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

```
#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

```
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()
        {
```

```java
            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

2 System Testing

1. Setup bashrc and vimrc
2. Look for compilation parameter and code it into bashrc
3. Test if c++ and java templates work properly on local and judge machine
4. Test "divide by 0" → RE/TLE?
5. Test stack size

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
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

4 Useful code

4.1 Leap year

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

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

```cpp
ll fast_pow(ll base, ll exp, ll mod)
{
    if (exp == 0)
        return 1LL;
    ll res = 1;
    while (exp > 0) {
        if (exp & 1) {
            res = ((res % mod) * (base % mod)) % mod;
        }
        exp >>= 1;
        base = (base * base) % mod;
    }
    return res;
}
```

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

注意負數的 case! C++ 是看被除數決定正負號的。

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

4.4 Extended Euclidean Algorithm

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 $gcd(a, b)$.

```cpp
ll ext_gcd(ll a, ll b, ll &x, ll &y)
{
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }

    ll x1, y1;
    ll gcd = ext_gcd(b % a, a, x1, y1);

    x = y1 - (b / a) * x1;
    y = x1;

    return gcd;
}
```

4.5 Mod Inverse

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

Case 2 m *is prime*: $a^{m-2} \equiv a^{-1} mod\ m$ (use Fermat's little theorem)

4.6 Prime Generator

```
bool is_prime[N];
vector<ll> primes;
void init()
{
    fill(is_prime, is_prime + N, true);
    for (int i = 2; i < N; i++) {
        if (is_prime[i] == true) {
            primes.push_back(i);
            for (int j = i * i; j < N; j += i)
                is_prime[j] = false;
        }
    }
}
```

4.7 Binomial Coefficient

```
int binomialCoeff(int n, int k)
{
    int res = 1;

    if ( k > n - k ) // Since C(n, k) = C(n, n-k)
        k = n - k;

    for (int i = 0; i < k; ++i) // n...n-k / 1...k
    {
        res *= (n - i);
        res /= (i + 1);
    }

    return res;
}
```

4.8 scanf/printf reference

4.9 STL quick reference

4.9.1 Map

```
map<T1, T2> m; // iterable
void clear();
void erase(T1 key);
it find(T1 key); // <key, val>
void insert(pair<T1, T2> P);
T2 &[](T1 key); // if key not in map, new key will be inserted with default val
it lower_bound(T1 key); // = m.end() if not found, *it = <key, val>
it upper_bound(T1 key); // = m.end() if not found, *it = <key, val>
```

4.9.2 Set

```
set<T> s; // iterable
void clear();
size_t count(T val); // number of val in set
void erase(T val);
it find(T val); // = s.end() if not found
void insert(T val);
it lower_bound(T val); // = s.end() if not found, *it = <key, val>
it upper_bound(T val); // = s.end() if not found, *it = <key, val>
```

4.9.3 Algorithm

```
// return if i is smaller than j
comp = [&](const T &i, const T &j) -> bool;
vector<T> v;
bool any_of(v.begin(), v.end(), [&](const T &i) -> bool);
bool all_of(v.begin(), v.end(), [&](const T &i) -> bool);
void copy(inp.begin(), in.end(), out.begin());
int count(v.begin(), v.end(), int val); // number of val in v
it unique(v.begin(), v.end());          // it - v.begin() = size
// after calling, v[nth] will be n-th smallest elem in v
void nth_element(v.begin(), nth_it, bin_comp);
void merge(in1.begin(), in1.end(), in2.begin(), in2.end(), out.begin(), comp);
// include union, intersection, difference, symmetric_difference(xor)
void set_union(in1.begin(), in1.end(), in2.begin(), in2.end(), out.begin(), comp);
bool next_permutation(v.begin(), v.end());
// v1, v2 need sorted already, whether v1 includes v2
bool inclues(v1.begin(), v1.end(), v2.begin(), v2.end());
it find(v.begin(), v.end(), T val); // = v.end() if not found
it search(v1.begin(), v1.end(), v2.begin(), v2.end());
it lower_bound(v.begin(), v.end(), T val);
it upper_bound(v.begin(), v.end(), T val);
bool binary_search(v.begin(), v.end(), T val); // exist in v ?
void sort(v.begin(), v.end(), comp);
void stable_sort(v.begin(), v.end(), comp);
```

