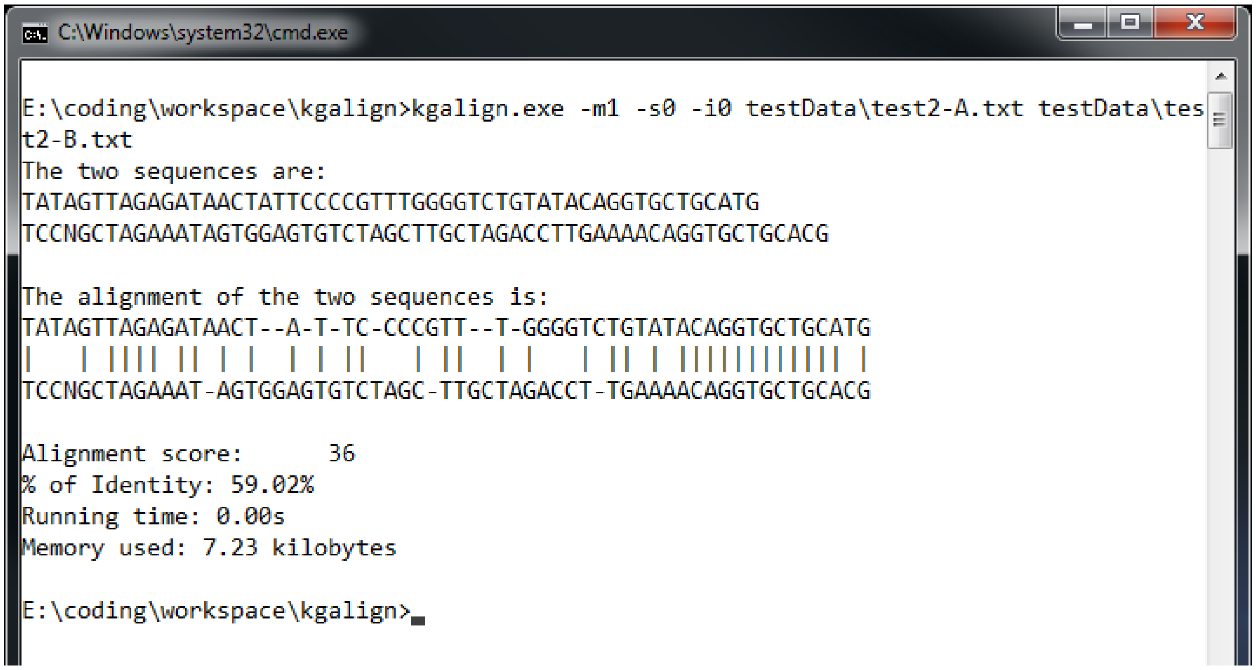

```
E:\coding\workspace\kgalign>kgalign.exe -m2 -s1 -i2 testData\test1-A.txt testData\tes
t1-B.txt
The two sequences are:
TCGCTTGTAGATGAG
TCGCTAAGAGATCAG

The alignment of the two sequences is:
TCGCTTGTAGATGAG
|||||   |||| ||
TCGCTAAGAGATCAG

Alignment score:      18
% of Identity: 73.33%
Running time: 0.00s
Memory used: 0.36 kilobytes

E:\coding\workspace\kgalign>
```

2) Result of the second example:

