

```

#include "Library.h"

template<class T, bool EQ(T,T)>
Pair<List<T>*,int> dynamic_lcs(const T *a, int alen, const T *b, int blen) {
    Pair<List<T>*,int> *table = new Pair<List<T>*,int>[(alen+1)*(blen+1)];
    Pair<List<T>*,int> *ent, *ent2, *ent3, ret;
    int i,j;
    for(i=0;i<=blen;i++) {
        table[alen*(blen+1)+i].a = 0;
        table[alen*(blen+1)+i].b = 0;
    }
    for(i=0;i<=alen;i++) {
        table[(blen+1)*i+blen].a = 0;
        table[(blen+1)*i+blen].b = 0;
    }
    for(i=alen-1;i>=0;i--) {
        for(j=blen-1;j>=0;j--) {
            ent = &table[i*(blen+1)+j];
            if(EQ(a[i],b[j])) {
                ent2 = &table[(i+1)*(blen+1)+(j+1)];
                ent->a = cons(a[i],ent2->a);
                ent->b = ent2->b + 1;
            } else {
                ent2 = &table[(i+1)*(blen+1)+(j+0)];
                ent3 = &table[(i+0)*(blen+1)+(j+1)];
                *ent = ent2->b > ent3->b ? *ent2 : *ent3;
            }
        }
    }
    ret = table[0];
    delete table;
    return ret;
}

```

Nov 08, 05 13:16

## Library.h

Page 1/2

```

#ifndef LIBRARY_H
#define LIBRARY_H

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <assert.h>
#ifndef _WIN32
#include <sys/time.h>
#else
#include <sys/timeb.h>
#endif

// A simple pair (or tuple) type
template<class T, class U> struct Pair { T a; U b; };

// A simple equality function for primitives
// (useful as a template argument)
template<class T> inline bool eq(T a, T b) { return a == b; }

// A linked list type (nil is 0)
template<class T> struct List { T car; List<T> *cdr; };

// Construct a new cons cell
//static int consCount;
template<class T> inline List<T> *cons(T car, List<T> *cdr) {
    List<T> *ret = new List<T>;
    ret->car = car;
    ret->cdr = cdr;
    //consCount++;
    return ret;
}

template <class T> bool eq(List<T> *x, List<T> *y) {
    if(!x) return !y;
    else if(!y) return false;
    else if(!eq(x->car,y->car)) return false;
    else return eq(x->cdr,y->cdr);
}

// Find the length of a linked list
template<class T> int length(List<T> *xs) {
    int len;
    for(len=0;xs;xs=xs->cdr) len++;
    return len;
}

// Return the first n elements from the list
template<class T> List<T> *take(int n, List<T> *xs) {
    List<T> *prev, *head;
    if(!xs) return 0;
    head = prev = cons<T>(xs->car,0);
    while(--n) {
        xs = xs->cdr;
        if(!xs) break;
        prev->cdr = cons<T>(xs->car,0);
        prev = prev->cdr;
    }
    return head;
}

// Concat returns a NEW list that shares b, a is not needed after a concat
template<class T> List<T> *concat(List<T> *a, List<T> *b) {
    List<T> *prev,*head;
    if(!a) return b;
    head = prev = cons<T>(a->car,0);
    for(;;) {
        a = a->cdr;
        if(!a) break;
        prev->cdr = cons<T>(a->car,0);
        prev = prev->cdr;
    }
    prev->cdr = b;
    return head;
}

```

```

}

template<class T> void deepFreeList(List<T> *xs) {
    List<T> *tmp;
    while(xs) {
        tmp = xs->cdr;
        delete xs;
        xs = tmp;
    }
}

// Generic output functions, fput(x,fp) and put(x) can be applied to any type
static inline size_t fput(char c, FILE *fp) { return fputc(c,fp); }
static inline size_t fput(const char *s, FILE *fp) { return fputs(s,fp); }
static inline size_t fput(int i, FILE *fp) { return fprintf(fp,"%d",i); }
template <class T>
size_t fput(List<T> *xs, FILE *fp) {
    for(;xs;xs=xs->cdr) if(fput(xs->car,fp)<0) return EOF;
    return 0;
}
template <class T,class U>
size_t fput(Pair<T,U> pair, FILE *fp) {
    if(putchar('(')<0) return EOF;
    if(fput(pair.a,fp)<0) return EOF;
    if(putchar(',')<0) return EOF;
    if(fput(pair.b)<0) return EOF;
    if(putchar(')')<0); return EOF;
    return 0;
}

template<class T>
size_t put(T x) {
    if(fput(x,stdout)<0) return EOF;
    if(putchar('\n')<0) return EOF;
    return 0;
}

// Time functions
#ifdef _WIN32
struct timeval { long tv_sec; long tv_usec; };
static inline int gettimeofday(struct timeval *tv, void *ignore) {
    struct _timeb buf;
    _ftime_s(&buf);
    tv->tv_sec = buf.time;
    tv->tv_usec = buf.millitm * 1000;
    return 0;
}
#endif

static inline unsigned long timeval_diff_ms(struct timeval a, struct timeval b) {
    return (a.tv_sec - b.tv_sec)*1000 + (a.tv_usec - b.tv_usec + 500) / 1000;
}

#endif

