

Contents

1 Contest Setup	1	12 Matrix	11
1.1 vimrc	1	12.1 高斯消去法	11
1.2 bashrc	2	12.2 高斯喬登	11
1.3 Grep Error and Warnings	2	13 Geometry	11
1.4 C++ template	2	13.1 EPS	11
1.5 Java template	2	13.2 Template	11
1.5.1 Java Issues	2	14 Math	12
2 System Testing	2	14.1 Euclid's formula (Pythagorean Triples)	12
3 Reminder	3	14.2 Difference between two consecutive numbers' square is odd	13
4 Topic list	3	14.3 Summation	13
5 Useful code	3	14.4 FFT	13
5.1 Leap year	3	14.5 Combination	14
5.2 Fast Exponentiation $O(\log(exp))$	3	14.5.1 Pascal triangle	14
5.3 Mod Inverse	3	14.5.2 線性	14
5.4 GCD $O(\log(a+b))$	3	14.6 重複組合	14
5.5 Extended Euclidean Algorithm GCD $O(\log(a+b))$	3	14.7 Chinese remainder theorem	14
5.6 Prime Generator	3	14.8 2-Circle relations	14
5.7 C++ Reference	4	14.9 Fun Facts	14
5.7.1 scanf/printf reference	4	15 Dynamic Programming - Problems collection	17
5.7.2 Map	4		
5.7.3 Set	4		
5.7.4 Algorithm	4		
5.7.5 String	4		
5.7.6 Priority Queue	4		
6 Search	4		
6.1 Ternary Search	4		
6.2 折半完全列舉	4		
6.3 Two-pointer 爬行法 (右跑左追)	4		
7 Basic data structure	4		
7.1 1D BIT	4		
7.2 2D BIT	4		
7.3 Union Find	5		
7.4 Segment Tree	5		
7.5 Sparse Table	6		
8 Tree	6		
8.1 LCA	6		
8.2 Tree Centroid	6		
8.3 Treap	6		
8.4 Merge Tree	6		
9 Graph	6		
9.1 Articulation point / edge	6		
9.2 CC	6		
9.2.1 BCC vertex	6		
9.2.2 BCC edge	6		
9.2.3 SCC	6		
9.3 Shortest Path	6		
9.3.1 Dijkstra (next-to-shortest path)	6		
9.3.2 SPFA	7		
9.3.3 Bellman-Ford $O(VE)$	7		
9.3.4 Floyd-Warshall $O(V^3)$	7		
9.4 MST	7		
9.4.1 Kruskal	7		
9.4.2 Prim	8		
10 Flow	8		
10.1 Max Flow (Dinic)	8		
10.2 Min Cost Flow	8		
10.3 Bipartite Matching	9		
11 String	9		
11.1 Rolling Hash	9		
11.2 KMP	10		
11.3 Z Algorithm	10		
11.4 Trie	10		

```

1 set number      " Show line numbers
2 set mouse=a     " Enable inaction via mouse
3 set showmatch   " Highlight matching brace
4 set cursorline  " Show underline
5 set cursorcolumn " highlight vertical column
6
7 filetype on "enable file detection
8 syntax on   "syntax highlight
9
10 set autoindent " Auto-indent new lines
11 set shiftwidth=4 " Number of auto-indent spaces
12 set smartindent " Enable smart-indent
13 set smarttab    " Enable smart-tabs
14 set tabstop=4   " Number of spaces per Tab
15
16 " -----Optional-----
17
18 set undolevels=10000 " Number of undo levels
19 set scrolloff=5     " Auto scroll
20
21 set hlsearch      " Highlight all search results
22 set smartcase     " Enable smart-case search
23 set ignorecase    " Always case-insensitive
24 set incsearch     " Searches for strings incrementally
25
26 highlight Comment ctermfg=cyan
27 set showmode
28
29 set encoding=utf-8
30 set fileencoding=utf-8
31 set scriptencoding=utf-8

```

1.2 bashrc

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

1.3 Grep Error and Warnings

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

1.4 C++ template

```
1| #include <bits/stdc++.h>
2|
3| using namespace std;
4|
5| typedef long long int ll;
6| typedef pair<int, int> ii;
7|
8| int main()
9| {
10|     return 0;
11| }
```

