Contents

1 Shell Script

```
#!/bin/bash
clear
g++ $1.cpp -DDBG -o $1
if [[ "$?" == "0" ]]; then
    echo Running
    ./$1 <$1.in> $1.out
    echo END
fi
```

2 Libraries

2.1 cstdlib

```
#include <cstdlib>
using namespace std;
{
// Function: String conversion
    double atof(const char* str);
        // char* 轉 double
    int atoi(const char * str);
        // char* 轉 int
    long int atol(const char * str);
        // char* 轉 long int
    long long int atoll(const char * str);
        // char* 轉 long long int
    double strtod(const char* str, char** endptr);
        // char* 轉 double;
    float strtof(const char* str, char** endptr);
        // char* 轉 float
    long int strtol(const char* str, char** endptr,
        int base);
        // char*(base) 轉 long int
            且指向轉換子字串之後
    long double strtold(const char* str, char**
        endptr);
        // char*(base) 轉 long double
            且指向轉換子字串之後
    long long int strtoll(const char* str, char**
        endptr, int base);
        // char*(base) 轉 long long int
            且指向轉換子字串之後
    unsigned long int strtoul(const char* str, char**
        endptr, int base);
        // char*(base) 轉 unsigned long int
            且指向轉換子字串之後
    unsigned long long int strtoull(const char* str,
        char** endptr, int base);
        // char*(base) 轉 unsigned long long int
            且指向轉換子字串之後
// Function: Integer arithmetics
    int abs(int n);
    long int llabs(long int n);
    long long int llabs(long long int n);
        // Absolute value
}
```

2.2 algorithm

```
#include <algorithm>
using namespace std;
{
    // FI(ForwaradIterator)
    // RAI(RandomAccessIterator)
    // BI(BidirectionalIterator)
    void sort(RAI first, RAI last);

    FI lower_bound(FI first, FI last, const T& k);
    /* 最左邊 ≥ k 的位置 */

    FI upper_bound(FI first, FI last, const T& k);
    /* 最左邊 > k 的位置 */

    pair<FI,FI> equal_range(FI first, FI last, const
        T& k);
    /* 等於 k 的範圍 [lower_bound, upper_bound) */

    bool next_permutation(BI first, BI last);
    /* 使用已經排序(由小到大)的資料，產生下一組排列 */

    bool prev_permutation(BI first, BI last);
    /* 針對逆向排序(由大到小)的資料，產生上一組排序 */
}
```

2.3 map

```
#include <map>
using namespace std;
{
/* Associative containers that store elements by a
    combination
 *  of a key value and a mapped value, in a specific
    order
 *  associated with key value.
 * [Key values] are used to sort and uniquely
    identify elements
 * [Mapped values] store the content associated to
    this key.
 */

// Constructor
    map<char,int> mp1;   // empty
    map<char,int> mp2 (mp1.begin(),mp1.end());  //
        range
    map<char,int> mp3 (mp1);    // copy
// Operator
    mp1['a'] = 3;
    mp1['b'] = 1;
    mp1['c'] = 2;
        // [] Access element by reference or insert
            new element if not found
        // =  Assign new content by replacing
// Iterator
    iterator begin();  // Return iterator to beginning
    iterator end();    // Return iterator to end
    iterator rbegin(); // Return reverse iterator to
        reverse beginning
    iterator rend();   // Return reverse iterator to
        reverse end
// Capacity
    bool empty();     // test if empty
    size_type size(); // return size
    size_type max_size();       // return maximum size
```

```
// Element access
    ['a']    // operator []
    at('a');// by reference / const
// Modifiers
    // Insert element
        pair<map::iterator,bool> insert(value_type&
            val);
            // value_type eg. for mp1 is
                pair<char,int>('x', 10)
        iterator insert(mp1.begin(),mp1.find('c'));
            // range
    // Clear content
        void clear();
    // Erase element
        void erase(iterator k); // by iterator
        size_type erase(const key_type& k); // by key
            // eg. mp1.erase('x') return erased
                element num
        void erase(iterator first, iterator last); //
            by range
    // Swap content of 2 same type map
        void swap (map& x);
// Operations
    // Find element by key, end() if none
        iterator find(const key_type& k);
    // Count elements with a specific key (max_val =
        1)
        size_type count (const key_type& k) const;
    // Iterator to lower bound
        iterator lower_bound(const key_type& k);
    // Iterator to upper bound
        iterator upper_bound(const key_type& k);
    // Get range of equal elements
        pair<iterator,iterator> equal_range(const
            key_type& k);
}
```

