Contents

1 Contest Setup

1.1 vimrc

```
set number        " Show line numbers
set mouse=a       " Enable inaction via mouse
set showmatch        " Highlight matching brace
set cursorline       " Show underline
set cursorcolumn     " highlight vertical column

filetype on "enable file detection
syntax on    "syntax highlight

set autoindent       " Auto-indent new lines
set shiftwidth=4     " Number of auto-indent spaces
set smartindent      " Enable smart-indent
set smarttab         " Enable smart-tabs
set tabstop=4    " Number of spaces per Tab

" ---------Optional----------

set undolevels=10000     " Number of undo levels
set scrolloff=5      " Auto scroll

set hlsearch     " Highlight all search results
set smartcase    " Enable smart-case search
set ignorecase   " Always case-insensitive
set incsearch    " Searches for strings incrementally

highlight Comment ctermfg=cyan
set showmode

set encoding=utf-8
set fileencoding=utf-8
scriptencoding=utf-8
```

1.2 bashrc

```
alias g++="g++ -Wall -Wextra -std=c++11 -O2"
```

1.3 Grep Error and Warnings

```
g++ main.cpp 2>&1 | grep -E 'warning|error'
```

1.4 C++ template

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef long long int ll;
typedef pair<int, int> ii;

int main()
{
    return 0;
}
```

1.5 Java template

```java
import java.io.*;
import java.util.*;

public class Main
{
    public static void main(String[] args)
    {
        MyScanner sc = new MyScanner();
        out = new PrintWriter(new BufferedOutputStream(System.out));
        // Start writing your solution here.

        // Stop writing your solution here.
        out.close();
    }

    public static PrintWriter out;

    public static class MyScanner
    {
        BufferedReader br;
        StringTokenizer st;

        public MyScanner()
        {
            br = new BufferedReader(new InputStreamReader(System.in));
        }

        boolean hasNext()
        {
            while (st == null || !st.hasMoreElements()) {
                try {
                    st = new StringTokenizer(br.readLine());
                } catch (Exception e) {
                    return false;
                }
            }
            return true;
        }

        String next()
        {
            if (hasNext())
                return st.nextToken();
            return null;
        }

        int nextInt()
        {
            return Integer.parseInt(next());
        }

        long nextLong()
        {
            return Long.parseLong(next());
        }

        double nextDouble()
        {
            return Double.parseDouble(next());
        }

        String nextLine()
        {
            String str = "";
            try {
                str = br.readLine();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return str;
        }
    }
}
```

1.5.1 Java Issues

1. Random Shuffle before sorting: $Random\ rnd = new\ Random();\ rnd.nextInt();$
2. Use StringBuilder for large output
3. Java has strict parsing rules. e.g. using sc.nextInt() to read a long will result in RE
4. For class sorting, use code: implements Comparable<Class name>. Or, use code: new Comparator<Interval>() {} at Collections.sort() second argument

2 System Testing

1. Setup bashrc and vimrc
2. Install Java 8, Eclipse 32-bit, g++ compiler
3. Remove Chinese input method
4. Look for compilation parameter and code it into bashrc
5. Test if c++ and java templates work properly on local and judge machine
6. Test "divide by 0" → RE/TLE?
7. Make a complete graph and run Floyd warshall, to test time complexity upper bound

8. Make a linear graph and use DFS to test stack size
9. Print output with extra newline and spaces

3 Reminder

1. 隊友的建議, 要認真聽! 通常隊友的建議都會突破你盲點
2. Read the problem statements carefully. Input and output specifications and constraints are crucial!
3. Estimate the **time complexity** and **memory complexity** carefully.
4. Time penalty is 20 minutes per WA, **don't rush**!
5. Sample test cases must all be tested and passed before every submission!
6. Test the corner cases, such as 0, 1, -1. Test all edge cases of the input specification.
7. Bus error: the code has $scanf, fgets$ but have nothing to read! Check if you have early termination but didn't handle it properly.
8. Binary search? 數學算式移項合併後查詢?
9. Two Pointer <-> Binary Search
10. Directed graph connectivity -> DFS. Undirected graph -> Union Find
11. Check connectivity of the graph if the problem statement doesn't say anything (just loop over all nodes!)
12. $longlong = int * int$ won't work!
13. Shifting for $longlongint$ should be something like $1LL << 35$
14. For continuous input problems, be sure to read in all input BEFORE terminating and start processing next the input.
15. Don't use anonymous struct
16. 因式分解
17. 有時候, 從答案推回來會容易些!
18. 寫出數學式, 有時就馬上出現答案了!

4 Topic list

1. enumeration
2. greedy
3. sorting, topological sort
4. binary search
5. 離散化
6. Dynamic programming, 矩陣快速冪
7. Pigeonhole
8. LCA (倍增法, LCA 轉 RMQ)

5 Useful code

5.1 Leap year

```
year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)
```

5.2 Fast Exponentiation $O(log(exp))$

Fermat's little theorem: 若 m 是質數, 則 $a^{m-1} \equiv 1 \pmod{m}$

```
ll fast_pow(ll a, ll b, ll M) {
    ll ans = 1;
    ll base = a % M;
    while (b) {
        if (b & 1)
            ans = ans * base % M;
        base = base * base % M;
        b >>= 1;
    }
```

```
    return ans;
}
```

5.3 Mod Inverse

Case 1: $gcd(a, m) = 1$: ax + my = gcd(a, m) = 1 (use ext_gcd)

Case 2: $m\ is\ prime$: $a^{m-2} \equiv a^{-1} mod\ m$

5.4 GCD $O(log(a + b))$

注意負數的 case! C++ 是看被除數決定正負號的。

```
ll gcd(ll a, ll b)
{
    return b == 0 ? a : gcd(b, a % b);
}
```

5.5 Extended Euclidean Algorithm GCD $O(log(a + b))$

Bezout identity $ax + by = gcd(a, b)$, where $gcd(a, b)$ is the smallest positive integer that can be written as ax + by, and every integer of the form ax + by is a multiple of $gcd(a, b)$.

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

5.6 Prime Generator

```
const ll MAX_NUM = 1e6; // 要是合數
bool is_prime[MAX_NUM];
vector<ll> primes;

void init_primes() {
    fill(is_prime, is_prime + MAX_NUM, true);
    is_prime[0] = is_prime[1] = false;
    for (ll i = 2; i < MAX_NUM; i++) {
        if (is_prime[i]) {
            primes.push_back(i);
            for (ll j = i * i; j < MAX_NUM; j += i)
                is_prime[j] = false;
        }
    }
}
```

5.7   C++ Reference

```
1  vector/deque
2      ::[]: [idx] -> val // O(1)
3      ::erase: [it] -> it
4      ::erase: [it s, it t] -> it
5      ::resize: [sz, val = 0] -> void
6      ::insert: [it, val] -> void // insert before it
7      ::insert: [it, cnt, val] -> void // insert before it
8      ::insert: [it pos, it from_s, it from_t] -> void // insert before
       it
9
10 set/mulitset
11     ::insert: [val] -> pair<it, bool> // bool: if val already exist
12     ::erase: [val] -> void
13     ::erase: [it] -> void
14     ::clear: [] -> void
15     ::find: [val] -> it
16     ::count: [val] -> sz
17     ::lower_bound: [val] -> it
18     ::upper_bound: [val] -> it
19     ::equal_range: [val] -> pair<it, int>
20
21 map/mulitmap
22     ::begin/end: [] -> it (*it = pair<key, val>)
23     ::[]: [val] -> map_t&
24     ::insert: [pair<key, val>] -> pair<it, bool>
25     ::erase: [key] -> sz
26     ::clear: [] -> void
27     ::find: [key] -> it
28     ::count: [key] -> sz
29     ::lower_bound: [key] -> it
30     ::upper_bound: [key] -> it
31     ::equal_range: [key] -> it
32
33 algorithm
34     ::any_of: [it s, it t, unary_func] -> bool // C++11
35     ::all_of: [it s, it t, unary_func] -> bool // C++11
36     ::none_of: [it s, it t, unary_func] -> bool // C++11
37     ::find: [it s, it t, val] -> it
38     ::find_if: [it s, it t, unary_func] -> it
39     ::count: [it s, it t, val] -> int
40     ::count_if: [it s, it t, unary_func] -> int
41     ::copy: [it fs, it ft, it ts] -> void // t should be allocated
42     ::equal: [it s1, it t1, it s2, it t2] -> bool
43     ::remove: [it s, it t, val] -> it (it = new end)
44     ::unique: [it s, it t] -> it (it = new end)
45     ::random_shuffle: [it s, it t] -> void
46     ::lower_bound: [it s, it t, val, binary_func(a, b): a < b] -> it
47     ::upper_bound: [it s, it t, val, binary_func(a, b): a < b] -> it
48     ::binary_search: [it s, it t, val] -> bool ([s, t) sorted)
49     ::merge: [it s1, it t1, it s2, it t2, it o] -> void (o allocated)
50     ::includes: [it s1, it t1, it s2, it t2] -> bool (if 2 included in
       1)
51
52
53 string::
54     ::replace(idx, len, string) -> void
55     ::replace(it s1, it t1, it s2, it t2) -> void
56 string <-> int
57     ::stringstream; // remember to clear
58     ::sscanf(s.c_str(), "%d", &i);
59     ::sprintf(result, "%d", i); string s = result;
60
61 numeric
62     ::accumulate(it s, it t, val init);
63
64 math/cstdlib
65     ::atan2(0, -1) -> pi
66     ::sqrt(db/ldb) -> db/ldb
67     ::fabs(db/ldb) -> db/ldb
68     ::abs(int) -> int
69     ::ceil(db/ldb) -> db/ldb
70     ::floor(db/ldb) -> db/ldb
71     ::llabs(ll) -> ll (C++11)
72     ::round(db/ldb) -> db/ldb (C99, C++11)
73     ::log2(db) -> db (C99)
74     ::log2(ldb) -> ldb (C++11)
75
76 ctype
77     ::toupper(char) -> char (remain same if input is not alpha)
78     ::tolower(char) -> char (remain same if input is not alpha)
79     ::isupper(char) -> bool
80     ::islower(char) -> bool
81     ::isalpha(char) -> bool
82     ::isdigit(char) -> bool
83
84 io printf/scanf
85     ::int:                  "%d"       /   "%d"
86     ::double:               "%lf","f"  /   "%lf"
87     ::string:               "%s"       /   "%s"
88     ::long long:            "%lld"     /   "%lld"
89     ::long double:          "%Lf"      /   "%Lf"
90     ::unsigned int:         "%u"       /   "%u"
91     ::unsigned long long:   "%ull"     /   "%ull"
92     ::oct:                  "0%o"
93     ::hex:                  "0x%x"
94     ::scientific:           "%e"
95     ::width:                "%05d"
96     ::precision:            "%.5f"
97     ::adjust left:          "%-5d"
98
99 io cin/cout
100    ::oct:                  cout << oct << showbase;
101    ::hex:                  cout << hex << showbase;
102    ::scientific:           cout << scientific;
103    ::width:                cout << setw(5);
104    ::precision:            cout << fixed << setprecision(5);
105    ::adjust left:          cout << setw(5) << left;
```

6 Search

6.1 Ternary Search

6.2 折半完全列舉

能用 vector 就用 vector

6.3 Two-pointer 爬行法（右跑左追）

6.4 N Puzzle