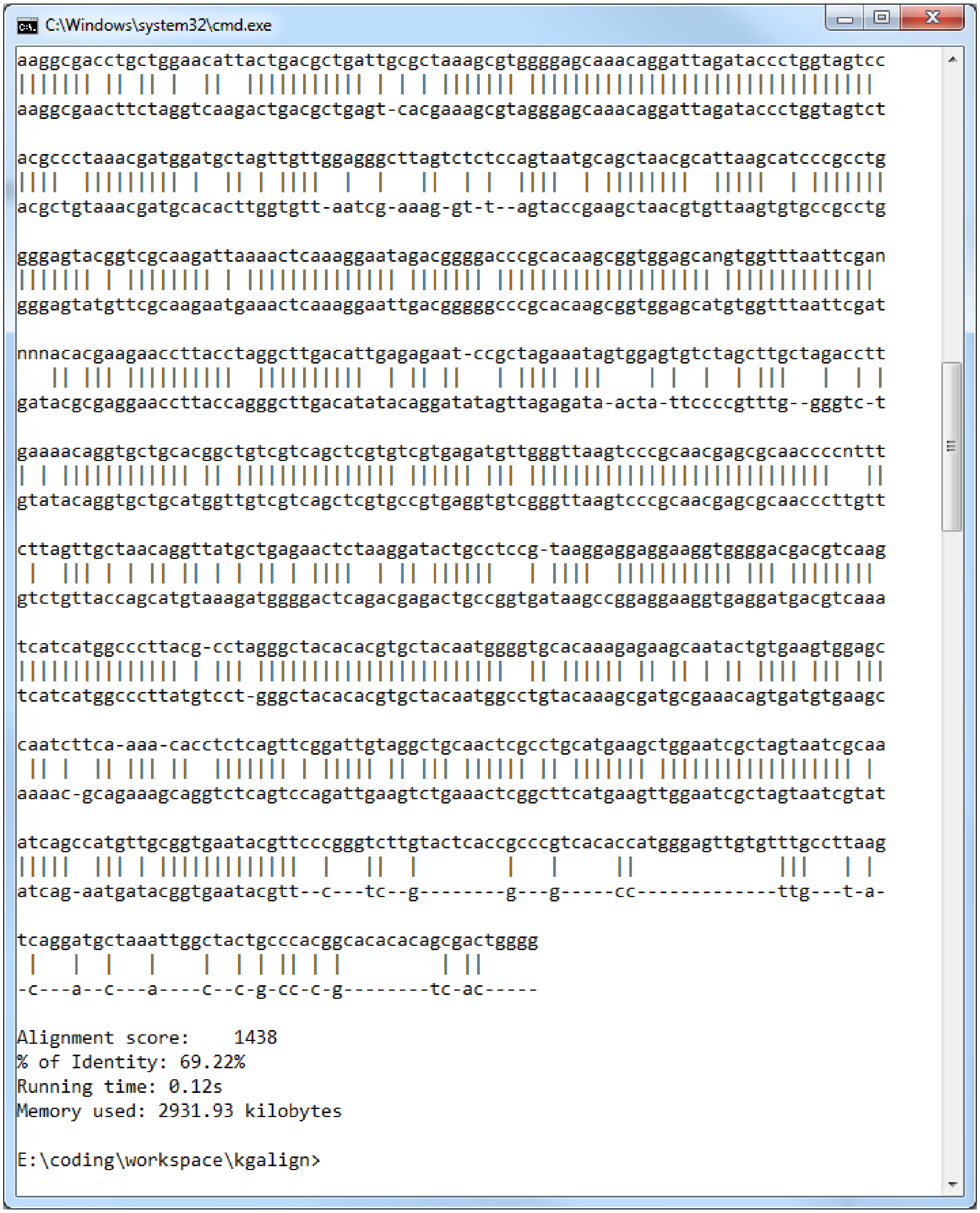

```
E:\coding\workspace\kgalign>kgalign.exe -m1 -s0 -i0 testData\test2-A.txt testData\tes
t2-B.txt
The two sequences are:
TATAGTTAGAGATAACTATTCCCCGTTTGGGGTCTGTATACAGGTGCTGCATG
TCCNGCTAGAAATAGTGGAGTGTCTAGCTTGCTAGACCTTGAAAACAGGTGCTGCACG

The alignment of the two sequences is:
TATAGTTAGAGATAACT--A-T-TC-CCCGTT--T-GGGGTCTGTATACAGGTGCTGCATG
|   | |||| || | |  | | ||   | ||  | |   | || | |||||||||||| |
TCCNGCTAGAAAT-AGTGGAGTGTCTAGC-TTGCTAGACCT-TGAAAACAGGTGCTGCACG

Alignment score:      36
% of Identity: 59.02%
Running time: 0.00s
Memory used: 7.23 kilobytes

E:\coding\workspace\kgalign>
```

3) Part result of the third example:



C:\Windows\system32\cmd.exe

```
aaggcgacctgctggaacattactgacgctgattgcgctaaagcgtggggagcaaacaggattagataccctggtagtcc
||||||| || || |  ||  ||||||||||| | | | ||||||| ||||||||||||||||||||||||||||||||
aaggcgaacttctaggtcaagactgacgctgagt-cacgaaagcgtagggagcaaacaggattagataccctggtagtct

acgccctaaacgatggatgctagttgttggagggcttagtctctccagtaatgcagctaacgcattaagcatcccgcctg
||||  ||||||||| |  || | ||||  |  |   ||  | |  ||||  | ||||||||  |||||  | |||||||
acgctgtaaacgatgcacacttggtgtt-aatcg-aaag-gt-t--agtaccgaagctaacgtgttaagtgtgccgcctg

gggagtacggtcgcaagattaaaactcaaaggaatagacggggacccgcacaagcggtggagcangtggtttaattcgan
||||||| | |||||||| | |||||||||||||| ||||||| |||||||||||||||||||| ||||||||||||||
gggagtatgttcgcaagaatgaaactcaaaggaattgacgggggcccgcacaagcggtggagcatgtggtttaattcgat

nnnacacgaagaaccttacctaggcttgacattgagagaat-ccgctagaaatagtggagtgtctagcttgctagacctt
   || ||| ||||||||||  ||||||||||  | || ||   | |||| |||    | |  |  | |||   |  | |
gatacgcgaggaaccttaccagggcttgacatatacaggatatagttagagata-acta-ttccccgtttg--gggtc-t

gaaaacaggtgctgcacggctgtcgtcagctcgtgtcgtgagatgttgggttaagtcccgcaacgagcgcaaccccnttt
| | |||||||||||| || ||||||||||||||| |||||| ||| ||||||||||||||||||||||||||||   ||
gtatacaggtgctgcatggttgtcgtcagctcgtgccgtgaggtgtcgggttaagtcccgcaacgagcgcaacccttgtt

cttagttgctaacaggttatgctgagaactctaaggatactgcctccg-taaggaggaggaaggtggggacgacgtcaag
 |  ||| | | || || | | || | ||||  | || ||||||   | ||||  ||||||||||| ||| ||||||||
gtctgttaccagcatgtaaagatggggactcagacgagactgccggtgataagccggaggaaggtgaggatgacgtcaaa

tcatcatggcccttacg-cctagggctacacacgtgctacaatggggtgcacaaagagaagcaatactgtgaagtggagc
||||||||||||||| | ||| |||||||||||||||||||||||  || |||||| || || | || |||| ||| |||
tcatcatggcccttatgtcct-gggctacacacgtgctacaatggcctgtacaaagcgatgcgaaacagtgatgtgaagc

caatcttca-aaa-cacctctcagttcggattgtaggctgcaactcgcctgcatgaagctggaatcgctagtaatcgcaa
 || |  || ||| ||  ||||||| | ||||| || ||| |||||| || ||||||| |||||||||||||||||| |
aaaac-gcagaaagcaggtctcagtccagattgaagtctgaaactcggcttcatgaagttggaatcgctagtaatcgtat

atcagccatgttgcggtgaatacgttcccgggtcttgtactcaccgcccgtcacaccatgggagttgtgtttgccttaag
|||||  ||| | |||||||||||||  |   ||  |        |   |     ||             |||   | |
atcag-aatgatacggtgaatacgtt--c---tc--g--------g---g-----cc-------------ttg---t-a-

tcaggatgctaaattggctactgcccacggcacacacagcgactgggg
 |   |  |   |    |  | | || | |         | ||
-c---a--c---a----c--c-g-cc-c-g--------tc-ac-----

Alignment score:    1438
% of Identity: 69.22%
Running time: 0.12s
Memory used: 2931.93 kilobytes

E:\coding\workspace\kgalign>
```