```

```

#include "Library.h"

template<class T, bool EQ(T,T),bool REVERSE>
void linear_space_lcs_b(const T *a, int alen, const T *b, int blen, int *ll) {
    int *k0 = new int[blen+1];
    int *k1 = new int[blen+1];
    int *tmp;
    int i,j;
    k1[blen] = 0;
    for(i=blen;i>=0;i--) k0[i] = 0;
    for(i=alen-1;i>=0;i--) {
        tmp = k0; k0 = k1; k1 = tmp;
        for(j=blen-1;j>=0;j--) {
            if(EQ(a[REVERSE ? alen-i-1 : i],b[REVERSE ? blen-j-1 : j]))
                k0[j] = k1[j+1] + 1;
            else
                k0[j] = k1[j] > k0[j+1] ? k1[j] : k0[j+1];
        }
    }
    for(i=blen;i>=0;i--) ll[i] = k0[i];
    delete k0;
    delete k1;
}

template<class T, bool EQ(T,T)>
Pair<List<T>*,int> linear_space_lcs(const T* a, int alen, const T* b, int blen) {
    Pair<List<T>*,int> ret = { 0, 0 },ret2,ret3;
    int *l1,*l2;
    int i,j,k,max;
    if(alen == 1) {
        for(j=blen-1;j>=0;j--) {
            if(EQ(a[0],b[j])) {
                ret.a = cons<T>(a[0],0);
                ret.b = 1;
                break;
            }
        }
    } else if(blen != 0) {
        i = alen / 2;
        l1 = new int[blen+1];
        l2 = new int[blen+1];
        linear_space_lcs_b<T,EQ,true>(a,i,b,blen,l1);
        linear_space_lcs_b<T,EQ,false>(a+i,alen-i,b,blen,l2);
        for(k=-1,max=-1,j=blen;j>=0;j--)
            if(l1[blen-j] + l2[j] > max) { max = l1[blen-j] + l2[j]; k = j; }

        ret2 = linear_space_lcs<T,EQ>(a,i,b,k);
        assert(ret2.b == l1[blen-k]);
        delete l1;

        ret3 = linear_space_lcs<T,EQ>(a+i,alen-i,b+k,blen-k);
        assert(ret3.b == l2[k]);
        delete l2;

        ret.a = concat(ret2.a,ret3.a);
        deepFreeList(ret2.a);
        ret.b = ret2.b + ret3.b;
    }
    return ret;
}

```

Oct 13, 05 23:19

## Memo.h

Page 1/1

```

#include "Library.h"

template<class T> struct MemoState {
    Pair<bool,Pair<List<T>*,int> > *table;
    const T *a, *b;
    int alen,blen;
};

template<class T, bool EQ(T,T)>
Pair<List<T>*,int> memo_lcs_internal(MemoState<T> *state, int ap, int bp) {
    Pair<bool,Pair<List<T>*,int> > *memo_ent = &state->table[ap*(state->blen+1)+bp];
    Pair<List<T>*,int> ret,ret2,ret3;
    if(memo_ent->a) return memo_ent->b;
    if(ap == state->alen || bp == state->blen) {
        ret.a = 0;
        ret.b = 0;
    } else if(EQ(state->a[ap],state->b[bp])) {
        ret2 = memo_lcs_internal<T,EQ>(state,ap+1,bp+1);
        ret.a = cons(state->a[ap],ret2.a);
        ret.b = ret2.b + 1;
    } else {
        ret2 = memo_lcs_internal<T,EQ>(state,ap+1,bp);
        ret3 = memo_lcs_internal<T,EQ>(state,ap,bp+1);
        ret = ret2.b > ret3.b ? ret2 : ret3;
    }
    memo_ent->a = true;
    memo_ent->b = ret;
    return ret;
}

template<class T, bool EQ(T,T)>
Pair<List<T>*,int> memo_lcs(const T *a, int alen, const T *b, int blen) {
    MemoState<T> state = {
        new Pair<bool,Pair<List<T>*,int> >[(alen+1)*(blen+1)],
        a, b, alen, blen
    };
    int i;
    for(i=0;i<(alen+1)*(blen+1);i++) state.table[i].a = false;
    Pair<List<T>*,int> ret = memo_lcs_internal<T,EQ>(&state, 0, 0);
    delete state.table;
    return ret;
}