```cpp
const int dr[4] = {0, 0, +1, -1};
const int dc[4] = {+1, -1, 0, 0};
const int dir[4] = {'R', 'L', 'D', 'U'};
const int INF = 0x3f3f3f3f;
const int FOUND = -1;
vector<char> path;
int A[15][15], Er, Ec;

int H() {
    int h = 0;
    for (int r = 0; r < 4; r++) {
        for (int c = 0; c < 4; c++) {
            if (A[r][c] == 0) continue;
            int expect_r = (A[r][c] - 1) / 4;
            int expect_c = (A[r][c] - 1) % 4;
            h += abs(expect_r - r) + abs(expect_c - c);
        }
    }
    return h;
}

int dfs(int g, int pdir, int bound) {
    int h = H();
    int f = g + h;
    if (f > bound) return f;
    if (h == 0) return FOUND;

    int mn = INF;
    for (int i = 0; i < 4; i++) {
        if (i == (pdir ^ 1)) continue;

        int nr = Er + dr[i];
        int nc = Ec + dc[i];
        if (nr < 0 || nr >= 4) continue;
        if (nc < 0 || nc >= 4) continue;

        path.push_back(dir[i]);
        swap(A[nr][nc], A[Er][Ec]);
        swap(nr, Er); swap(nc, Ec);
        int t = dfs(g + 1, i, bound);
        if (t == FOUND) return FOUND;
        if (t < mn) mn = t;
        swap(nr, Er); swap(nc, Ec);
        swap(A[nr][nc], A[Er][Ec]);
        path.pop_back();
    }

    return mn;
}

bool IDAstar() {
    int bound = H();
    for (;;) {
        int t = dfs(0, -1, bound);
        if (t == FOUND) return true;
        if (t == INF) return false;
        // 下次要搜的 bound >= 50，真的解也一定 >= 50，剪枝
        if (t >= 50) return false;
        bound = t;
    }
    return false;
}

bool solvable() {
    // cnt: 對於每一項 A[r][c] 有多少個小於它且在他之後的數，加總
    // (cnt + Er(1-based) % 2 == 0) <-> 有解
}
```

7 Basic data structure

7.1 1D BIT

```cpp
// BIT is 1-based
const int MAX_N = 20000; //這個記得改!
ll bit[MAX_N + 1];

ll sum(int i) {
    int s = 0;
    while (i > 0) {
        s += bit[i];
        i -= (i & -i);
    }
    return s;
}

void add(int i, ll x) {
    while (i <= MAX_N) {
        bit[i] += x;
        i += (i & -i);
    }
}
```

7.2 2D BIT

```cpp
// BIT is 1-based
const int MAX_N = 20000, MAX_M = 20000; //這個記得改!
ll bit[MAX_N + 1][MAX_M + 1];
```

```
 5   ll sum(int a, int b) {
 6       ll s = 0;
 7       for (int i = a; i > 0; i -= (i & -i))
 8           for (int j = b; j > 0; j -= (j & -j))
 9               s += bit[i][j];
10       return s;
11   }
12
13   void add(int a, int b, ll x) {
14       // MAX_N, MAX_M 須適時調整!
15       for (int i = a; i <= MAX_N; i += (i & -i))
16           for (int j = b; j <= MAX_M; j += (j & -j))
17               bit[i][j] += x;
18   }
```

7.3 Union Find

```
 1   #define N 20000 // 記得改
 2   struct UFDS {
 3       int par[N];
 4
 5       void init() {
 6           memset(par, -1, sizeof(par));
 7       }
 8
 9       int root(int x) {
10           return par[x] < 0 ? x : par[x] = root(par[x]);
11       }
12
13       void merge(int x, int y) {
14           x = root(x);
15           y = root(y);
16
17           if (x != y) {
18               if (par[x] > par[y])
19                   swap(x, y);
20               par[x] += par[y];
21               par[y] = x;
22           }
23       }
24   }
```

7.4 Segment Tree

```
 1   const int MAX_N = 100000;
 2   const int MAX_NN = (1 << 20); // should be bigger than MAX_N
 3
 4   int N;
 5   ll inp[MAX_N];
 6
 7   int NN;
 8   ll seg[2 * MAX_NN - 1];
 9   ll lazy[2 * MAX_NN - 1];
10   // lazy[u] != 0 : the subtree of u (u not included) is not up-to-date
```

```
11
12   void seg_gather(int u)
13   {
14       seg[u] = seg[u * 2 + 1] + seg[u * 2 + 2];
15   }
16
17   void seg_push(int u, int l, int m, int r)
18   {
19       if (lazy[u] != 0) {
20           seg[u * 2 + 1] += (m - l) * lazy[u];
21           seg[u * 2 + 2] += (r - m) * lazy[u];
22
23           lazy[u * 2 + 1] += lazy[u];
24           lazy[u * 2 + 2] += lazy[u];
25           lazy[u] = 0;
26       }
27   }
28
29   void seg_init()
30   {
31       NN = 1;
32       while (NN < N)
33           NN *= 2;
34
35       memset(seg, 0, sizeof(seg)); // val that won't affect result
36       memset(lazy, 0, sizeof(lazy)); // val that won't affect result
37       memcpy(seg + NN - 1, inp, sizeof(ll) * N); // fill in leaves
38   }
39
40   void seg_build(int u)
41   {
42       if (u >= NN - 1) { // leaf
43           return;
44       }
45
46       seg_build(u * 2 + 1);
47       seg_build(u * 2 + 2);
48       seg_gather(u);
49   }
50
51   void seg_update(int a, int b, int delta, int u, int l, int r)
52   {
53       if (l >= b || r <= a) {
54           return;
55       }
56
57       if (a <= l && r <= b) {
58           seg[u] += (r - l) * delta;
59           lazy[u] += delta;
60           return;
61       }
62
63       int m = (l + r) / 2;
64       seg_push(u, l, m, r);
65       seg_update(a, b, delta, u * 2 + 1, l, m);
66       seg_update(a, b, delta, u * 2 + 2, m, r);
```

```
67        seg_gather(u);
68 }
69
70 ll seg_query(int a, int b, int u, int l, int r)
71 {
72     if (l >= b || r <= a) {
73         return 0;
74     }
75
76     if (a <= l && r <= b) {
77         return seg[u];
78     }
79
80     int m = (l + r) / 2;
81     seg_push(u, l, m, r);
82     ll ans = 0;
83     ans += seg_query(a, b, u * 2 + 1, l, m);
84     ans += seg_query(a, b, u * 2 + 2, m, r);
85     seg_gather(u);
86
87     return ans;
88 }
```

7.5 Sparse Table

```
1 struct {
2     int sp[MAX_LOG_N][MAX_N]; // MAX_LOG_N = ceil(lg(MAX_N))
3
4     void build(int inp[], int n)
5     {
6         for (int j = 0; j < n; j++)
7             sp[0][j] = inp[j];
8
9         for (int i = 1; (1 << i) <= n; i++)
10            for (int j = 0; j + (1 << i) <= n; j++)
11                sp[i][j] = min(sp[i-1][j],sp[i-1][j+(1 << (i - 1))]);
12    }
13
14    int query(int l, int r) // [l, r)
15    {
16        int k = floor(log2(r - l));
17        return min(sp[k][l], sp[k][r - (1 << k)]);
18    }
19 } sptb;
```

8 Tree

8.1 LCA

```
1 const int MAX_N = 10000;
2 const int MAX_LOG_N = 14; // (1 << MAX_LOG_N) > MAX_N
3
4 int N;
5 int root;
```

```
6 int dep[MAX_N];
7 int par[MAX_LOG_N][MAX_N];
8
9 vector<int> child[MAX_N];
10
11 void dfs(int u, int p, int d) {
12     dep[u] = d;
13     for (int i = 0; i < int(child[u].size()); i++) {
14         int v = child[u][i];
15         if (v != p) {
16             dfs(v, u, d + 1);
17         }
18     }
19 }
20
21 void build() {
22     // par[0][u] and dep[u]
23     dfs(root, -1, 0);
24
25     // par[i][u]
26     for (int i = 0; i + 1 < MAX_LOG_N; i++) {
27         for (int u = 0; u < N; u++) {
28             if (par[i][u] == -1)
29                 par[i + 1][u] = -1;
30             else
31                 par[i + 1][u] = par[i][par[i][u]];
32         }
33     }
34 }
35
36 int lca(int u, int v) {
37     if (dep[u] > dep[v]) swap(u, v); // 讓 v 較深
38     int diff = dep[v] - dep[u]; // 將 v 上移到與 u 同層
39     for (int i = 0; i < MAX_LOG_N; i++) {
40         if (diff & (1 << i)) {
41             v = par[i][v];
42         }
43     }
44
45     if (u == v) return u;
46
47     for (int i = MAX_LOG_N - 1; i >= 0; i--) { // 必需倒序
48         if (par[i][u] != par[i][v]) {
49             u = par[i][u];
50             v = par[i][v];
51         }
52     }
53     return par[0][u];
54 }
```

8.2 Tree Center