- **Runtime and Memory Comparison with galign.exe**

I randomly generated 15 pairs dataset to test the runtime and memory used in kgalign.exe, and compared with galign.exe. As the Table 1 shown, kgalign.exe and galign.exe got the same optimal score on all datasets. However, results show kgalign.exe takes less space than galign.exe, and kgalign.exe takes more time than galign.exe. Figure 1 and 2 also show the same conclusion. For the worst case, kgalign.exe takes the same space as galign.exe, and in this case, it need compute all $(n+1)(m+1)$ values. However, when the two sequences are highly similar, kgalign.exe just need to compute a small part of $(n+1)(m+1)$ values. In this case, kgalign.exe saves space and time.

Let k be the value that we guess how many difference in the two sequences. The kgalign.exe guesses k from 1, 2, 3 … until the difference in the current optimal alignment path is less than k. Thus, the runtime is $O(n) + O(2n) + \ldots + O(2^i n)$, 2^i is larger than K which is the really difference in optimal alignment of the two sequences. Therefore, the runtime is $O(2Kn)$. When n is not equal to m, we let r be the difference between the current optimal alignment of the two sequences, here k equals $r - |m - n|$.When $m > n$, the runtime is $O((m - n)n + n) + O((m - n)n + 2n) + \ldots + O((m - n)n + 2^i n)$ until $(m - n)n + 2^i$ is larger than K. In this case, the runtime is also $O(2Kn)$. When $n < m$, it's the same as $m > n$. Therefore, the time and space complexity are $O(Kn)$.

Table 1 Memory and runtime Comparison between kgalign.exe and galign.exe

Test Dataset	Lengths of the two sequences		kgalign.exe				galign.exe		
	n	*m*	*k*	*memory* [a]	*time* [b]	*score*	*memory*	*time*	*score*
1	5	10	2	0.22	0	4	0.36	0	4
2	10	9	8	0.36	0	3	0.57	0	3
3	30	25	16	2.10	0	16	4.05	0	16
4	80	80	64	16.86	0	50	32.37	0	50
5	120	110	64	29.01	0	73	66.08	0	73
6	320	20	1	25.96	0	20	33.42	0	20
7	500	50	1	91.85	0	50	126.27	0.015	50
8	1,200	1,400	1,024	4,727.86	0.219	835	8,221.53	0.062	835
9	3000	4,000	2,048	31,660.80	2.121	2,228	58,643.40	0.421	2,228
10	5,000	4,000	2,048	43,577.14	2.823	2,893	97,719.90	0.748	2,893
11	7,500	7,500	8,192	174,549.33	4.602	4,897	274,764.00	1.856	4,897
12	10,000	11,000	8,192	293,641.00	12.761	6,841	537,258.00	4.041	6,841
13	11,000	10,500	8,192	291,088.00	10.078	7,012	564,117.00	4.150	7,012
14	20,000	19,080	16,384	1,022,759.50	52.135	12,771	- [c]	-	-
15	-	-	-	-	-	-	-	-	-

[a] The unit of memory is kilobyte.
[b] The unite of time is second.
[c] "-" means the dataset cannot run.
[d] All the datasets are run with default scoring function, which is 1 for match, 0 for mismatch, 0 for indel.