4.9.4 String

4.9.5 Priority Queue

```
bool cmp(ii a, ii b)
{
    if(a.first == b.first)
        return a.second > b.second;
    return b.first > a.first;
}

priority_queue< ii, vector<ii>, function<bool(ii, ii)> > pq(cmp);
```

5 Search

5.1 Ternary Search

5.2 折半完全列舉

5.3 Two-pointer 爬行法

6 Basic data structure

6.1 1D BIT

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

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

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

6.2 2D BIT

```
// 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];

ll sum(int a, int b) {
    ll s = 0;
    for (int i = a; i > 0; i -= (i & -i))
        for (int j = b; j > 0; j -= (j & -j))
            s += bit[i][j];
    return s;
}

void add(int a, int b, ll x) {
    // MAX_N, MAX_M 須適時調整!
    for (int i = a; i <= MAX_N; i += (i & -i))
        for (int j = b; j <= MAX_M; j += (j & -j))
            bit[i][j] += x;
}
```

6.3 Union Find

```
#define N 20000 // 記得改
struct UFDS {
    int par[N];

    void init() {
        memset(par, -1, sizeof(par));
    }

    int root(int x) {
        return par[x] < 0 ? x : par[x] = root(par[x]);
    }

    void merge(int x, int y) {
        x = root(x);
        y = root(y);

        if (x != y) {
            if (par[x] > par[y])
                swap(x, y);
            par[x] += par[y];
            par[y] = x;
        }
    }
}
```

6.4 Segment Tree

6.5 Sparse Table

```
struct {
    int sp[MAX_LOG_N][MAX_N]; // MAX_LOG_N = ceil(lg(MAX_N))

    void build(int inp[], int n) {
        for (int j = 0; j < n; j++) {
            sp[0][j] = inp[j];
        }

        for (int i = 1; (1 << i) <= n; i++)
            for (int j = 0; j + (1 << i) <= n; j++)
                sp[i][j] =
                    min(sp[i - 1][j], sp[i - 1][j + (1 << (i - 1))]);
    }

    int query(int l, int r) { // [l, r)
        int k = floor(log2(r - l));

        return min(sp[k][l], sp[k][r - (1 << k)]);
    }
} sptb;
```

7 Dynamic Programming

8 Tree

8.1 LCA

8.2 Tree Centroid

8.3 Treap

9 Graph

9.1 Articulation point / edge

9.2 CC

9.2.1 BCC vertex

9.2.2 BCC edge

9.2.3 SCC

9.3 Shortest Path

9.3.1 Dijkatra

9.3.2 Dijkatra (next-to-shortest path)

9.3.3 SPFA

9.3.4 Bellman-Ford

9.3.5 Floyd-Warshall

9.4 MST

9.4.1 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

9.4.2 Prim

9.5 Flow

9.5.1 Max Flow (Dinic)

9.5.2 Min-Cut

9.5.3 Min Cost Max Flow

9.5.4 Maximum Bipartite Graph

10 String

10.1 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 $0xdefaced$

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

typedef long long ll;

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;
}
```

10.2 KMP

```
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++;
```

```cpp

        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++;
            }
        }
    }

    return res;
}
```

10.3 Z Algorithm

10.4 Trie

```cpp
#define N 600010
struct node {
    int child[26];
    bool ending;
} trie[N];