```
1 int diameter = 0, radius[N], deg[N]; // deg = in + out degree
2 int findRadius()
3 {
```

```
 4        queue<int> q; // add all leaves in this group
 5        for (auto i : group)
 6            if (deg[i] == 1)
 7                q.push(i);
 8
 9        int mx = 0;
10        while (q.empty() == false) {
11            int u = q.front();
12            q.pop();
13
14            for (int v : g[u]) {
15                deg[v]--;
16                if (deg[v] == 1) {
17                    q.push(v);
18                    radius[v] = radius[u] + 1;
19                    mx = max(mx, radius[v]);
20                }
21            }
22        }
23
24        int cnt = 0; // crucial for knowing if there are 2 centers or not
25        for (auto j : group)
26            if (radius[j] == mx)
27                cnt++;
28
29        // add 1 if there are 2 centers (radius, diameter)
30        diameter = max(diameter, mx * 2 + (cnt == 2));
31        return mx + (cnt == 2);
32 }
```

8.3   Treap

```
 1 // Remember srand(time(NULL))
 2 struct Treap { // val: bst, pri: heap
 3     int pri, size, val;
 4     Treap *lch, *rch;
 5     Treap() {}
 6     Treap(int v) {
 7         pri = rand();
 8         size = 1;
 9         val = v;
10         lch = rch = NULL;
11     }
12 };
13
14 inline int size(Treap* t) {
15     return (t ? t->size : 0);
16 }
17 // inline void push(Treap* t) {
18 //     push lazy flag
19 // }
20 inline void pull(Treap* t) {
21     t->size = 1 + size(t->lch) + size(t->rch);
22 }
23
```

```
24 int NN = 0;
25 Treap pool[30000];
26
27 Treap* merge(Treap* a, Treap* b) { // a < b
28     if (!a || !b) return (a ? a : b);
29     if (a->pri > b->pri) {
30         // push(a);
31         a->rch = merge(a->rch, b);
32         pull(a);
33         return a;
34     }
35     else {
36         // push(b);
37         b->lch = merge(a, b->lch);
38         pull(b);
39         return b;
40     }
41 }
42
43 void split(Treap* t, Treap*& a, Treap*& b, int k) {
44     if (!t) { a = b = NULL; return; }
45     // push(t);
46     if (size(t->lch) < k) {
47         a = t;
48         split(t->rch, a->rch, b, k - size(t->lch) - 1);
49         pull(a);
50     }
51     else {
52         b = t;
53         split(t->lch, a, b->lch, k);
54         pull(b);
55     }
56 }
57
58 // get the rank of val
59 // result is 1-based
60 int get_rank(Treap* t, int val) {
61     if (!t) return 0;
62     if (val < t->val)
63         return get_rank(t->lch, val);
64     else
65         return get_rank(t->rch, val) + size(t->lch) + 1;
66 }
67
68 // get kth smallest item
69 // k is 1-based
70 Treap* get_kth(Treap*& t, int k) {
71     Treap *a, *b, *c, *d;
72     split(t, a, b, k - 1);
73     split(b, c, d, 1);
74     t = merge(a, merge(c, d));
75     return c;
76 }
77
78 void insert(Treap*& t, int val) {
79     int k = get_rank(t, val);
```

```
80      Treap *a, *b;
81      split(t, a, b, k);
82      pool[NN] = Treap(val);
83      Treap* n = &pool[NN++];
84      t = merge(merge(a, n), b);
85  }
86
87  // Implicit key treap init
88  void insert() {
89      for (int i = 0; i < N; i++) {
90          int val; scanf("%d", &val);
91          root = merge(root, new_treap(val)); // implicit key(index)
92      }
93  }
```

9 Graph

9.1 Articulation point / Bridge

```
1  // timer = 1, dfs arrays init to 0, set root carefully!
2  int timer, dfsTime[N], dfsLow[N], root;
3  bool articulationPoint[N]; // set<ii> bridge;
4  void findArticulationPoint(int u, int p)
5  {
6      dfsTime[u] = dfsLow[u] = timer++;
7
8      int child = 0; // root child counter for articulation point
9      for(auto v : g[u]) { // vector<int> g[N]; // undirected graph
10          if(v == p) // don't go back to parent
11              continue;
12
13          if(dfsTime[v] == 0) {
14              child++; // root child counter for articulation point
15              findArticulationPoint(v, u);
16              dfsLow[u] = min(dfsLow[u], dfsLow[v]);
17
18              // <= for articulation point, < for bridge
19              if(dfsTime[u] <= dfsLow[v] && root != u)
20                  articulationPoint[u] = true;
21              // special case for articulation point root only
22              if(u == root && child >= 2)
23                  articulationPoint[u] = true;
24          } else { // visited before (back edge)
25              dfsLow[u] = min(dfsLow[u], dfsTime[v]);
26          }
27      }
28  }
```

9.2 2-SAT

$$(x_i \vee x_i) \text{ 建邊}(\neg x_i, x_j)$$
$$(x_i \vee x_j) \text{ 建邊}(\neg x_i, x_j), (\neg x_j, x_i)$$
$$p \vee (q \wedge r)$$
$$= ((p \wedge q) \vee (p \wedge r))$$
$$p \oplus q$$
$$= \neg((p \wedge q) \vee (\neg p \wedge \neg q))$$
$$= (\neg p \vee \neg q) \wedge (p \vee q)$$

```
1  // 建圖
2  // (x1 or x2) and ... and (xi or xj)
3  // (xi or xj) 建邊
4  // ~xi -> xj
5  // ~xj -> xi
6
7  tarjan(); // scc 建立的順序是倒序的拓璞排序
8  for (int i = 0; i < 2 * N; i += 2) {
9      if (belong[i] == belong[i ^ 1]) {
10          // 無解
11      }
12  }
13  for (int i = 0; i < 2 * N; i += 2) { // 迭代所有變數
14      if (belong[i] < belong[i ^ 1]) { // i 的拓璞排序比 ~i 的拓璞排序大
15          // i = T
16      }
17      else {
18          // i = F
19      }
20  }
```

9.3 CC

9.3.1 BCC vertex

9.3.2 BCC edge

9.3.3 SCC

First of all we run DFS on the graph and sort the vertices in decreasing of their finishing time (we can use a stack).
Then, we start from the vertex with the greatest finishing time, and for each vertex v that is not yet in any SCC, do : for each u that v is reachable by u and u is not yet in any SCC, put it in the SCC of vertex v. The code is quite simple.

9.4 Shortest Path

Time complexity notations: V = vertex, E = edge

9.4.1 Dijkatra (next-to-shortest path)

密集圖別用 priority queue!

```
1  struct Edge {
2      int to, cost;
```

```cpp
};

typedef pair<int, int> P; // <d, v>
const int INF = 0x3f3f3f3f;

int N, R;
vector<Edge> g[5000];

int d[5000];
int sd[5000];

int solve() {
    fill(d, d + N, INF);
    fill(sd, sd + N, INF);
    priority_queue< P, vector<P>, greater<P> > pq;

    d[0] = 0;
    pq.push(P(0, 0));

    while (!pq.empty()) {
        P p = pq.top(); pq.pop();
        int v = p.second;

        if (sd[v] < p.first) // 比次短距離還大, 沒用, 跳過
            continue;

        for (size_t i = 0; i < g[v].size(); i++) {
            Edge& e = g[v][i];
            int nd = p.first + e.cost;
            if (nd < d[e.to]) { // 更新最短距離
                swap(d[e.to], nd);
                pq.push(P(d[e.to], e.to));
            }
            if (d[e.to] < nd && nd < sd[e.to]) { // 更新次短距離
                sd[e.to] = nd;
                pq.push(P(sd[e.to], e.to));
            }
        }
    }

    return sd[N-1];
}
```

9.4.2 SPFA

```cpp
typedef pair<int, int> ii;
vector< ii > g[N];

bool SPFA()
{
    vector<ll> d(n, INT_MAX);
    d[0] = 0; // origin

    queue<int> q;
    vector<bool> inqueue(n, false);
    vector<int> cnt(n, 0);
```

```cpp
    q.push(0);
    inqueue[0] = true;
    cnt[0]++;

    while(q.empty() == false) {
        int u = q.front();
        q.pop();
        inqueue[u] = false;

        for(auto i : g[u]) {
            int v = i.first, w = i.second;
            if(d[u] + w < d[v]) {
                d[v] = d[u] + w;
                if(inqueue[v] == false) {
                    q.push(v);
                    inqueue[v] = true;
                    cnt[v]++;

                    if(cnt[v] == n) { // loop!
                        return true;
                    }
                }
            }
        }
    }

    return false;
}
```

9.4.3 Bellman-Ford $O(VE)$

```cpp
vector<pair<ii, int>> edge; // store graph by edge: ((u, v), w)

void BellmanFord()
{
    ll d[n]; // n: total nodes
    fill(d, d + n, INT_MAX);
    d[0] = 0; // src is 0
    bool loop = false;
    for (int i = 0; i < n; i++) {
        // Do n - 1 times. If the n-th time still has relaxation, loop
        exists
        bool hasChange = false;
        for (int j = 0; j < (int)edge.size(); j++) {
            int u = edge[j].first.first, v = edge[j].first.second, w =
        edge[j].second;
            if (d[u] != INT_MAX && d[u] + w < d[v]) {
                hasChange = true;
                d[v] = d[u] + w;
            }
        }

        if (i == n - 1 && hasChange == true)
            loop = true;
        else if (hasChange == false)
            break;
```

```
24        }
25  }
```

9.4.4 Floyd-Warshall $O(V^3)$

The graph is stored using adjacency matrix. The initial state is $diagnal = 0$ and $others = INF$. (If INF is int, use long long for the matrix)
If diagonal numbers are negative $\leftarrow$ cycle .