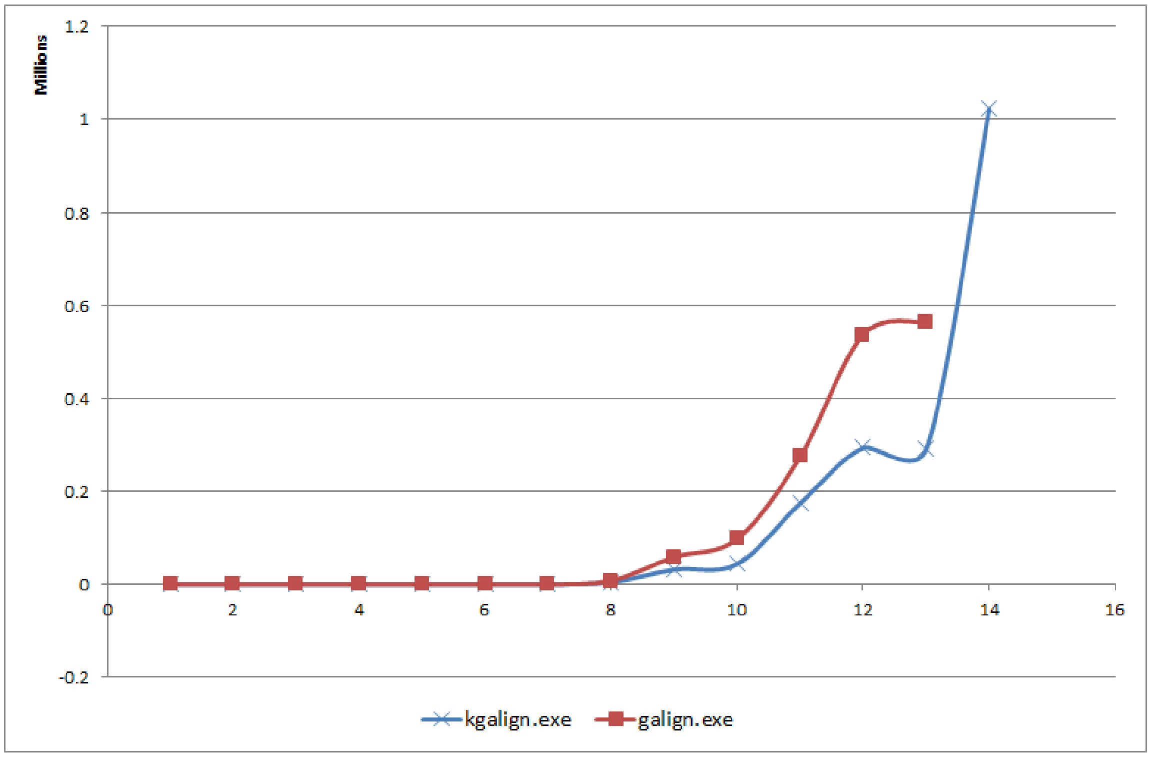


Figure 1 Memory Used Comparison between kgalign.exe and galign.exe in different datasets

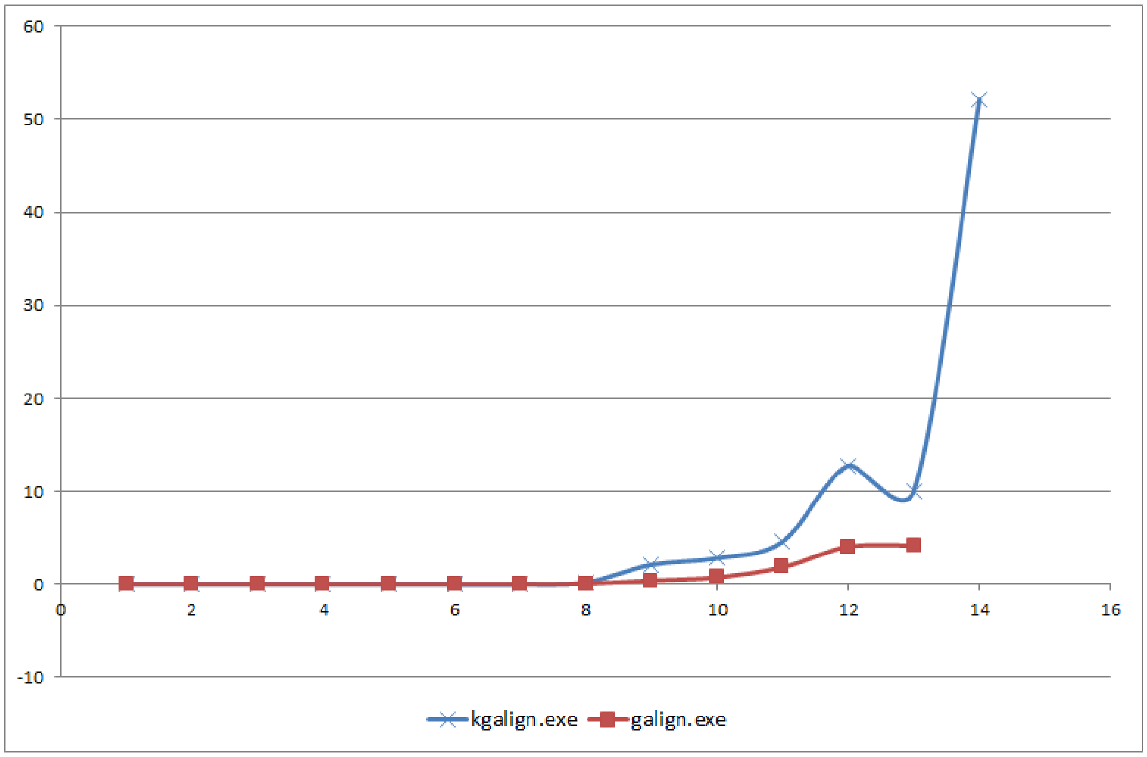
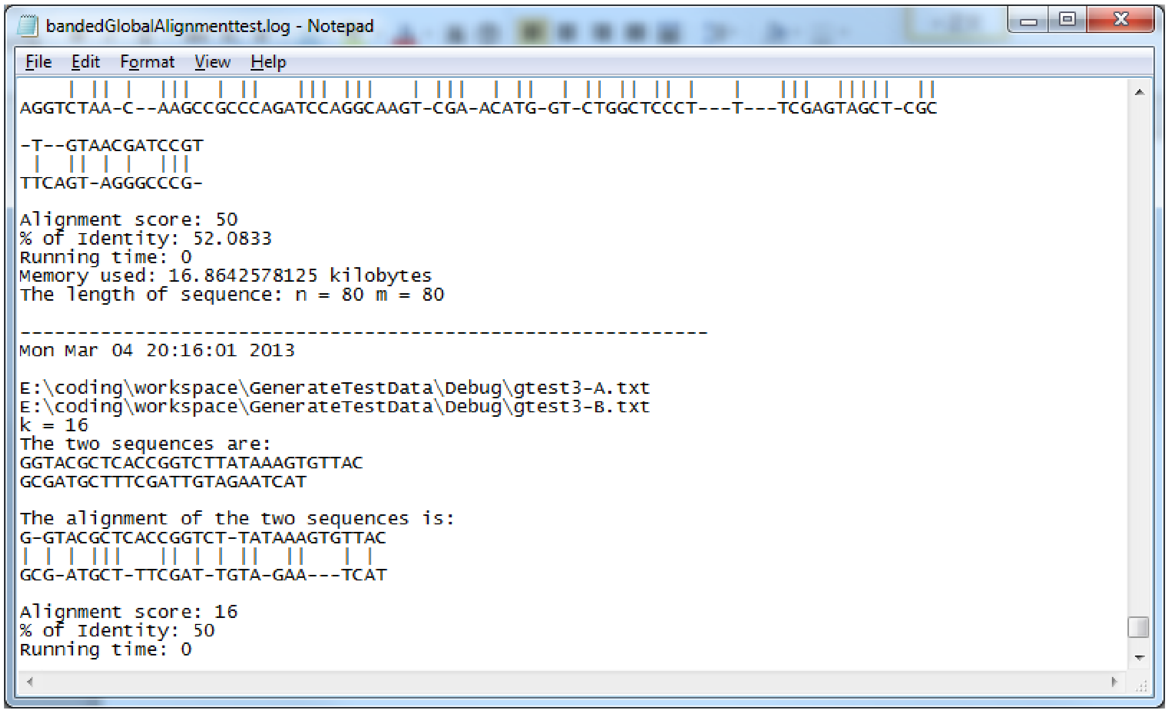