/*
root is 0
memset(trie, 0, sizeof(trie));
freeNode = 1;
*/
int freeNode;
void insert(string &str, int pos, int node)
{
    if (pos == (int)str.length()) {
        trie[node].ending = true;
    } else { // find which way to go
        int c = str[pos] - 'a';
        if (trie[node].child[c] == 0) // give a new node
            trie[node].child[c] = freeNode++;
        insert(str, pos + 1, trie[node].child[c]);
    }
}
```

10.5 Suffix Array

11 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

11.1 EPS

$a > b \to a - b > 0 \to a - b > EPS$ (stands for positive)
$a \geq b \to a - b \geq 0 \to a - b > -EPS$ (stands for positive or zero)

11.2 Template

```cpp
// if the points are given in doubles form, change the code accordingly

typedef long long ll;

typedef pair<ll, ll> pt;  // points are stored using long long
typedef pair<pt, pt> seg; // segments are a pair of points

#define x first
#define y second

#define EPS 1e-9

pt operator+(pt a, pt b)
{
    return pt(a.x + b.x, a.y + b.y);
}

pt operator-(pt a, pt b)
{
    return pt(a.x - b.x, a.y - b.y);
}

pt operator*(pt a, int d)
{
    return pt(a.x * d, a.y * d);
}

ll cross(pt a, pt b)
{
    return a.x * b.y - a.y * b.x;
}

int ccw(pt a, pt b, pt c)
{
    ll res = cross(b - a, c - a);
    if (res > 0) // left turn
        return 1;
    else if (res == 0) // straight
        return 0;
    else // right turn
```

```cpp
        return -1;
}

double dist(pt a, pt b)
{
    double dx = a.x - b.x;
    double dy = a.y - b.y;
    return sqrt(dx * dx + dy * dy);
}

bool zero(double x)
{
    return fabs(x) <= EPS;
}

bool overlap(seg a, seg b)
{
    return ccw(a.x, a.y, b.x) == 0 && ccw(a.x, a.y, b.y) == 0;
}

bool intersect(seg a, seg b)
{
    if (overlap(a, b) == true) { // non-proper intersection
        double d = 0;
        d = max(d, dist(a.x, a.y));
        d = max(d, dist(a.x, b.x));
        d = max(d, dist(a.x, b.y));
        d = max(d, dist(a.y, b.x));
        d = max(d, dist(a.y, b.y));
        d = max(d, dist(b.x, b.y));

        // d > dist(a.x, a.y) + dist(b.x, b.y)
        if (d - (dist(a.x, a.y) + dist(b.x, b.y)) > EPS)
            return false;
        return true;
    }
    //                     |
    // Equal sign for -----| case
    // non qeual sign => proper intersection
    if (ccw(a.x, a.y, b.x) * ccw(a.x, a.y, b.y) <= 0 &&
        ccw(b.x, b.y, a.x) * ccw(b.x, b.y, a.y) <= 0)
        return true;
    return false;
}

double area(vector<pt> pts)
{
    double res = 0;
    int n = pts.size();
    for (int i = 0; i < n; i++)
        res += (pts[i].y + pts[(i + 1) % n].y) * (pts[(i + 1) % n].x - pts[i].x);
    return res / 2.0;
}

vector<pt> halfHull(vector<pt> &points)
{
    vector<pt> res;

    for (int i = 0; i < (int)points.size(); i++) {
        while ((int)res.size() >= 2 &&
                ccw(res[res.size() - 2], res[res.size() - 1], points[i]) < 0)
            res.pop_back(); // res.size() - 2 can't be assign before size() >= 2
        // check, bitch

        res.push_back(points[i]);
    }

    return res;
}

vector<pt> convexHull(vector<pt> &points)
{
    vector<pt> upper, lower;

    // make upper hull
    sort(points.begin(), points.end());

    upper = halfHull(points);
    // make lower hull
    reverse(points.begin(), points.end());
    lower = halfHull(points);