```
1  for(int k = 0; k < N; k++)
2      for(int i = 0; i < N; i++)
3          for(int j = 0; j < N; j++)
4              dp[i][j] = min(dp[i][j], dp[i][k] + dp[k][j]);
```

9.5 MST

9.5.1 Kruskal

1. Store the graph by $(weight, (from, to))$
2. Sort the graph by $weight$
3. Start from the smallest weight, and keep adding edges that won't form a cycle with the current MST set
4. Early termination condition: $n - 1$ edges has been added, NOT size of the union-find set

9.5.2 Prim

```
1  priority_queue<ii, vector<ii>, greater<ii>> pq;
2  pq.push(ii(0, 0)); // push (0, origin)
3
4  uf.init(n + 1); // init union find
5  int ans = 0, earlyTermination = 0;
6  while (!pq.empty())
7  {
8      ii cur = pq.top();
9      pq.pop();
10
11     int u = cur.second;
12     if (u != 0 && (uf.root(0) == uf.root(u))) // check loop
13         continue;
14     uf.merge(0, u);
15     ans += cur.first;
16     earlyTermination++;
17     if (earlyTermination == n) // origin node is dummy node
18         break;
19
20     for (int i = 0; i < (int)g[u].size(); i++) {
21         int v = g[u][i].first, w = g[u][i].second;
22
23         if (uf.root(0) != uf.root(v)) {
24             pq.push(ii(w, v));
25         }
26     }
27  }
```

10 Flow

10.1 Max Flow (Dinic)

```
1  struct Edge {
2      int to, cap, rev;
3      Edge(int a, int b, int c) {
4          to = a;
5          cap = b;
6          rev = c;
7      }
8  };
9
10 const int INF = 0x3f3f3f3f;
11 const int MAX_V = 20000 + 10;
12 // vector<Edge> g[MAX_V];
13 vector< vector<Edge> > g(MAX_V);
14 int level[MAX_V];
15 int iter[MAX_V];
16
17 inline void add_edge(int u, int v, int cap) {
18     g[u].push_back((Edge){v, cap, (int)g[v].size()});
19     g[v].push_back((Edge){u, 0, (int)g[u].size() - 1});
20 }
21
22 void bfs(int s) {
23     memset(level, -1, sizeof(level));
24     queue<int> q;
25
26     level[s] = 0;
27     q.push(s);
28
29     while (!q.empty()) {
30         int v = q.front(); q.pop();
31         for (int i = 0; i < int(g[v].size()); i++) {
32             const Edge& e = g[v][i];
33             if (e.cap > 0 && level[e.to] < 0) {
34                 level[e.to] = level[v] + 1;
35                 q.push(e.to);
36             }
37         }
38     }
39 }
40
41 int dfs(int v, int t, int f) {
42     if (v == t) return f;
43     for (int& i = iter[v]; i < int(g[v].size()); i++) {
44         Edge& e = g[v][i];
45         if (e.cap > 0 && level[v] < level[e.to]) {
46             int d = dfs(e.to, t, min(f, e.cap));
47             if (d > 0) {
48                 e.cap -= d;
49                 g[e.to][e.rev].cap += d;
50                 return d;
51             }
52         }
```

```
52            }
53        }
54        return 0;
55 }
56
57 int max_flow(int s, int t) { // dinic
58     int flow = 0;
59     for (;;) {
60         bfs(s);
61         if (level[t] < 0) return flow;
62         memset(iter, 0, sizeof(iter));
63         int f;
64         while ((f = dfs(s, t, INF)) > 0) {
65             flow += f;
66         }
67     }
68 }
```

10.2 Min Cost Flow

```
1  #define st first
2  #define nd second
3
4  typedef pair<double, int> pii;
5  const double INF = 1e10;
6
7  struct Edge {
8      int to, cap;
9      double cost;
10     int rev;
11 };
12
13 const int MAX_V = 2 * 100 + 10;
14 int V;
15 vector<Edge> g[MAX_V];
16 double h[MAX_V];
17 double d[MAX_V];
18 int prevv[MAX_V];
19 int preve[MAX_V];
20 // int match[MAX_V];
21
22 void add_edge(int u, int v, int cap, double cost) {
23     g[u].push_back((Edge){v, cap, cost, (int)g[v].size()});
24     g[v].push_back((Edge){u, 0, -cost, (int)g[u].size() - 1});
25 }
26
27 double min_cost_flow(int s, int t, int f) {
28     double res = 0;
29     fill(h, h + V, 0);
30     fill(match, match + V, -1);
31     while (f > 0) {
32         // dijkstra 找最小成本增廣路徑
33         // without h will reduce to SPFA = O(V*E)
34         fill(d, d + V, INF);
```

```
35        priority_queue< pii, vector<pii>, greater<pii> > pq;
36
37        d[s] = 0;
38        pq.push(pii(d[s], s));
39
40        while (!pq.empty()) {
41            pii p = pq.top(); pq.pop();
42            int v = p.nd;
43            if (d[v] < p.st) continue;
44            for (size_t i = 0; i < g[v].size(); i++) {
45                const Edge& e = g[v][i];
46                if (e.cap > 0 && d[e.to] > d[v] + e.cost + h[v] - h[e.
   to]) {
47                    d[e.to] = d[v] + e.cost + h[v] - h[e.to];
48                    prevv[e.to] = v;
49                    preve[e.to] = i;
50                    pq.push(pii(d[e.to], e.to));
51                }
52            }
53        }
54
55        // 找不到增廣路徑
56        if (d[t] == INF) return -1;
57
58        // 維護 h[v]
59        for (int v = 0; v < V; v++)
60            h[v] += d[v];
61
62        // 找瓶頸
63        int bn = f;
64        for (int v = t; v != s; v = prevv[v])
65            bn = min(bn, g[prevv[v]][preve[v]].cap);
66
67        // // find match
68        // for (int v = prevv[t]; v != s; v = prevv[prevv[v]]) {
69        //     int u = prevv[v];
70        //     match[v] = u;
71        //     match[u] = v;
72        // }
73
74        // 更新剩餘圖
75        f -= bn;
76        res += bn * h[t]; // SPFA: res += bn * d[t]
77        for (int v = t; v != s; v = prevv[v]) {
78            Edge& e = g[prevv[v]][preve[v]];
79            e.cap -= bn;
80            g[v][e.rev].cap += bn;
81        }
82    }
83    return res;
84 }
```

10.3 Bipartite Matching

```cpp
const int MAX_V = ...;
int V;
vector<int> g[MAX_V];
int match[MAX_V];
bool used[MAX_V];

void add_edge(int u, int v) {
    g[u].push_back(v);
    g[v].push_back(u);
}

// 回傳有無找到從 v 出發的增廣路徑
// (首尾都為未匹配點的交錯路徑)
// [待確認] 每次遞迴都找一個未匹配點 v 及匹配點 u
bool dfs(int v) {
    used[v] = true;
    for (size_t i = 0; i < g[v].size(); i++) {
        int u = g[v][i], w = match[u];
        // 尚未配對或可從 w 找到增廣路徑 (即路徑繼續增長)
        if (w < 0 || (!used[w] && dfs(w))) {
            // 交錯配對
            match[v] = u;
            match[u] = v;
            return true;
        }
    }
    return false;
}

int bipartite_matching() { // 匈牙利演算法
    int res = 0;
    memset(match, -1, sizeof(match));
    for (int v = 0; v < V; v++) {
        if (match[v] == -1) {
            memset(used, false, sizeof(used));
            if (dfs(v)) {
                res++;
            }
        }
    }
    return res;
}
```

11 String

11.1 Rolling Hash

1. Use two rolling hashes if needed.
2. The prime for pre-calculation can be 137 and 257, for modulo can be $1e9 + 7$ and $0xdefaced$

```cpp
#define N 1000100
#define B 137
#define M 1000000007

typedef long long ll;
```

```cpp
char inp[N];
int len;
ll p[N], h[N];

void init()
{ // build polynomial table and hash value
    p[0] = 1; // b to the ith power
    for (int i = 1; i <= len; i++) {
        h[i] = (h[i - 1] * B % M + inp[i - 1]) % M; // hash value
        p[i] = p[i - 1] * B % M;
    }
}

ll get_hash(int l, int r) // [l, r] of the inp string array
{
    return ((h[r + 1] - (h[l] * p[r - l + 1])) % M + M) % M;
}
```

11.2 KMP

```cpp
void fail()
{
    int len = strlen(pat);

    f[0] = 0;
    int j = 0;
    for (int i = 1; i < len; i++) {
        while (j != 0 && pat[i] != pat[j])
            j = f[j - 1];

        if (pat[i] == pat[j])
            j++;

        f[i] = j;
    }
}

int match()
{
    int res = 0;
    int j = 0, plen = strlen(pat), tlen = strlen(text);

    for (int i = 0; i < tlen; i++) {
        while (j != 0 && text[i] != pat[j])
            j = f[j - 1];

        if (text[i] == pat[j]) {
            if (j == plen - 1) { // find match
                res++;
                j = f[j];
            } else {
                j++;
            }
        }
    }
}
```

11.3   Z Algorithm

```
int len = strlen(inp), z[len];
z[0] = 0; // initial

int l = 0, r = 0; // z box bound [l, r]
for (int i = 1; i < len; i++)
{
    if (i > r) { // i not in z box
        l = r = i; // z box contains itself only
        while (r < len && inp[r - l] == inp[r])
            r++;
        z[i] = r - l;
        r--;
    } else { // i in z box
        if (z[i - l] + i < r) // over shoot R bound
            z[i] = z[i - l];
        else {
            l = i;
            while (r < len && inp[r - l] == inp[r])
                r++;
            z[i] = r - l;
            r--;
        }
    }
}
```

11.4   Trie

注意 count 的擺放位置，視題意可以擺在迴圈外