Figure 2 Running Time Comparison between kgalign.exe and galign.exe in different datasets

- ## Running log

Running log file records all run of kgalign.exe and the corresponding results.



```
    | || |  |||  | ||   ||| |||   | |||  | ||  | || || || |   |   |||  |||||  ||
AGGTCTAA-C--AAGCCGCCCAGATCCAGGCAAGT-CGA-ACATG-GT-CTGGCTCCCT---T---TCGAGTAGCT-CGC

-T--GTAACGATCCGT
 |  || | |  |||
TTCAGT-AGGGCCCG-

Alignment score: 50
% of Identity: 52.0833
Running time: 0
Memory used: 16.8642578125 kilobytes
The length of sequence: n = 80 m = 80

------------------------------------------------------------
Mon Mar 04 20:16:01 2013

E:\coding\workspace\GenerateTestData\Debug\gtest3-A.txt
E:\coding\workspace\GenerateTestData\Debug\gtest3-B.txt
k = 16
The two sequences are:
GGTACGCTCACCGGTCTTATAAAGTGTTAC
GCGATGCTTTCGATTGTAGAATCAT

The alignment of the two sequences is:
G-GTACGCTCACCGGTCT-TATAAAGTGTTAC
| | | |||   || | | ||  ||   | |
GCG-ATGCT-TTCGAT-TGTA-GAA---TCAT

Alignment score: 16
% of Identity: 50
Running time: 0
```