2.4 set

```
#include <set>
using namespace std;
{
/* Containers that store unique elements following a
    specific order
 * The value of an element each must be unique(like a
    key)
 * The value of the elements cannot be modified, but
    can be inserted or removed
 * Elements are sorted in order by their key
 */

// Constructor
    set<int> first;
        // empty
    int myints[]= {10,20,30,40,50};
    set<int> second (myints,myints+5);
        // range
    set<int> third (second);
        // copy
// Operator
    set& operator = (const set& x); // copy
// Iterators
    begin(); // Return iterator to beginning
    end(); // Return iterator to end
    rbegin(); // Return reverse iterator to reverse
        beginning
    rend(); // Return reverse iterator to reverse end
// Capacity:
    bool empty();
        //Test whether container is empty
    size_type size();
        // Return container size
    size_type max_size();
        // Return maximum size
// Modifiers:
    pair<iterator,bool> insert(val);
        // Insert element true if inserted false if
            existed
    void/size_type/void erase(val/iterator/range);
        // Erase elements
    void swap (set& x);
        // Swap content
    void clear();
        // Clear content
// Operations
    iterator find(val) const;
        // Get iterator to element, end if none
    size_type count(val) const;
        // Count elements with a specific value
    iterator lower_bound(val) const;
        // Return iterator to lower bound
    iterator upper_bound(val) const;
        // Return iterator to upper bound
    pair<iterator,iterator> equal_range(val) const;
        // Get range of equal elements
}
```

2.5 vector

```
#include <vector>
using namespace std;
{
/* Sequence containers that can change in size.
 * Like arrays their elements can be accessed using
     offsets on regular pointers to its elements
 *  but consume more memory in exchange for grow
     dynamically in an efficient way.
 */
// Constructor
    vector<int> first; // empty
    vector<int> second (4,100); // four ints with
        value 100
    vector<int> third (second.begin(),second.end());
        // range
    vector<int> fourth (third); // copy
// Iterators:
    begin();
    end();
    rbegin();
    rend();
// Capacity:
    size();
    max_size();
    resize(size_type n);
    resize(size_type n, value_type val);
        // resize vector with first n elements
    bool empty();
//Element access:
    myvector[index]; //operator[]
        //Access element
    at(index); // by reference
        //Access element
    front();
        /Access first element
    back();
        // Access last element
//Modifiers:
    assign(range, fill);    // fill"n, val"
        // Assign/replace new content
    push_back(val);
        //Add element at the end
    pop_back();
        //Delete last element
    iterator insert(iterator, val);
    insert(iterator, fill);
    insert(iterator, range);
        //Insert elements
    iterator erase(iterator);
    iterator erase(range);
        //Erase elements
    swap(vector<>& x);
        //Swap content
```

```
    clear();
        //Clear content
}
```