```
struct Node {
    int cnt;
    Node* nxt[2];
    Node() {
        cnt = 0;
        fill(nxt, nxt + 2, nullptr);
    }
};

const int MAX_Q = 200000;
int Q;

int NN = 0;
Node data[MAX_Q * 30];
Node* root = &data[NN++];

void insert(Node* u, int x) {
    for (int i = 30; i >= 0; i--) {
        int t = ((x >> i) & 1);
        if (u->nxt[t] == nullptr) {
```

```
            u->nxt[t] = &data[NN++];
        }

        u = u->nxt[t];
        u->cnt++;
    }
}

void remove(Node* u, int x) {
    for (int i = 30; i >= 0; i--) {
        int t = ((x >> i) & 1);
        u = u->nxt[t];
        u->cnt--;
    }
}

int query(Node* u, int x) {
    int res = 0;
    for (int i = 30; i >= 0; i--) {
        int t = ((x >> i) & 1);
        // if it is possible to go the another branch
        // then the result of this bit is 1
        if (u->nxt[t ^ 1] != nullptr && u->nxt[t ^ 1]->cnt > 0) {
            u = u->nxt[t ^ 1];
            res |= (1 << i);
        }
        else {
            u = u->nxt[t];
        }
    }
    return res;
}
```

11.5   Suffix Array

12   Matrix

12.1   Gauss Jordan

```
typedef long long ll;
typedef vector<ll> vec;
typedef vector<vec> mat;

vec gauss_jordan(mat A) {
    int n = A.size(), m = A[0].size();
    for (int i = 0; i < n; i++) {
        // float: find j s.t. A[j][i] is max
        // mod: find min j s.t. A[j][i] is not 0
        int pivot = i;
        for (int j = i; j < n; j++) {
            // if (fabs(A[j][i]) > fabs(A[pivot])) {
            //     pivot = j;
            // }
            if (A[pivot][i] != 0) {
                pivot = j;
```

```
17                break;
18            }
19        }
20
21        swap(A[i], A[pivot]);
22        if (A[i][i] == 0) { // if (fabs(A[i][i]) < eps)
23            // 無解或無限多組解
24            // 可改成 continue, 全部做完後再判
25            return vec();
26        }
27
28        ll divi = inv(A[i][i]);
29        for (int j = i; j < m; j++) {
30            // A[i][j] /= A[i][i];
31            A[i][j] = (A[i][j] * divi) % MOD;
32        }
33
34        for (int j = 0; j < n; j++) {
35            if (j != i) {
36                for (int k = i + 1; k < m; k++) {
37                    // A[j][k] -= A[j][i] * A[i][k];
38                    ll p = (A[j][i] * A[i][k]) % MOD;
39                    A[j][k] = (A[j][k] - p + MOD) % MOD;
40                }
41            }
42        }
43    }
44
45    vec x(n);
46    for (int i = 0; i < n; i++)
47        x[i] = A[i][m - 1];
48    return x;
49 }
```

12.2 Determinant

```
1  typedef long long ll;
2  typedef vector<ll> vec;
3  typedef vector<vec> mat;
4
5  ll determinant(mat m) { // square matrix
6      const int n = m.size();
7      ll det = 1;
8      for (int i = 0; i < n; i++) {
9          for (int j = i + 1; j < n; j++) {
10             int a = i, b = j;
11             while (m[b][i]) {
12                 ll q = m[a][i] / m[b][i];
13                 for (int k = 0; k < n; k++)
14                     m[a][k] = m[a][k] - m[b][k] * q;
15                 swap(a, b);
16             }
17
18             if (a != i) {
19                 swap(m[i], m[j]);
```

```
20                 det = -det;
21             }
22         }
23
24         if (m[i][i] == 0)
25             return 0;
26         else
27             det *= m[i][i];
28     }
29     return det;
30 }
```

13 Geometry

1. Keep things in integers as much as possible!
2. Try not to divide
3. If you have decimals, if they are fixed precision, you can usually just multiply all the input and use integers instead

13.1 EPS

= 0: $fabs \le eps$
< 0: $< -eps$
> 0: $> +eps$

13.2 Template

```
1  // if the points are given in doubles form, change the code accordingly
2
3  typedef long long ll;
4
5  typedef pair<ll, ll> pt;   // points are stored using long long
6  typedef pair<pt, pt> seg; // segments are a pair of points
7
8  #define x first
9  #define y second
10
11 #define EPS 1e-9
12
13 pt operator+(pt a, pt b)
14 {
15     return pt(a.x + b.x, a.y + b.y);
16 }
17
18 pt operator-(pt a, pt b)
19 {
20     return pt(a.x - b.x, a.y - b.y);
21 }
22
23 pt operator*(pt a, int d)
24 {
25     return pt(a.x * d, a.y * d);
26 }
27
28 ll cross(pt a, pt b)
29 {
```

```
30         return a.x * b.y - a.y * b.x;
31    }
32
33    int ccw(pt a, pt b, pt c)
34    {
35        ll res = cross(b - a, c - a);
36        if (res > 0) // left turn
37            return 1;
38        else if (res == 0) // straight
39            return 0;
40        else // right turn
41            return -1;
42    }
43
44    double dist(pt a, pt b)
45    {
46        double dx = a.x - b.x;
47        double dy = a.y - b.y;
48        return sqrt(dx * dx + dy * dy);
49    }
50
51    bool zero(double x)
52    {
53        return fabs(x) <= EPS;
54    }
55
56    bool overlap(seg a, seg b)
57    {
58        return ccw(a.x, a.y, b.x) == 0 && ccw(a.x, a.y, b.y) == 0;
59    }
60
61    bool intersect(seg a, seg b)
62    {
63        if (overlap(a, b) == true) { // non-proper intersection
64            double d = 0;
65            d = max(d, dist(a.x, a.y));
66            d = max(d, dist(a.x, b.x));
67            d = max(d, dist(a.x, b.y));
68            d = max(d, dist(a.y, b.x));
69            d = max(d, dist(a.y, b.y));
70            d = max(d, dist(b.x, b.y));
71
72            // d > dist(a.x, a.y) + dist(b.x, b.y)
73            if (d - (dist(a.x, a.y) + dist(b.x, b.y)) > EPS)
74                return false;
75            return true;
76        }
77        //                    |
78        // Equal sign for -----| case
79        // non qeual sign => proper intersection
80        if (ccw(a.x, a.y, b.x) * ccw(a.x, a.y, b.y) <= 0 &&
81            ccw(b.x, b.y, a.x) * ccw(b.x, b.y, a.y) <= 0)
82            return true;
83        return false;
84    }
85
```

```
86    double area(vector<pt> pts)
87    {
88        double res = 0;
89        int n = pts.size();
90        for (int i = 0; i < n; i++)
91            res += (pts[i].y + pts[(i + 1) % n].y) * (pts[(i + 1) % n].x -
    pts[i].x);
92        return res / 2.0;
93    }
94
95    vector<pt> halfHull(vector<pt> &points)
96    {
97        vector<pt> res;
98
99        for (int i = 0; i < (int)points.size(); i++) {
100           while ((int)res.size() >= 2 &&
101                   ccw(res[res.size() - 2], res[res.size() - 1], points[i])
     < 0)
102               res.pop_back(); // res.size() - 2 can't be assign before
      size() >= 2
103               // check, bitch
104
105           res.push_back(points[i]);
106       }
107
108       return res;
109   }
110
111   vector<pt> convexHull(vector<pt> &points)
112   {
113       vector<pt> upper, lower;
114
115       // make upper hull
116       sort(points.begin(), points.end());
117
118       upper = halfHull(points);
119       // make lower hull
120       reverse(points.begin(), points.end());
121       lower = halfHull(points);
122
123       // merge hulls
124       if ((int)upper.size() > 0) // yes sir~
125           upper.pop_back();
126       if ((int)lower.size() > 0)
127           lower.pop_back();
128
129       vector<pt> res(upper.begin(), upper.end());
130       res.insert(res.end(), lower.begin(), lower.end());
131
132       return res;
133   }
134
135   bool completelyInside(vector<pt> &outer, vector<pt> &inner)
136   {
137       int even = 0, odd = 0;
138       for (int i = 0; i < (int)inner.size(); i++) {
```

```
139        // y = slope * x + offset
140        int cntIntersection = 0;
141        ll slope = rand() % INT_MAX + 1;
142        ll offset = inner[i].y - slope * inner[i].x;
143
144        ll farx = 111111 * (slope >= 0 ? 1 : -1);
145        ll fary = farx * slope + offset;
146        seg a = seg(pt(inner[i].x, inner[i].y), pt(farx, fary));
147        for (int j = 0; j < (int)outer.size(); j++) {
148            seg b = seg(outer[j], outer[(j + 1) % (int)outer.size()]);
149
150            if ((b.x.x * slope + offset == b.x.y) ||
151                (b.y.x * slope + offset == b.y.y)) { // on-line
152                i--;
153                break;
154            }
155
156            if (intersect(a, b) == true)
157                cntIntersection++;
158        }
159
160        if (cntIntersection % 2 == 0) // outside
161            even++;
162        else
163            odd++;
164    }
165
166    return odd == (int)inner.size();
167 }
168
169 // srand(time(NULL))
170 // rand()
```

14 Math

14.1 Euclid's formula (Pythagorean Triples)

$a = p^2 - q^2$
$b = 2pq$ **(always even)**
$c = p^2 + q^2$

14.2 Difference between two consecutive numbers' square is odd

$(k + 1)^2 - k^2 = 2k + 1$

14.3 Summation

$\sum_{k=1}^{n} 1 = n$
$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$

14.4 FFT

