

Contents

1	Data Structures	2
1.1	pair	2
1.2	map	2
1.3	set	2
1.4	queue	2
1.5	deque	2
1.6	priority queue	2
1.7	stack	2
1.8	vector	2
1.9	list	2
1.10	string	2
1.11	heap	2
1.12	valarray	2
2	Numerical	3
2.1	Number Theory	3
2.2	Combinatorics	4
3	Graph	5
3.1	Connected components	5
3.2	Flood fill	5
3.3	Topological sorting	5
3.4	Minimum Spanning Tree	5
3.5	Shortest Path	7
3.6	Transitive Closure	7
3.7	Matching	7
3.8	Euler Cycle and Chinese Postman	7
3.9	Edge and Vertex Connectivity	7
3.10	Network Flow	7
3.11	Planarity detection	7
4	Geometry	8
4.1	Geometric primitives	8
4.2	Intersection detection	8
4.3	Convex Hull	8
4.4	Nearest Neighbour	8
4.5	Triangulation	8
4.6	Range search	8

4.7	Voronoi diagram	8
4.8	Point location	8
4.9	Polygon partitioning	8
4.10	Maintaining line arrangements	8
4.11	Minkowski sum	8
5	Pattern	9
5.1	String Matching	9
5.2	Longest common subsequence	9
5.3	Shortest common superstring	9
6	Hard problems	10
7	Input/Output	11
7.1	simple/for/while solve	11
8	Idioms	12
8.1	Calendrical Calculations	12
8.2	Timetable search	12
9	Misc	13
9.1	Knapsack	13
9.2	Bin packing	13
9.3	13

Chapter 1

Data Structures

pair	2
map	2
set	2
queue	2
deque	2
priority queue	2
stack	2
vector	2
list	2
string	2
heap	2
valarray	2

- 1.1 pair
- 1.2 map
- 1.3 set
- 1.4 queue
- 1.5 deque
- 1.6 priority queue
- 1.7 stack
- 1.8 vector
- 1.9 list
- 1.10 string
- 1.11 heap
- 1.12 valarray

Chapter 2

Numerical

Number Theory	3
gcd	3
euclid	3
Combinatorics	4

2.1 Number Theory

Miscellaneous algorithms in number theory.

2.1.1 GCD

Usage `d = gcd(a, b);` $a > b$

Complexity $O(\log(b))$

Characteristics

Example

```
gcd( 10, 6 ) == 2
```

Valladolid 202

2.1.2 Euclid

Usage `d = euclid(a, b, x, y);` $a > b$, x and y are return values that satisfy $ax+by = d$.
sadsad

```
asdsadasasdsadas asdsadasasdsadas asdsadas asdsadas asdsadas asdsadas asdsadas dfd-  
fasdsadasasdsadas asdsadasasdsadas asdsadas
```

Complexity $O(\log(b))$

Characteristics x and y have (hopefully, probably, don't know...) the smallest absolute value

Example

```
euclid( 10, 6, x, y ) == 2, (x,y)==(-1,2)
```

Valladolid 202

Listing 2.1: gcd.cpp — 4a431429

```
int gcd( int a, int b ) {  
    if( b==0 )  
        return a;  
    else  
        return gcd( b, a%b );  
}
```

Listing 2.2: euclid.cpp — ee83259c

```
int euclid( int a, int b, int &x, int &y ) {  
    if( b==0 ) {  
        x = 1;  
        y = 0;  
        return a;  
    } else {  
        int d = euclid( b, a%b, y, x );  
        y -= a/b*x;  
        return d;  
    }  
}
```

2.2 Combinatorics

Combinatorics is fundamental...

Chapter 3

Graph

Connected components	5
Flood fill	5
Topological sorting	5
Minimum Spanning Tree	5
kruskal	5
Shortest Path	7
Transitive Closure	7
Matching	7
Euler Cycle and Chinese Postman	7
Edge and Vertex Connectivity	7
Network Flow	7
Planarity detection	7

3.1 Connected components

3.2 Flood fill

3.3 Topological sorting

3.4 Minimum Spanning Tree

MST is a nice problem.

3.4.1 Kruskal

Usage `kruskal(graph, tree, n);`

Complexity $O(E \log E)$

Characteristics Kruskal is faster when dealing with...