    // merge hulls
    if ((int)upper.size() > 0) // yes sir~
        upper.pop_back();
    if ((int)lower.size() > 0)
        lower.pop_back();

    vector<pt> res(upper.begin(), upper.end());
    res.insert(res.end(), lower.begin(), lower.end());

    return res;
}

bool completelyInside(vector<pt> &outer, vector<pt> &inner)
{
    int even = 0, odd = 0;
    for (int i = 0; i < (int)inner.size(); i++) {
        // y = slope * x + offset
        int cntIntersection = 0;
        ll slope = rand() % INT_MAX + 1;
        ll offset = inner[i].y - slope * inner[i].x;

        ll farx = 111111 * (slope >= 0 ? 1 : -1);
        ll fary = farx * slope + offset;
        seg a = seg(pt(inner[i].x, inner[i].y), pt(farx, fary));
        for (int j = 0; j < (int)outer.size(); j++) {
            seg b = seg(outer[j], outer[(j + 1) % (int)outer.size()]);
```

```
            if ((b.x.x * slope + offset == b.x.y) ||
                (b.y.x * slope + offset == b.y.y)) { // on-line
                i--;
                break;
            }

            if (intersect(a, b) == true)
                cntIntersection++;
        }

        if (cntIntersection % 2 == 0) // outside
            even++;
        else
            odd++;
    }