```cpp
typedef unsigned int ui;
typedef long double ldb;
const ldb pi = atan2(0, -1);

struct Complex {
    ldb real, imag;
    Complex(): real(0.0), imag(0.0) {;}
    Complex(ldb a, ldb b) : real(a), imag(b) {;}
    Complex conj() const {
        return Complex(real, -imag);
    }
    Complex operator + (const Complex& c) const {
        return Complex(real + c.real, imag + c.imag);
    }
    Complex operator - (const Complex& c) const {
        return Complex(real - c.real, imag - c.imag);
    }
    Complex operator * (const Complex& c) const {
        return Complex(real*c.real - imag*c.imag, real*c.imag + imag*c.real);
    }
    Complex operator / (ldb x) const {
        return Complex(real / x, imag / x);
    }
    Complex operator / (const Complex& c) const {
        return *this * c.conj() / (c.real * c.real + c.imag * c.imag);
    }
};

inline ui rev_bit(ui x, int len){
    x = ((x & 0x55555555u) << 1)  | ((x & 0xAAAAAAAAu) >> 1);
    x = ((x & 0x33333333u) << 2)  | ((x & 0xCCCCCCCCu) >> 2);
    x = ((x & 0x0F0F0F0Fu) << 4)  | ((x & 0xF0F0F0F0u) >> 4);
    x = ((x & 0x00FF00FFu) << 8)  | ((x & 0xFF00FF00u) >> 8);
    x = ((x & 0x0000FFFFu) << 16) | ((x & 0xFFFF0000u) >> 16);
    return x >> (32 - len);
}

// flag = -1 if ifft else +1
void fft(vector<Complex>& a, int flag = +1) {
    int n = a.size(); // n should be power of 2

    int len = __builtin_ctz(n);
    for (int i = 0; i < n; i++) {
        int rev = rev_bit(i, len);

        if (i < rev)
            swap(a[i], a[rev]);
    }

    for (int m = 2; m <= n; m <<= 1) { // width of each item
        auto wm = Complex(cos(2 * pi / m), flag * sin(2 * pi / m));
        for (int k = 0; k < n; k += m) { // start idx of each item
            auto w = Complex(1, 0);
```

```
54          for (int j = 0; j < m / 2; j++) { // iterate half
55              Complex t = w * a[k + j + m / 2];
56              Complex u = a[k + j];
57              a[k + j] = u + t;
58              a[k + j + m / 2] = u - t;
59              w = w * wm;
60          }
61      }
62  }
63
64  if (flag == -1) { // if it's ifft
65      for (int i = 0; i < n; i++)
66          a[i].real /= n;
67  }
68 }
69
70 vector<int> mul(const vector<int>& a, const vector<int>& b) {
71     int n = int(a.size()) + int(b.size()) - 1;
72     int nn = 1;
73     while (nn < n)
74         nn <<= 1;
75
76     vector<Complex> fa(nn, Complex(0, 0));
77     vector<Complex> fb(nn, Complex(0, 0));
78     for (int i = 0; i < int(a.size()); i++)
79         fa[i] = Complex(a[i], 0);
80     for (int i = 0; i < int(b.size()); i++)
81         fb[i] = Complex(b[i], 0);
82
83     fft(fa, +1);
84     fft(fb, +1);
85     for (int i = 0; i < nn; i++) {
86         fa[i] = fa[i] * fb[i];
87     }
88     fft(fa, -1);
89
90     vector<int> c;
91     for(int i = 0; i < nn; i++) {
92         int val = int(fa[i].real + 0.5);
93         if (val) {
94             while (int(c.size()) <= i)
95                 c.push_back(0);
96             c[i] = 1;
97         }
98     }
99
100    return c;
101 }
```

14.5 Combination

14.5.1 Pascal triangle

```
1 #define N 210
2 ll C[N][N];
3
```

```
4  void Combination() {
5      for(ll i=0; i<N; i++) {
6          C[i][0] = 1;
7          C[i][i] = 1;
8      }
9
10     for(ll i=2; i<N; i++) {
11         for(ll j=1; j<=i; j++) {
12             C[i][j] = (C[i-1][j] + C[i-1][j-1])%M; // if needed, mod it
13         }
14     }
15 }
```

14.5.2 線性

```
1  ll binomialCoeff(ll n, ll k)
2  {
3      ll res = 1;
4
5      if ( k > n - k ) // Since C(n, k) = C(n, n-k)
6          k = n - k;
7
8      for (int i = 0; i < k; ++i) // n...n-k / 1...k
9      {
10         res *= (n - i);
11         res /= (i + 1);
12     }
13
14     return res;
15 }
```

14.6 重複組合

14.7 Chinese remainder theorem

```
1  typedef long long ll;
2
3  struct Item {
4      ll m, r;
5  };
6
7  ll extgcd(ll a, ll b, ll &x, ll &y)
8  {
9      if (b == 0) {
10         x = 1;
11         y = 0;
12         return a;
13     } else {
14         ll d = extgcd(b, a % b, y, x);
15         y -= (a / b) * x;
16         return d;
17     }
18 }
19
```

```
20  Item extcrt(const vector<Item> &v)
21  {
22      ll m1 = v[0].m, r1 = v[0].r, x, y;
23
24      for (int i = 1; i < int(v.size()); i++) {
25          ll m2 = v[i].m, r2 = v[i].r;
26          ll g = extgcd(m1, m2, x, y); // now x = (m/g)^(-1)
27
28          if ((r2 - r1) % g != 0)
29              return {-1, -1};
30
31          ll k = (r2 - r1) / g * x % (m2 / g);
32          k = (k + m2 / g) % (m2 / g); // for the case k is negative
33
34          ll m = m1 * m2 / g;
35          ll r = (m1 * k + r1) % m;
36
37          m1 = m;
38          r1 = (r + m) % m; // for the case r is negative
39      }
40
41      return (Item) {
42          m1, r1
43      };
44  }
```

14.8   2-Circle relations

$d = $ 圓心距, R, r 為半徑 $(R \geq r)$
內切: $d = R - r$
外切: $d = R + r$
內離: $d < R - r$
外離: $d > R + r$
相交: $d < R + r$ 且 $d > R - r$

14.9   Fun Facts

1. 如果 $\frac{b}{a}$ 是最簡分數, 則 $1 - \frac{b}{a}$ 也是
2.

15   Dynamic Programming - Problems collection

```
1   // # 零一背包 (poj 1276)
2   fill(dp, dp + W + 1, 0);
3   for (int i = 0; i < N; i++)
4       for (int j = W; j >= items[i].w; j--)
5           dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
6   return dp[W];
7
8   // # 多重背包二進位拆解 (poj 1276)
9   for_each(ll v, w, num) {
10      for (ll k = 1; k <= num; k *= 2) {
11          items.push_back((Item) {k * v, k * w});
12          num -= k;
13      }
14      if (num > 0)
15          items.push_back((Item) {num * v, num * w});
16  }
17
18  // # 完全背包
19  // dp[i][j] = 前 i + 1 個物品, 在重量 j 下所能組出的最大價值
20  // 第 i 個物品, 不放或至少放一個
21  // dp[i][j] = max(dp[i - 1][j], dp[i][j - w[i]] + v[i])
22  fill(dp, dp + W + 1, 0);
23  for (int i = 0; i < N; i++)
24      for (int j = w[i]; j <= W; j++)
25          dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
26  return dp[W];
27
28  // # Coin Change (2015 桂冠賽 E)
29  // dp[i][j] = 前 i + 1 個物品, 組出 j 元的方法數
30  // 第 i 個物品, 不用或用至少一個
31  // dp[i][j] = dp[i - 1][j] + dp[i][j - coin[i]]
32
33  // # Cutting Sticks (2015 桂冠賽 F)
34  // 補上二個切點在最左與最右
35  // dp[i][j] = 使 (i, j) 區間中的所有切點都被切的最小成本
36  // dp[i][j] = min(dp[i][c] + dp[c][j] + (p[j] - p[i]) for i < c < j)
37  // dp[i][i + 1] = 0
38  // ans = dp[0][N + 1]
39
40  // # Throwing a Party (itsa dp 06)
41  // 給定一棵有根樹, 代表公司職位層級圖, 每個人有其權重, 現從中選一個點集合出來,
42  // 且一個人不能與其上司一都在集合中, 並最大化集合的權重和, 輸出該總和。
43  // dp[u][0/1] = u 在或不在集合中, 以 u 為根的子樹最大權重和
44  // dp[u][0] = max(max(dp[c][0], dp[c][1]) for children c of u) + val[u]
45  // dp[u][1] = max(dp[c][0] for children c of u)
46  // bottom up dp
47
48  // # LIS (O(N^2))
49  // dp[i] = 以 i 為結尾的 LIS 的長度
50  // dp[i] = max(dp[j] for 0 <= j < i) + 1
51  // ans = max(dp)
52
53  // # LIS (O(nlgn)), poj 1631
54  // dp[i] = 長度為 i + 1 的 LIS 的最後一項的最小值, 不存在時為 INF
55  fill(dp, dp + N, INF);
56  for (int i = 0; i < N; i++)
57      *lower_bound(dp, dp + N, A[i]) = A[i];
58  ans = lower_bound(dp, dp + N, INF) - dp;
59
60  // # Maximum Subarray
61
62  // # Not equal on a Segment (cf edu7 C)
63  // 給定長度為 n 的陣列 a[] 與 m 個詢問。
64  // 針對每個詢問 l, r, x 請輸出 a[l, r] 中不等於 x 的任一位置。
65  // 不存在時輸出 -1
66  // dp[i] = max j such that j < i and a[j] != a[i]
67  // dp[0] = -1
68  // dp[i] = dp[i - 1] if a[i] == a[i - 1] else i - 1
69  // 針對每筆詢問 l, r, x
70  // 1. a[r] != x                    -> 輸出 r
```

```
// 2. a[r] = x && dp[r] >= l    -> 輸出 dp[r]
// 3. a[r] = x && dp[r] < l      -> 輸出 -1

// # bitmask dp, poj 2686
// 給定一個無向帶權圖, 代表 M 個城市之間的路, 與 N 張車票,
// 每張車票有一個數值 t[i], 若欲使用車票 t[i] 從城市 u 經由路徑 d[u][v] 走到城市
//     v,
// 所花的時間為 d[u][v] / t[i]。請問, 從城市 A 走到城市 B 最快要多久?
// dp[S][v] = 從城市 A 到城市 v 的最少時間, 其中 S 為用過的車票的集合
// 考慮前一個城市 u 是誰, 使用哪個車票 t[i] 而來, 可以得到轉移方程式:
// dp[S][v] = min([
//      dp[S - {v}][u] + d[u][v] / t[i]
//      for all city u has edge to v, for all ticket in S
// ])