```

```
#include "Library.h"

template<class T, bool EQ(T,T)>
Pair<List<T>*,int> naive_lcs(const T *a, int alen, const T *b, int blen) {
  Pair<List<T>*,int> ret,ret2,ret3;
  if(!alen || !blen) {
    ret.a = 0;
    ret.b = 0;
  } else if(EQ(a[0],b[0])) {
    ret2 = naive_lcs<T,EQ>(a+1,alen-1,b+1,blen-1);
    ret.a = cons(a[0],ret2.a);
    ret.b = ret2.b + 1;
  } else {
    ret2 = naive_lcs<T,EQ>(a+1,alen-1,b,blen);
    ret3 = naive_lcs<T,EQ>(a,alen,b+1,blen-1);
    ret = ret2.b > ret3.b ? ret2 : ret3;
  }
  return ret;
}
```

Nov 08, 05 10:00

Bench.cc

Page 1/2

```

#include <sys/time.h>
#include <sys/resource.h>

#include "Library.h"

// LCS Implementations
#include "Naive.h"
#include "Memo.h"
#include "Dynamic.h"
#include "LinearSpace.h"

// Test Data
Pair<const char *, const char *>
my_pair_1 = {
    "hello lcs are you ok?",
    "hello lcs! how are you today?"
},
my_pair_2 = {
    "hello lcs algorithm are you ok today? i'm doing pretty well myself",
    "hello LCS algorithm how are you today? i am doing pretty bad myself"
};

extern Pair<const char *,const char *>
test_pair_20, test_pair_100, test_pair_500,
test_pair_1000, test_pair_5000,
test_pair_10000, test_pair_20000, test_pair_40000,
test_binary_pair_20, test_binary_pair_100, test_binary_pair_500,
test_binary_pair_1000, test_binary_pair_5000,
test_binary_pair_10000, test_binary_pair_20000, test_binary_pair_40000;

Pair<const char *, const char *> *examples[] = {
    &my_pair_1, &my_pair_2,
    &test_pair_20, &test_binary_pair_20,
    &test_pair_100, &test_binary_pair_100,
    &test_pair_500, &test_binary_pair_500,
    &test_pair_1000, &test_binary_pair_1000,
    &test_pair_5000, &test_binary_pair_5000,
    &test_pair_10000, &test_binary_pair_10000,
    &test_pair_20000, &test_binary_pair_20000,
    &test_pair_40000, &test_binary_pair_40000
};

static struct {
    Pair<List<char>*,int> (*f)(const char *a, int alen, const char *b, int blen);
    const char *name;
    bool linear_space;
} algorithms[] = {
    { naive_lcs<char,eq>, "naive_lcs", false },
    { memo_lcs<char,eq>, "memo_lcs", false },
    { dynamic_lcs<char,eq>, "dynamic_lcs", false },
    { linear_space_lcs<char,eq>, "linear_space_lcs", true }
};

// Test function
int main(int argc, char **argv) {
    Pair<List<char>*,int> ret;
    unsigned int i,j;
    struct timeval start,end;
    struct rusage rusage_start, rusage_end;
    unsigned long wall_diff,user_diff,sys_diff;
    int alen, blen;
    const char *a;
    bool quick = argc > 1 && strcmp(argv[1],"quick")==0;
    bool veryquick = argc > 1 && strcmp(argv[1],"veryquick")==0;
    bool skip[sizeof(examples)/sizeof(examples[0])] = { false };
    List<char> *firstResult;
    int reqExample;
    const char *reqAlg;
    int reqLength;
    reqAlg = (argc > 1 && !quick && !veryquick) ? argv[1] : NULL;
    reqExample = argc > 2 ? strtol(argv[2],NULL,10) : -1;
    reqLength = argc > 3 ? strtol(argv[3],NULL,10) : -1;
    for(i=0;i<sizeof(examples)/sizeof(examples[0]);i++) {
        if(reqExample != -1 && (int)i != reqExample) continue;