//Source Code

```
//Alignment.cpp
#include "Alignment.h"

int main(int argc, char *argv[])
{
	vector<int> w;	//w[0] is match, w[1] is mismatch, w[2] is indel
	w.push_back(1), w.push_back(0), w.push_back(0);
	vector<string> seq;
	if (argc < 3 || argc > 6) {
		cout << ErrorInfo << endl;
		return EXIT_FAILURE;
	}

	for (int t = 1; t < argc; t++) {
		if (argv[t][0] == '-') {
			int score = 0;
			if (argv[t][1] == 'm') {
				sscanf(argv[t], "-m%d", &score);
				w[MATCH] = score;
			} else if (argv[t][1] == 's') {
				sscanf(argv[t], "-s%d", &score);
				w[MISMATCH] = -score;
			} else if (argv[t][1] == 'i') {
				sscanf(argv[t], "-i%d", &score);
				w[INDEL] = -score;
			}
		} else {
			if (checkFilePath(argv[t]))
				seq.push_back(argv[t]);
			else {
				printf("Cannot open the file: %s.\n", argv[t]);
				return EXIT_FAILURE;
			}
		}
	}
	if (seq.size() != 2) {
		printf("Please input paths of TWO DNA sequence file!\n");
		cout << ErrorInfo << endl;
		return EXIT_FAILURE;
	}

	CBandedGlobalAlignment bgl(seq[0], seq[1], w);
	bgl.runBandedGlobalAlignment();
	return EXIT_SUCCESS;
}
//Alignment.h
#ifndef ALIGNMENT_H_
#define ALIGNMENT_H_

#include "BandedGlobalAlignment.h"

#include <vector>
#include <string>
#include <cstdio>
#include <fstream>
```

```
#include <iostream>

using namespace std;
using namespace spaceBandedGlobalAlignment;

const string ErrorInfo =
			"Please input correct parameters! For example:\n\
			>galign.exe -m2 -s1 -i2 seq1.txt seq2.txt\n\
			scoring function with +2 for match, -1 for mismatch and -2 for indels\
			to align two DNA sequences in seq1.txt and seq2.txt.\n";

int checkFilePath(const string & strPath)
{
		ifstream fin(strPath.c_str());
		int r = 0;
		if (!fin.good())
			r = 0;
		else
			r = 1;
		fin.close();
		return r;
}
#endif /* ALIGNMENT_H_ */

// BandedGlobalAlignment.cpp
#include "BandedGlobalAlignment.h"

using namespace spaceBandedGlobalAlignment;

CBandedGlobalAlignment::CBandedGlobalAlignment(const string & filePath1,
			const string & filePath2, const vector<int> & weight)
{
		filePathU = filePath1;
		filePathV = filePath2;
		w = weight;
		rU.clear();
		rV.clear();
		alignScore = 0;
		lfIdentity = 0.0;

		TimeStart = 0.0;
		TimeEnd = 0.0;
		memory = 0.0;

		m = 0;
		n = 0;
		d = 0;
}

CBandedGlobalAlignment::~CBandedGlobalAlignment()
{
		filePathU.clear();
		filePathV.clear();

		U.clear();
		V.clear();
```

```
        w.clear();

        rU.clear();
        rV.clear();

        for (int i = 0; i <= n; i++) {
                s[i].clear();
                l[i].clear();
        }
        s.clear();
        l.clear();
        L.clear();
        R.clear();

        ossGlLog.clear();
}

void CBandedGlobalAlignment::outputLog(const string & strOut)
{
        ifstream fin("bandedGlobalAlignment.log");

        ostringstream strTemp;

        if (fin.good()) {
                string strVal;
                while (!fin.eof()) {
                        getline(fin, strVal, '\n');
                        strTemp << strVal << endl;
                }
        }

        fin.close();

        ofstream fout("bandedGlobalAlignment.log");
        fout << strOut;
        fout << strTemp.str();
        fout.close();
}

void CBandedGlobalAlignment::outPutsANDl()
{
        for (int i = 0; i <= n; i++) {
                for (int j = 0; j <= m; j++) {
                        if (j < L[i] || j > R[i])
                                printf("* ");
                        else
                                printf("%d ", s[i][slCOL(i, j)]);
                }
                printf("\n");
        }

        for (int i = 0; i <= n; i++) {
                for (int j = 0; j <= m; j++) {
                        if (j < L[i] || j > R[i])
                                printf("* ");
```

```
                    else
                            printf("%c ", l[i][slCOL(i, j)]);
                }
                printf("\n");
        }
}

int CBandedGlobalAlignment::readString(const string & strPath, string & str)
{
        ifstream fin(strPath.c_str());
        if (!fin.good()) {
                printf("Cannot open the file: %s.\n", strPath.c_str());
                ossGlLog << "Cannot open the file: %s." << endl;
                fin.close();
                return 0;
        }
        bool vaild = true;
        string strTmp;
        str.clear();
        while (!fin.eof()) {
                getline(fin, strTmp, '\n');
                for (int i = 0; i < (int) strTmp.size(); i++) {
                        if (strTmp[0] == '>')
                                continue;
                        if (strTmp[i] != 'A' && strTmp[i] != 'C' && strTmp[i] != 'G'
                                        && strTmp[i] != 'T' && strTmp[i] != 'N' && strTmp[i] != 'a'
                                        && strTmp[i] != 'c' && strTmp[i] != 'g' && strTmp[i] != 't'
                                        && strTmp[i] != 'n')