// # Tug of War
// N 個人參加拔河比賽, 每個人有其重量 w[i], 欲使二隊的人數最多只差一, 雙方的重量和越
//     接近越好
// 請問二隊的重量和分別是多少?
// dp[i][j][k] = 只考慮前 i + 1 個人, 可不可以使左堆的重量為 j, 且左堆的人數為 k
// dp[i][j][k] = dp[i - 1][j - w[i][k - 1] or dp[i - 1][j][k]
// dp[i][j] = (dp[i - 1][j - w[i]] << 1) | (dp[i - 1][j])

// # Modulo Sum (cf 319 B)
// 給定長度為 N 的序列 A 與一正整數 M, 請問該序列中有無一個子序列, 子序列的總合是 M
//     的倍數
// 若 N > M, 則根據鴿籠原理, 必有至少兩個前綴和的值 mod M 為相同值, 解必定存在
// dp[i][j] = 前 i + 1 個數可否組出 mod m = j 的數
// dp[i][j] = true if
//      dp[i - 1][(j - (a[i] mod m)) mod m] or
//      dp[i - 1][j] or
//      j = a[i] % m

// # POJ 2229
// 給定正整數 N, 請問將 N 拆成一堆 2^x 之和的方法數
// dp[i] = 拆解 N 的方法數
// dp[i] = dp[i / 2] if i is odd
//        = dp[i - 1] + dp[i / 2] if i is even

// # POJ 3616
// 給定 N 個區間 [s, t], 每個區間有權重 w[i], 從中選出一些不相交的區間, 使權重和最
//     大
// dp[i] = 考慮前 i + 1 個區間, 且必選第 i 個區間的最大權重和
// dp[i] = max(dp[j] | 0 <= j < i) + w[i]
// ans = max(dp)

// # POJ 2184
// N 隻牛每隻牛有權重 <s, f>, 從中選出一些牛的集合,
// 使得 sum(s) + sum(f) 最大, 且 sum(s) > 0, sum(f) > 0。
// 枚舉 sum(s) , 將 sum(s) 視為重量對 f 做零一背包。

// # POJ 3666
// 給定長度為 N 的序列, 請問最少要加多少值, 使得序列單調遞增
// dp[i][j] = 使序列前 i+1 項變為單調, 且將 A[i] 變為「第 j 小的數」的最小成本
// dp[i][j] = min(dp[i - 1][k] | 0 <= k <= j) + abs(S[j] - A[i])
// min(dp[i - 1][k] | 0 <= k <= j) 動態維護

for (int j = 0; j < N; j++)
    dp[0][j] = abs(S[j] - A[0]);
for (int i = 1; i < N; i++) {
    int pre_min_cost = dp[i][0];
    for (int j = 0; j < N; j++) {
        pre_min_cost = min(pre_min_cost, dp[i-1][j]);
        dp[i][j] = pre_min_cost + abs(S[j] - A[i]);
    }
}
ans = min(dp[N - 1])

// # POJ 3734
// N 個 blocks 上色, R, G, Y, B, 上完色後紅色的數量與綠色的數量都要是偶數。請問方
//     法數。
// dp[i][0/1/2/3] = 前 i 個 blocks 上完色, 紅色數量為奇數/偶數, 綠色數量為數/偶
//     數
// 用遞推, 考慮第 i + 1 個 block 的顏色, 找出個狀態的轉移, 整理可發現
// dp[i + 1][0] = dp[i][2] + dp[i][1] + 2 * dp[i][0]
// dp[i + 1][1] = dp[i][3] + dp[i][0] + 2 * dp[i][1]
// dp[i + 1][2] = dp[i][0] + dp[i][3] + 2 * dp[i][2]
// dp[i + 1][3] = dp[i][1] + dp[i][2] + 2 * dp[i][3]
// 矩陣快速冪加速求 dp[N - 1][0][0]