    return odd == (int)inner.size();
}

// srand(time(NULL))
// rand()
```

12 Math

12.1 Euclid's formula

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

12.2 Difference between two consecutive numbers' square is odd

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

12.3 Summation

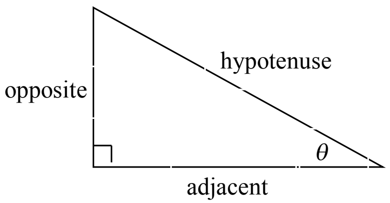
$\sum_{k=1}^{n} 1 = n$
$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$

Trig Cheat Sheet

Definition of the Trig Functions

Right triangle definition
For this definition we assume that

$0 < \theta < \frac{\pi}{2}$ or $0° < \theta < 90°$.

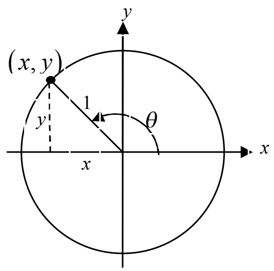


$$\sin\theta = \frac{\text{opposite}}{\text{hypotenuse}} \qquad \csc\theta = \frac{\text{hypotenuse}}{\text{opposite}}$$

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

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

Unit circle definition
For this definition θ is any angle.



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

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

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

Facts and Properties

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

$\sin\theta$, θ can be any angle
$\cos\theta$, θ can be any angle
$\tan\theta$, $\theta \neq \left(n+\frac{1}{2}\right)\pi,\quad n = 0, \pm 1, \pm 2, \ldots$
$\csc\theta$, $\theta \neq n\pi,\quad n = 0, \pm 1, \pm 2, \ldots$
$\sec\theta$, $\theta \neq \left(n+\frac{1}{2}\right)\pi,\quad n = 0, \pm 1, \pm 2, \ldots$
$\cot\theta$, $\theta \neq n\pi,\quad n = 0, \pm 1, \pm 2, \ldots$

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.

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

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

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

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

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

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

Formulas and Identities

Tangent and Cotangent Identities

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

Reciprocal Identities

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

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

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

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

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

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

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

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

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

Degrees to Radians Formulas
If x is an angle in degrees and t is an angle in radians then

$$\frac{\pi}{180} = \frac{t}{x} \quad \Rightarrow \quad t = \frac{\pi x}{180} \quad \text{and} \quad x = \frac{180t}{\pi}$$

Half Angle Formulas (alternate form)

$$\sin\frac{\theta}{2} = \pm\sqrt{\frac{1-\cos\theta}{2}} \qquad \sin^2\theta = \frac{1}{2}\left(1-\cos(2\theta)\right)$$

$$\cos\frac{\theta}{2} = \pm\sqrt{\frac{1+\cos\theta}{2}} \qquad \cos^2\theta = \frac{1}{2}\left(1+\cos(2\theta)\right)$$

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

Sum and Difference Formulas

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

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

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

Product to Sum Formulas

$$\sin\alpha\sin\beta = \frac{1}{2}\left[\cos(\alpha-\beta) - \cos(\alpha+\beta)\right]$$

$$\cos\alpha\cos\beta = \frac{1}{2}\left[\cos(\alpha-\beta) + \cos(\alpha+\beta)\right]$$

$$\sin\alpha\cos\beta = \frac{1}{2}\left[\sin(\alpha+\beta) + \sin(\alpha-\beta)\right]$$

$$\cos\alpha\sin\beta = \frac{1}{2}\left[\sin(\alpha+\beta) - \sin(\alpha-\beta)\right]$$

Sum to Product Formulas

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

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

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

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

Cofunction Formulas

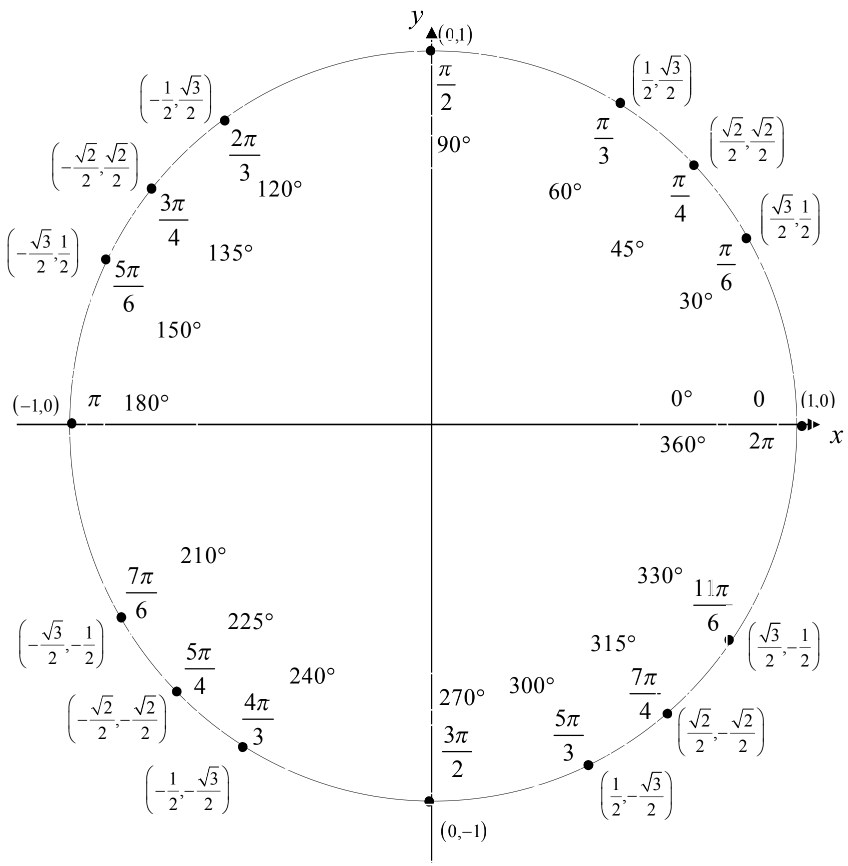
$$\sin\left(\frac{\pi}{2}-\theta\right) = \cos\theta \qquad \cos\left(\frac{\pi}{2}-\theta\right) = \sin\theta$$

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

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

© 2005 Paul Dawkins

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

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

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

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

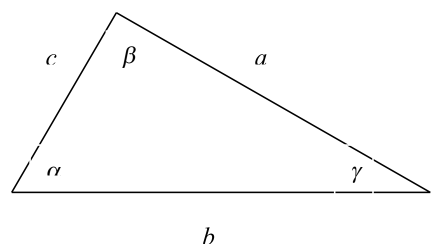
Alternate Notation

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

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

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

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

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

© 2005 Paul Dawkins