2.6 string

```
#include <string>
using namespace std;
{
/*Strings are objects that represent sequences of
    characters.*/
// Not Member Functions
    //Convert from strings
        int stoi(const string& str, size_t* idx=0,
            int base=10);
            // string起idx 轉 int(base)
                且idx標示轉換子字串之後
        long stol(const string& str, size_t* idx=0,
            int base=10);
            // string起idx 轉 long int(base)
                且idx標示轉換子字串之後
        unsigned long stoul(const string& str,
            size_t* idx=0, int base=10);
            // string起idx 轉 unsinged int(base)
                且idx標示轉換子字串之後
        long long stoll(const string& str, size_t*
            idx=0, int base=10);
            // string起idx 轉 long long(base)
                且idx標示轉換子字串之後
        unsigned long long stoull (const string& str,
            size_t* idx=0, int base=10);
            // string起idx 轉 unsinged long
                long(base) 且idx標示轉換子字串之後
        float stof(const string& str, size_t* idx=0);
            // string起idx 轉 float
                且idx標示轉換子字串之後
        double stod(const string&  str, size_t*
            idx=0);
            // string起idx 轉 double
                且idx標示轉換子字串之後
        long double stold (const string& str, size_t*
            idx=0);
            // string起idx 轉 long double
                且idx標示轉換子字串之後

    //Convert to strings
    string to_string (int val);
    string to_string (long val);
    string to_string (long long val);
    string to_string (unsigned val);
    string to_string (unsigned long val);
    string to_string (unsigned long long val);
    string to_string (float val);
    string to_string (double val);
    string to_string (long double val);
        //Convert numerical value to string
// Member Function
    //operator=
        //String assignment
//Iterators:
    begin();
    end();
    rbegin();
    rend();
//Capacity:
    size();
    length();
        //Return length of string, in terms of bytes
    max_size();
        //Return maximum size of string
    resize(n);
    resize(n, val);
        //Resize string
    clear();
        //Clear string
    empty();
        //Test if string is empty
    shrink_to_fit();
        //Shrink to fit

//Element access:
    //char operator[]
        //Get character of string
    char at(index);
        //Get character in string
    char back();
        //Access last character
    char front();
        //Access first character

//Modifiers:
    // string operator +=
    append (const string& str); // string
    append (const string& str, size_t subpos, size_t
        sublen);//substring
    append (const char* s);// char*
    append (const char* s, size_t n);//buffer(first n
        char)
    append (size_t n, char c);//fill
    append (InputIterator first, InputIterator last);
        //range
        // Append to string
    push_back(char);
        //Append character to string
    string (1)
    assign (string);//string
    assign (string, size_t subpos, size_t
        sublen);//substring
    assign (const char* s);//c-string
    assign (const char* s, size_t n);//buffer(first n
        char)
    assign (size_t n, char c);//fill
    assign (InputIterator first, InputIterator
        last);//range
        //Assign new content to string
    insert (size_t pos, const string& str);//string
    insert (size_t pos, const string& str, size_t
        subpos, size_t sublen);//substring
    insert (size_t pos, const char* s);//char*
    insert (size_t pos, const char* s, size_t
        n);//buffer
    insert (size_t pos, size_t n, char c);//fill
    void insert (iterator p, size_t n, char c);//fill
    iterator insert (iterator p, char c);//char
    void insert (iterator p, InputIterator first,
        InputIterator last);//range
        //Insert into string before posision(pos)
    erase(size_t pos = 0, size_t len = npos);
    iterator erase (iterator p);
    iterator erase (iterator first, iterator
        last);//range
        //Erase characters from string
    replace (size_t pos,  size_t len,  const string&
        str);//string
    replace (iterator i1, iterator i2, const string&
        str);//string
    replace (size_t pos,  size_t len,  const string&
        str,
                size_t subpos, size_t
                    sublen);//substring
    replace (size_t pos,  size_t len,  const char*
        s);//char*
    replace (iterator i1, iterator i2, const char*
        s);//char*
    replace (size_t pos,  size_t len,  const char* s,
        size_t n);//buffer
    replace (iterator i1, iterator i2, const char* s,
        size_t n);//buffer
    replace (size_t pos,  size_t len,  size_t n, char
        c);//fill
```

```
110     replace (iterator i1, iterator i2, size_t n, char
            c);//fill
111     replace (iterator i1, iterator i2, range);//range
112         //Replaces the portion of the string that
                begins at character pos and spans len
                characters
113         //  or [i1,i2)
114     swap(string& str);
115         //Swap string values
116     pop_back();
117         //Delete last character
118
119 //String operations:
120     const char* c_str() const;
121         //Get C string equivalent (public member
                function )
122     const char* data() const;
123         //Get string data (public member function )
124     size_t copy (char* s, size_t len, size_t pos = 0)
            const;
125         //Copy sequence of characters from string
                (public member function )
126 find
127 //Find content in string (public member function )
128 rfind
129 //Find last occurrence of content in string (public
        member function )
130     string substr (size_t pos = 0, size_t len = npos)
            const;
131         //Generate substring that starts at character
                position pos and spans len characters
132         // (or until the end of the string, whichever
                comes first). (public member function )
133     int compare (string) const;//string
134     int compare (size_t pos, size_t len, string)
            const;//substring
135     int compare (size_t pos, size_t len, string,
            size_t subpos, size_t sublen)
            const;//substring
136     int compare (const char* s) const;//char*
137     int compare (size_t pos, size_t len, const char*
            s) const;//char*
138     int compare (size_t pos, size_t len, const char*
            s, size_t n) const;//buffer
139         //Compare strings (public member function )
140
141 //Member constants
142     npos //Maximum value for size_t
143
144 // Non-member function overloads
145     operator +
146         //Concatenate strings (function )
147     relational operators == != >= > <= <
148         //Relational operators for string (function )
149     swap(str1, str2);
150         //Exchanges the values of two strings
                (function )
151     operator>>
152         //Extract string from stream (function )
153     operator<<
154         //Insert string into stream (function )
155     getline(istream, string);
156         //Get line from stream into string (function )
157 }
```

3 Algorithms

3.1 最短路

3.1.1 Bellman-Ford

3.1.2 Dijkstra’s

3.2 LIS - Longest Increasing Subsequence

4 Formula

4.1 thm

- 中文測試
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$