// # POJ 3171
// 數線上, 給定 N 個區間 [s[i], t[i]], 每個區間有其代價, 求覆蓋區間 [M, E] 的最
//     小代價。
// dp[i][j] = 最多使用前 i + 1 個區間, 使 [M, j] 被覆蓋的最小代價
// 考慮第 i 個區間用或不用, 可得:
// dp[i][j] =
//      1. min(dp[i - 1][k] for k in [s[i] - 1, t[i]]) + cost[i] if j =
//      t[i]
//      2. dp[i - 1][j] if j ≠ t[i]
// 壓空間, 使用線段樹加速。
// dp[t[i]] = min(dp[t[i]],
//      min(dp[i - 1][k] for k in [s[i] - 1, t[i]]) + cost[i]
// )
fill(dp, dp + E + 1, INF);
seg.init(E + 1, INF);
int idx = 0;
while (idx < N && A[idx].s == 0) {
    dp[A[idx].t] = min(dp[A[idx].t], A[idx].cost);
    seg.update(A[idx].t, A[idx].cost);
    idx++;
}
for (int i = idx; i < N; i++) {
    ll v = min(dp[A[i].t], seg.query(A[i].s - 1, A[i].t + 1) + A[i].
    cost);
    dp[A[i].t] = v;
    seg.update(A[i].t, v);
}
```
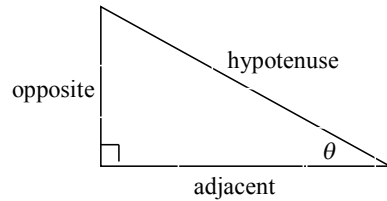
Trig Cheat Sheet

Definition of the Trig Functions

Right triangle definition
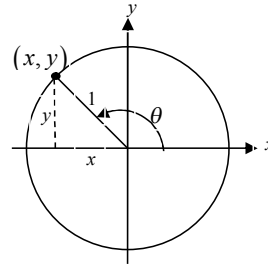
For this definition we assume that
$0 < \theta < \dfrac{\pi}{2}$ or $0° < \theta < 90°$.

$\sin\theta = \dfrac{\text{opposite}}{\text{hypotenuse}}$ $\qquad$ $\csc\theta = \dfrac{\text{hypotenuse}}{\text{opposite}}$

$\cos\theta = \dfrac{\text{adjacent}}{\text{hypotenuse}}$ $\qquad$ $\sec\theta = \dfrac{\text{hypotenuse}}{\text{adjacent}}$

$\tan\theta = \dfrac{\text{opposite}}{\text{adjacent}}$ $\qquad$ $\cot\theta = \dfrac{\text{adjacent}}{\text{opposite}}$

Unit circle definition

For this definition θ is any angle.

$\sin\theta = \dfrac{y}{1} = y$ $\qquad$ $\csc\theta = \dfrac{1}{y}$

$\cos\theta = \dfrac{x}{1} = x$ $\qquad$ $\sec\theta = \dfrac{1}{x}$

$\tan\theta = \dfrac{y}{x}$ $\qquad$ $\cot\theta = \dfrac{x}{y}$

Facts and Properties

Domain

The domain is all the values of θ that can be plugged into the function.

$\sin\theta$, $\quad \theta$ can be any angle

$\cos\theta$, $\quad \theta$ can be any angle

$\tan\theta$, $\quad \theta \neq \left(n+\dfrac{1}{2}\right)\pi, \;\; n = 0, \pm 1, \pm 2, \ldots$

$\csc\theta$, $\quad \theta \neq n\pi, \;\; n = 0, \pm 1, \pm 2, \ldots$

$\sec\theta$, $\quad \theta \neq \left(n+\dfrac{1}{2}\right)\pi, \;\; n = 0, \pm 1, \pm 2, \ldots$

$\cot\theta$, $\quad \theta \neq n\pi, \;\; n = 0, \pm 1, \pm 2, \ldots$

Range

The range is all possible values to get out of the function.

$-1 \le \sin\theta \le 1$ $\qquad$ $\csc\theta \ge 1$ and $\csc\theta \le -1$

$-1 \le \cos\theta \le 1$ $\qquad$ $\sec\theta \ge 1$ and $\sec\theta \le -1$

$-\infty < \tan\theta < \infty$ $\qquad$ $-\infty < \cot\theta < \infty$

Period

The period of a function is the number, T, such that $f(\theta + T) = f(\theta)$. So, if ω is a fixed number and θ is any angle we have the following periods.

$\sin(\omega\theta) \;\rightarrow\; T = \dfrac{2\pi}{\omega}$

$\cos(\omega\theta) \;\rightarrow\; T = \dfrac{2\pi}{\omega}$

$\tan(\omega\theta) \;\rightarrow\; T = \dfrac{\pi}{\omega}$

$\csc(\omega\theta) \;\rightarrow\; T = \dfrac{2\pi}{\omega}$

$\sec(\omega\theta) \;\rightarrow\; T = \dfrac{2\pi}{\omega}$

$\cot(\omega\theta) \;\rightarrow\; T = \dfrac{\pi}{\omega}$

Formulas and Identities

Tangent and Cotangent Identities

$\tan\theta = \dfrac{\sin\theta}{\cos\theta}$ $\qquad$ $\cot\theta = \dfrac{\cos\theta}{\sin\theta}$

Reciprocal Identities

$\csc\theta = \dfrac{1}{\sin\theta}$ $\qquad$ $\sin\theta = \dfrac{1}{\csc\theta}$

$\sec\theta = \dfrac{1}{\cos\theta}$ $\qquad$ $\cos\theta = \dfrac{1}{\sec\theta}$

$\cot\theta = \dfrac{1}{\tan\theta}$ $\qquad$ $\tan\theta = \dfrac{1}{\cot\theta}$

Pythagorean Identities

$\sin^2\theta + \cos^2\theta = 1$

$\tan^2\theta + 1 = \sec^2\theta$

$1 + \cot^2\theta = \csc^2\theta$

Even/Odd Formulas

$\sin(-\theta) = -\sin\theta$ $\qquad$ $\csc(-\theta) = -\csc\theta$

$\cos(-\theta) = \cos\theta$ $\qquad$ $\sec(-\theta) = \sec\theta$

$\tan(-\theta) = -\tan\theta$ $\qquad$ $\cot(-\theta) = -\cot\theta$

Periodic Formulas

If n is an integer.

$\sin(\theta + 2\pi n) = \sin\theta$ $\qquad$ $\csc(\theta + 2\pi n) = \csc\theta$

$\cos(\theta + 2\pi n) = \cos\theta$ $\qquad$ $\sec(\theta + 2\pi n) = \sec\theta$

$\tan(\theta + \pi n) = \tan\theta$ $\qquad$ $\cot(\theta + \pi n) = \cot\theta$

Double Angle Formulas

$\sin(2\theta) = 2\sin\theta\cos\theta$

$\cos(2\theta) = \cos^2\theta - \sin^2\theta$

$\qquad\quad = 2\cos^2\theta - 1$

$\qquad\quad = 1 - 2\sin^2\theta$

$\tan(2\theta) = \dfrac{2\tan\theta}{1 - \tan^2\theta}$

Degrees to Radians Formulas

If x is an angle in degrees and t is an angle in radians then

$\dfrac{\pi}{180} = \dfrac{t}{x}$ $\;\Rightarrow\;$ $t = \dfrac{\pi x}{180}$ $\quad$ and $\quad$ $x = \dfrac{180t}{\pi}$

Half Angle Formulas $\qquad$ (alternate form)

$\sin\dfrac{\theta}{2} = \pm\sqrt{\dfrac{1 - \cos\theta}{2}}$ $\qquad$ $\sin^2\theta = \dfrac{1}{2}\big(1 - \cos(2\theta)\big)$

$\cos\dfrac{\theta}{2} = \pm\sqrt{\dfrac{1 + \cos\theta}{2}}$ $\qquad$ $\cos^2\theta = \dfrac{1}{2}\big(1 + \cos(2\theta)\big)$

$\tan\dfrac{\theta}{2} = \pm\sqrt{\dfrac{1 - \cos\theta}{1 + \cos\theta}}$ $\qquad$ $\tan^2\theta = \dfrac{1 - \cos(2\theta)}{1 + \cos(2\theta)}$

Sum and Difference Formulas

$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$

$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$

$\tan(\alpha \pm \beta) = \dfrac{\tan\alpha \pm \tan\beta}{1 \mp \tan\alpha\tan\beta}$

Product to Sum Formulas

$\sin\alpha\sin\beta = \dfrac{1}{2}\big[\cos(\alpha - \beta) - \cos(\alpha + \beta)\big]$

$\cos\alpha\cos\beta = \dfrac{1}{2}\big[\cos(\alpha - \beta) + \cos(\alpha + \beta)\big]$

$\sin\alpha\cos\beta = \dfrac{1}{2}\big[\sin(\alpha + \beta) + \sin(\alpha - \beta)\big]$

$\cos\alpha\sin\beta = \dfrac{1}{2}\big[\sin(\alpha + \beta) - \sin(\alpha - \beta)\big]$

Sum to Product Formulas

$\sin\alpha + \sin\beta = 2\sin\left(\dfrac{\alpha + \beta}{2}\right)\cos\left(\dfrac{\alpha - \beta}{2}\right)$

$\sin\alpha - \sin\beta = 2\cos\left(\dfrac{\alpha + \beta}{2}\right)\sin\left(\dfrac{\alpha - \beta}{2}\right)$

$\cos\alpha + \cos\beta = 2\cos\left(\dfrac{\alpha + \beta}{2}\right)\cos\left(\dfrac{\alpha - \beta}{2}\right)$

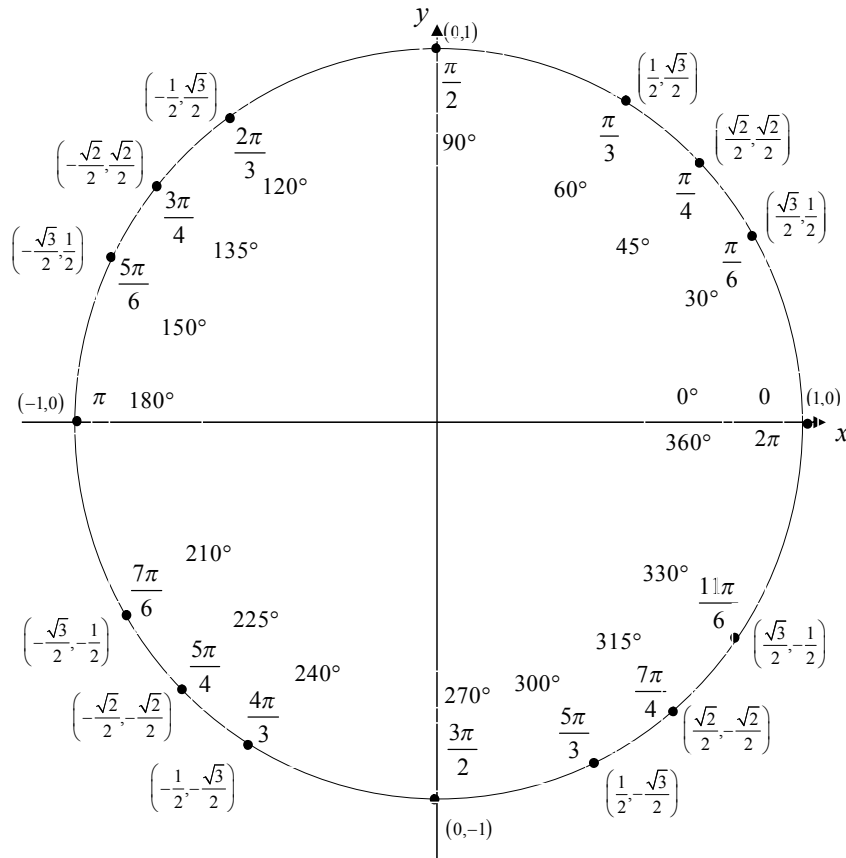$\cos\alpha - \cos\beta = -2\sin\left(\dfrac{\alpha + \beta}{2}\right)\sin\left(\dfrac{\alpha - \beta}{2}\right)$

Cofunction Formulas

$\sin\left(\dfrac{\pi}{2} - \theta\right) = \cos\theta$ $\qquad$ $\cos\left(\dfrac{\pi}{2} - \theta\right) = \sin\theta$

$\csc\left(\dfrac{\pi}{2} - \theta\right) = \sec\theta$ $\qquad$ $\sec\left(\dfrac{\pi}{2} - \theta\right) = \csc\theta$

$\tan\left(\dfrac{\pi}{2} - \theta\right) = \cot\theta$ $\qquad$ $\cot\left(\dfrac{\pi}{2} - \theta\right) = \tan\theta$

Unit Circle

For any ordered pair on the unit circle (x, y) : $\cos\theta = x$ and $\sin\theta = y$

Example

$$\cos\left(\frac{5\pi}{3}\right) = \frac{1}{2} \qquad \sin\left(\frac{5\pi}{3}\right) = -\frac{\sqrt{3}}{2}$$

Inverse Trig Functions

Definition

$y = \sin^{-1} x$ is equivalent to $x = \sin y$

$y = \cos^{-1} x$ is equivalent to $x = \cos y$

$y = \tan^{-1} x$ is equivalent to $x = \tan y$

Domain and Range

Function	Domain	Range
$y = \sin^{-1} x$	$-1 \le x \le 1$	$-\frac{\pi}{2} \le y \le \frac{\pi}{2}$
$y = \cos^{-1} x$	$-1 \le x \le 1$	$0 \le y \le \pi$
$y = \tan^{-1} x$	$-\infty < x < \infty$	$-\frac{\pi}{2} < y < \frac{\pi}{2}$

Inverse Properties

$$\cos\left(\cos^{-1}(x)\right) = x \qquad \cos^{-1}\left(\cos(\theta)\right) = \theta$$

$$\sin\left(\sin^{-1}(x)\right) = x \qquad \sin^{-1}\left(\sin(\theta)\right) = \theta$$

$$\tan\left(\tan^{-1}(x)\right) = x \qquad \tan^{-1}\left(\tan(\theta)\right) = \theta$$
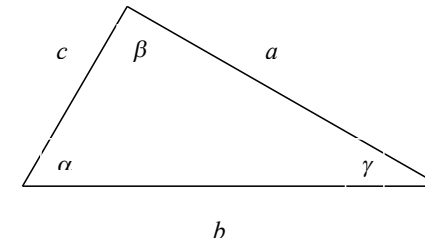
Alternate Notation

$$\sin^{-1} x = \arcsin x$$

$$\cos^{-1} x = \arccos x$$

$$\tan^{-1} x = \arctan x$$

Law of Sines, Cosines and Tangents

Law of Sines

$$\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c}$$

Law of Cosines

$$a^2 = b^2 + c^2 - 2bc\cos\alpha$$

$$b^2 = a^2 + c^2 - 2ac\cos\beta$$

$$c^2 = a^2 + b^2 - 2ab\cos\gamma$$

Mollweide's Formula

$$\frac{a+b}{c} = \frac{\cos\frac{1}{2}(\alpha - \beta)}{\sin\frac{1}{2}\gamma}$$

Law of Tangents

$$\frac{a-b}{a+b} = \frac{\tan\frac{1}{2}(\alpha - \beta)}{\tan\frac{1}{2}(\alpha + \beta)}$$

$$\frac{b-c}{b+c} = \frac{\tan\frac{1}{2}(\beta - \gamma)}{\tan\frac{1}{2}(\beta + \gamma)}$$

$$\frac{a-c}{a+c} = \frac{\tan\frac{1}{2}(\alpha - \gamma)}{\tan\frac{1}{2}(\alpha + \gamma)}$$