                                vaild = false;
                        else
                                str += strTmp[i];
                }
        }
        if (vaild == false) {
                printf(
                                "The DNA file %s contains characters which is not a, c, g, t or n. \
                                These characters have been deleted.\n",
                                strPath.c_str());

                ossGlLog << "The DNA file " << strPath
                                << " contains characters which is not a, c, g, t or n. \
                                These characters have been deleted."
                                << endl;
        }
        fin.close();
        return 1;
}

void CBandedGlobalAlignment::stringReverse(string & str)
{
        int n = str.size();
        char c;
        for (int i = 0; i < n / 2; i++) {
                c = str[i];
                str[i] = str[n - i - 1];
```

```
		str[n - i - 1] = c;
	}
}

MatchLabel CBandedGlobalAlignment::charMatch(const char & a, const char & b)
{
	if (a == '-' && b == '-')
		return INDEL;
	else if (a == b)
		return MATCH;
	else
		return MISMATCH;
}

pair<int, char> CBandedGlobalAlignment::max(const int & s1, const int & s2,
		const int & s3)
{
	/*if two of them are equal, then there are more than one optimal path*/
	if (s1 >= s2) {
		if (s1 >= s3)
			return pair<int, char>(s1, DIAG);
		else
			return pair<int, char>(s3, LEFT);
	} else {
		if (s2 >= s3)
			return pair<int, char>(s2, UP);
		else
			return pair<int, char>(s3, LEFT);
	}
}

void CBandedGlobalAlignment::outputfastaFormat(const string & str)
{
	for (int t = 0; t < (int) str.size(); t++) {
		if (t % 80 == 0 && t != 0) {
			printf("\n");
			ossGlLog << endl;
		}
		printf("%c", str[t]);
		ossGlLog << str[t];
	}
	printf("\n");
	ossGlLog << endl;
}

void CBandedGlobalAlignment::outputResultString(const string & str,
		const int & start, const int & end)
{
	for (int i = start; i <= end; i++) {
		printf("%c", str[i]);
		ossGlLog << str[i];
	}
	printf("\n");
	ossGlLog << endl;
}
```

```
void CBandedGlobalAlignment::resultDisplay()
{
        printf("The two sequences are:\n");
        ossGlLog << "The two sequences are:\n" << endl;

        outputfastaFormat(U);
        printf("\n");
        outputfastaFormat(V);
        printf("\n");
        ossGlLog << endl;

        printf("The alignment of the two sequences is:\n");
        ossGlLog << "The alignment of the two sequences is:" << endl;
        string midline;
        int start = 0;
        int i = 0;
        for (i = 0; i < (int) rU.size(); i++) {
                if (i % 80 == 0 && i != 0) {
                        outputResultString(rU, start, i - 1);
                        outputResultString(midline, start, i - 1);
                        outputResultString(rV, start, i - 1);
                        printf("\n");
                        ossGlLog << endl;
                        start = i;
                }
                if (rU[i] == '-' || rV[i] == '-') {
                        midline.push_back(' ');
                } else if (rU[i] == rV[i]) {
                        midline.push_back('|');
                } else {
                        midline.push_back(' ');
                }
        }
        outputResultString(rU, start, i - 1);
        outputResultString(midline, start, i - 1);
        outputResultString(rV, start, i - 1);
        printf("\n");
        ossGlLog << endl;

        printf("Alignment score:  %6d\n", alignScore);
        ossGlLog << "Alignment score: " << alignScore << endl;

        printf("%% of Identity: %.2lf%%\n", (double) lfIdentity * 100);
        ossGlLog << "% of Identity: " << (double) lfIdentity * 100 << endl;

        printf("Running time: %.2lfs\n",
                        (double) (TimeEnd - TimeStart) / CLOCKS_PER_SEC );
        ossGlLog << "Running time: "
                        << (double) (TimeEnd - TimeStart) / CLOCKS_PER_SEC << endl;

        printf("Memory used: %.2lf kilobytes\n", memory / 1024);
        char chr[100];
        sprintf(chr, "%.10lf", memory / 1024);
        ossGlLog << "Memory used: " << chr << " kilobytes" << endl;
        ossGlLog << "The length of the two sequences: n = " << n << " m = " << m
                        << endl;
```

```
}

void CBandedGlobalAlignment::setLR(const int & k)
{
	if (m >= n) {
		for (int i = 0; i <= n; i++) {
			L[i] = i - k / 2 > 0 ? i - k / 2 : 0;
			R[i] = i + d + k / 2 < m ? i + d + k / 2 : m;
		}
	} else {
		for (int i = 0; i <= n; i++) {
			L[i] = i - d - k / 2 > 0 ? i - d - k / 2 : 0;
			R[i] = i + k / 2 < m ? i + k / 2 : m;
		}
	}
}

bool CBandedGlobalAlignment::bandedGlobalAlignAlgorithm(const int & k)
{
	setLR(k);
	for (int i = 0; i <= n; i++) {
		s[i].clear();
		l[i].clear();
		for (int j = 0; j < R[i] - L[i] + 1; j++) {
			s[i].push_back(0);
			l[i].push_back(DIAG);
		}
	}

	s[0][0] = 0;
	for (int j = L[0]; j <= R[0]; j++) {