```

```

a = examples[i]->a;
alen = strlen(examples[i]->a);
blen = strlen(examples[i]->b);
if(reqLength != -1) {
    alen = alen > reqLength ? reqLength : alen;
    blen = blen > reqLength ? reqLength : blen;
}
firstResult = 0;
printf("Running example: %d(%d/%d,%s)\n",i,alen,blen,a[0]!='0' || a[0]!='1'?"{0,1}":"{A,C,G,T}");
for(j=0;j<sizeof(algorithms)/sizeof(algorithms[0]);j++) {
    if(reqAlg && strcmp(algorithms[j].name,reqAlg)!=0) continue;
    if(skip[j]) continue;
    if(!algorithms[j].linear_space) {
        if(quick && (alen > 1500 || blen > 1500)) continue;
        if(alen > 15000 || blen > 15000) {
            printf("WARNING: skipping %s because input is too large\n", algorithms[j].name);
            continue;
        }
    }
    printf("Runing algorithm: %s\n",algorithms[j].name);

    getrusage(RUSAGE_SELF,&rusage_start);
    gettimeofday(&start,NULL);
    ret = algorithms[j].f(examples[i]->a,alen,examples[i]->b,blen);
    gettimeofday(&end,NULL);
    getrusage(RUSAGE_SELF,&rusage_end);

    wall_diff = timeval_diff_ms(end,start);
    user_diff = timeval_diff_ms(rusage_end.ru_utime,rusage_start.ru_utime);
    sys_diff = timeval_diff_ms(rusage_end.ru_stime,rusage_start.ru_stime);

    printf("LCS Length: %d\n",ret.b);
    printf("LCS String: ");
    put(take(72,ret.a));
    printf("Wallclock Time (ms): %li\n",wall_diff);
    printf("CPU Time (ms): %li\n",user_diff+sys_diff);
    printf("User Time (ms): %li\n", user_diff);
    printf("System Time (ms): %li\n", sys_diff);
    //printf("Cons cell count: %d\n",consCount); consCount=0;
    if(wall_diff > (unsigned long) 10*alen) {
        printf("WARNING: %s is too slow, not running again\n",algorithms[j].name);
        skip[j] = true;
    }
    if(firstResult) {
        if(!eq(ret.a,firstResult)) {
            fprintf(stderr,"ERROR: Different algorithms returned different results\n");
            exit(1);
        }
    }
    else {
        firstResult = ret.a;
    }
    printf("=====\n");
}
printf("\n");
if(veryquick) break;
}
return 0;
}

```

Nov 08, 05 13:16

makedata.pl

Page 1/1

```
#!/usr/bin/perl -w

use strict;

my @alphas = ([qw(A C T G)],[qw(0 1)]);
my @lengths = qw(20 100 500 1000 5000 10000 20000 40000);

sub makeseq {
    my ($alpha,$len) = @_;
    my $buf = '';
    while($len--){
        $buf .= $alpha->[rand() * scalar(@{$alpha})];
    }
    $buf;
}

print "// THIS FILE IS AUTO-GENERATED! DO NOT EDIT\n";
print "#include \"Library.h\"\n";

foreach my $len (@lengths) {
    foreach my $alpha (@alphas) {
        my $name = $alpha->[0] =~ /^[01]$/ ? "test_binary" : "test";
        print "Pair<const char *, const char *> ${name}_pair_$len = {\n";
        foreach my $done (0,1) {
            my $s = makeseq($alpha,$len + int (rand() * $len / 3 - $len/6));
            print " //Sequence of " . length($s) . " chars";
            while($s =~ s/^(.{1,72})//) {
                print "\n \"$1\"";
            }
            print ", " unless ($done);
            print "\n";
        }
        print "};\n";
    }
}
}
```

Nov 08, 05 13:15

## GNUmakefile

Page 1/1

```

CXXFLAGS = -Wall -Werror -O2 -g
A2PS = a2ps
A2PSOPTS = \
    --header="Brian Alliet" --left-footer= --right-footer="%Q" \
    -l -M letter -l -l 100

HEADERS = $(wildcard *.h)
SCRIPTS = $(wildcard *.pl)
UNITS = Bench.cc

all: Bench

TestData.cc: makedata.pl
    perl ./${< > ${@

Bench: Bench.o TestData.o
Bench.o: Bench.cc $(HEADERS)

%.o: %.cc Library.h
    $(CXX) $(CXXFLAGS) -c -o ${@ ${<

%: %.o
    $(CXX) $(LD_FLAGS) -o ${@ ${^

clean:
    rm -f *.o Bench

test: Bench
    ./${<

veryquicktest: Bench
    ./${< veryquick
quicktest: Bench
    ./${< quick

rebuild-results:
    rm -f results.txt
    $(MAKE) results.txt

results.txt:
    $(MAKE) all
    ./Bench > results.txt

doc.ps: $(HEADERS) $(UNITS) $(SCRIPTS) GNUmakefile
    $(A2PS) $(A2PSOPTS) -o ${@ ${^

%.pdf: %.ps
    ps2pdf ${<

pdf: doc.pdf
    @if [ "$(uname -s)" = "Darwin" ]; then open ${^; else xpdf ${^; fi

push: doc.pdf
    darcs push charger.brianweb.net:/home/darcs/cs800
    scp doc.pdf charger.brianweb.net:/home/darcs/cs800

```