Example

```
vector< vector<pair<int,double> > > edges;  
  
edges.resize( 100 );
```

```
kruskal( edges, edges, 100 );
```

Valladolid 10147

Listing 3.1: sets.cpp — b654c18a

```
#include <vector>

class sets {
    struct set_elem {
        int head, rank; // rank is a pseudo-height with height<=rank
        set_elem( int anElem ) : head(anElem), rank(0) {}
    };
    vector< set_elem > elems;

    int get_head( int i ) { // Find head of set with path-compression
        if ( i != elems[i].head )
            elems[i].head = get_head( elems[i].head );
        return elems[i].head;
    }

public:
    sets( int nElems ) {
        elems.reserve(nElems);
        for( int i=0; i<nElems; i++ )
            elems.push_back( set_elem(i) );
    }

    bool equal( int a, int b ) {
        return ( get_head(a) == get_head(b) );
    }

    void link( int a, int b ) { // union sets
        a = get_head(a);
        b = get_head(b);

        if( elems[a].rank > elems[b].rank )
            elems[b].head = a;
        else {
            elems[a].head = b;
            if( elems[a].rank == elems[b].rank )
                elems[b].rank++;
        }
    };
};
```

```
if( i < (*iter).first ) // Undirected: only use half of the edges
    edges.push_back( make_pair( (*iter).second,
                                make_pair(i,(*iter).first)) );
}

// Clear tree
for( int i=0; i<n; i++ )
    tree[i].clear();

sort( edges.begin(), edges.end() );

// Add edges in order of non-decreasing weight
int numEdges = edges.size();
for( int i=0; i<numEdges; i++ ) {
    pair<int,int> &edge = edges[i].second;

    // Add edge if the edge-endpoints aren't in the same set
    if( !sets.equal(edge.first, edge.second) ) {
        sets.link( edge.first, edge.second );
        tree[edge.first].push_back( make_pair( edge.second, edges[i].first ) );
        tree[edge.second].push_back( make_pair( edge.first, edges[i].first ) );
    }
}
}
```

Listing 3.2: kruskal.cpp — 826d118b

```
#include <algorithm>
#include <vector>
#include "1_sets.cpp"

template<class V>
void kruskal( const V &graph, V &tree, int n ) {
    typedef typename V::value_type E;
    typedef typename E::const_iterator E_iter;
    typedef typename E::value_type::second_type D;

    sets sets(n);
    vector< pair< D, pair<int,int> > > edges;

    // Convert all edges into a single edge-list
    for( int i=0; i<n; i++ ) {
        for( E_iter iter=graph[i].begin(); iter!=graph[i].end(); iter++ ) {
```

3.5 Shortest Path

Shortest path is a nice easy problem.

3.6 Transitive Closure

3.7 Matching

3.8 Euler Cycle and Chinese Postman

3.9 Edge and Vertex Connectivity

3.10 Network Flow

3.11 Planarity detection

Chapter 4

Geometry

Geometric primitives	8
Intersection detection	8
Convex Hull	8
Nearest Neighbour	8
Triangulation	8
Range search	8
Voronoi diagram	8
Point location	8
Polygon partitioning	8
Maintaining line arrangements	8
Minkowski sum	8

- 4.1 Geometric primitives
- 4.2 Intersection detection
- 4.3 Convex Hull
- 4.4 Nearest Neighbour
- 4.5 Triangulation
- 4.6 Range search
- 4.7 Voronoi diagram
- 4.8 Point location
- 4.9 Polygon partitioning
- 4.10 Maintaining line arrangements
- 4.11 Minkowski sum

Chapter 5

Pattern

String Matching	9
Longest common subsequence	9
Shortest common superstring	9

5.1 String Matching

5.2 Longest common subsequence

5.3 Shortest common superstring

Chapter 6

Hard problems

Chapter 7

Input/Output

simple/for/while solve	11
----------------------------------	----

7.1 simple/for/while solve

Chapter 8

Idioms

Calendrical Calculations	12
Timetable search	12

8.1 Calendrical Calculations

8.2 Timetable search

Chapter 9

Misc

Knapsack	13
Bin packing	13
.	13

9.1 Knapsack

9.2 Bin packing

9.3

Listings

2.1	gcd	4
2.2	euclid	4
3.1	sets	6
3.2	kruskal	6