		s[0][slCOL(0, j)] = j * w[INDEL];
		l[0][slCOL(0, j)] = LEFT;
	}

	for (int i = 1; i <= n; i++) {
		s[i][0] = i * w[INDEL];
		l[i][0] = UP;
		for (int j = L[i]; j <= R[i]; j++) {
			int s1 = -inf, s2 = -inf, s3 = -inf;
			int s1Col = slCOL(i - 1, j)- 1;
			int s2Col = slCOL(i - 1, j);
			int s3Col = slCOL(i, j)- 1;

			if (j - 1 >= L[i - 1] && j - 1 <= R[i - 1]) {
				s1 = s[i - 1][s1Col] + w[charMatch(U[I], V[J])];
			}
			if (j >= L[i - 1] && j <= R[i - 1]) {
				s2 = s[i - 1][s2Col] + w[INDEL];
			}
			if (j - 1 >= L[i] && j - 1 <= R[i]) {
				s3 = s[i][s3Col] + w[INDEL];
			}

			pair<int, char> charMatchResult = max(s1, s2, s3);
			s[i][slCOL(i, j)] = charMatchResult.first;
```

```
				l[i][slCOL(i, j)] = charMatchResult.second;
			}
		}

		int nDiff = 0;
		int p = n, q = m;
		rU.clear();
		rV.clear();
		while (p >= 0 && q >= 0 && (p + q != 0)) { // trace back from s[n][m] to s[0][0]
			if (l[p][slCOL(p, q)] == DIAG) {
				rU.push_back(U[P]);
				rV.push_back(V[Q]);
				if (U[P] != V[Q])
					nDiff++;
				p = p - 1;
				q = q - 1;
			} else if (l[p][slCOL(p, q)] == UP) {
				rU.push_back(U[P]);
				rV.push_back('-');
				p = p - 1;
				nDiff++;
			} else if (l[p][slCOL(p, q)] == LEFT) {
				rU.push_back('-');
				rV.push_back(V[Q]);
				q = q - 1;
				nDiff++;
			}

			if (nDiff > d + k) {
				return false;
			}

		}

		stringReverse(rU);
		stringReverse(rV);
		alignScore = s[n][slCOL(n, m)];

		int cnt = 0;
		for (int t = 0; t < (int) rU.size(); t++) {
			if (rU[t] == rV[t])
				cnt++;
		}
		lfIdentity = (double) cnt / rV.size();
		memory = sizeof(rU[0]) * rU.size() + sizeof(rV[0]) * rV.size()
				+ sizeof(U[0]) * U.size() + sizeof(V[0]) * V.size();
		for (int i = 0; i <= n; i++) {
			memory += sizeof(l[0][0])
					+ sizeof(s[0][0]) * ((double) R[i] - L[i] + 1);
		}
		return true;
}

void CBandedGlobalAlignment::runBandedGlobalAlignment()
{
		ossGlLog << "\n------------------------------------------------------------"
```

```
				<< endl;
	time_t rawtime;
	time(&rawtime);
	ossGlLog << asctime(localtime(&rawtime)) << endl;
	ossGlLog << filePathU << endl;
	ossGlLog << filePathV << endl;
	if (!readString(filePathU, U) || !readString(filePathV, V))
		return;
	n = U.size();
	m = V.size();
	d = m >= n ? m - n : n - m;
	if (n == 0 && m == 0) {
		printf(
				"The two strings are empty. Please check the DNA fasta files.\n");
		ossGlLog
				<< "The two strings are empty. Please check the DNA fasta files."
				<< endl;
		return;
	}

	L.resize(n + 1);
	R.resize(n + 1);

	s.resize(n + 1);
	l.resize(n + 1);

	TimeStart = clock();
	for (int k = 1;; k *= 2) {
		if (bandedGlobalAlignAlgorithm(k))
			break;
	}
	TimeEnd = clock();
	resultDisplay();
	outputLog(ossGlLog.str());
}

// BandedGlobalAlignment.h
/*
 * MATH 578A Homework2
 * Banded Global Alignment Algorithm
 * BandedGlobalAlignment.cpp
 *
 * Author: Haifeng Chen
 * Contact: haifengc at usc dot edu

 * Compiler:
 * (1) Ubuntu 12.10 32-bit, g++ (Ubuntu/Linaro 4.7.2-2ubuntu1) 4.7.2
 * (2) Windows 7 32-bit, MinGW GCC 4.7.2

 * Created on: Mar 3, 2013 – 2pm
 */

#ifndef BANDEDGLOBALALIGNMENT_H_
#define BANDEDGLOBALALIGNMENT_H_

#include <ctime>
```

```
#include <cmath>
#include <limits>
#include <vector>
#include <string>
#include <cstdio>
#include <fstream>
#include <cstdlib>
#include <cstring>
#include <sstream>
#include <iostream>

using namespace std;

namespace spaceBandedGlobalAlignment
{

#define I (i - 1)
#define J (j - 1)
#define P (p - 1)
#define Q (q - 1)
#define DIAG ('a')
#define UP ('b')
#define LEFT ('c')
#define slCOL(row, col) ((col) - L[(row)])
#define inf (std::numeric_limits<int>::max())

enum MatchLabel
{
	MATCH = 0, MISMATCH = 1, INDEL = 2
};

class CBandedGlobalAlignment
{
public:
	CBandedGlobalAlignment(const string & filePath1, const string & filePath2,
				const vector<int> & weigth);
	~CBandedGlobalAlignment();
	void runBandedGlobalAlignment();
private:
	string filePathU;
	string filePathV;
	string U;
	string V;
	vector<int> w;
	string rU;
	string rV;
	int alignScore;
	double lfIdentity;

	vector<vector<int> > s;
	vector<vector<char> > l;

	vector<int> L;
	vector<int> R;

	int n;
```

```
        int m;
        int d;

        clock_t TimeStart, TimeEnd;
        double memory;

        ostringstream ossGlLog;

        void outPutsANDl();
        void resultDisplay();
        void setLR(const int & k);
        void stringReverse(string & str);
        void outputLog(const string & strOut);
        void outputfastaFormat(const string & str);
        bool bandedGlobalAlignAlgorithm(const int & k);
        int readString(const string & strPath, string & str);
        MatchLabel charMatch(const char & a, const char & b);
        void outputResultString(const string & str, const int & start,
                        const int & end);
        pair<int, char> max(const int & s1, const int & s2, const int & s3);
};

}

#endif /* BANDEDGLOBALALIGNMENT_H_ */
```