1.5 Java template

```
1| import java.io.*;
2| import java.util.*;
3|
4| public class Main
5| {
6|     public static void main(String[] args)
7|     {
8|         MyScanner sc = new MyScanner();
9|         out = new PrintWriter(new BufferedOutputStream(System.out));
10|         // Start writing your solution here.
11|
12|         // Stop writing your solution here.
13|         out.close();
14|     }
15|
16|     public static PrintWriter out;
17|
18|     public static class MyScanner
19|     {
20|         BufferedReader br;
21|         StringTokenizer st;
22|
23|         public MyScanner()
24|         {
25|             br = new BufferedReader(new InputStreamReader(System.in));
26|         }
27|
28|         boolean hasNext()
29|         {
```

```
30|         while (st == null || !st.hasMoreElements()) {
31|             try {
32|                 st = new StringTokenizer(br.readLine());
33|             } catch (Exception e) {
34|                 return false;
35|             }
36|         }
37|         return true;
38|     }
39|
40|     String next()
41|     {
42|         if (hasNext())
43|             return st.nextToken();
44|         return null;
45|     }
46|
47|     int nextInt()
48|     {
49|         return Integer.parseInt(next());
50|     }
51|
52|     long nextLong()
53|     {
54|         return Long.parseLong(next());
55|     }
56|
57|     double nextDouble()
58|     {
59|         return Double.parseDouble(next());
60|     }
61|
62|     String nextLine()
63|     {
64|         String str = "";
65|         try {
66|             str = br.readLine();
67|         } catch (IOException e) {
68|             e.printStackTrace();
69|         }
70|         return str;
71|     }
72| }
73| }
```

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

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 heap 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 \leftrightarrow Binary Search
10. Directed graph connectivity \rightarrow DFS. Undirected graph \rightarrow 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* \ll 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

4 Topic list

1. enumeration
2. greedy
3. sorting, topological sort
4. binary search
5. 離散化
6. 矩陣快速幂
7. Pigeonhole
8. DFS 轉換成 RMQ

5 Useful code

5.1 Leap year

```
1 || 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 $a^{m-1} \equiv 1 \pmod{m}$

```
1 ll fast_pow(ll a, ll b, ll M) {
2     ll ans = 1;
3     ll base = a % M;
4     while (b) {
5         if (b & 1)
6             ans = ans * base % M;
7         base = base * base % M;
8         b >>= 1;
9     }
10    return ans;
11 }
```

5.3 Mod Inverse

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

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

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。

```
1 ll gcd(ll a, ll b)
2 {
3     return b == 0 ? a : gcd(b, a % b);
4 }
```

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 $\gcd(a, b)$.

```
1 ll extgcd(ll a, ll b, ll& x, ll& y) {
2     if (b == 0) {
3         x = 1;
4         y = 0;
5         return a;
6     }
7     else {
8         ll d = extgcd(b, a % b, y, x);
9         y -= (a / b) * x;
10        return d;
11    }
12 }
```

5.6 Prime Generator

```
1 const ll MAX_NUM = 1e6; // 要是合數
2 bool is_prime[MAX_NUM];
3 vector<ll> primes;
4
5 void init_primes() {
6     fill(is_prime, is_prime + MAX_NUM, true);
7     is_prime[0] = is_prime[1] = false;
8     for (ll i = 2; i < MAX_NUM; i++) {
9         if (is_prime[i]) {
10            primes.push_back(i);
11            for (ll j = i * i; j < MAX_NUM; j += i)
12                is_prime[j] = false;
13        }
14    }
15 }
```

5.7 C++ Reference

5.7.1 scanf/printf reference

5.7.2 Map

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

5.7.3 Set

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

5.7.4 Algorithm

```
1 // return if i is smaller than j
2 comp = [&](const T &i, const T &j) -> bool;
3 vector<T> v;
4 bool any_of(v.begin(), v.end(), [&](const T &i) -> bool);
5 bool all_of(v.begin(), v.end(), [&](const T &i) -> bool);
6 void copy(inp.begin(), inp.end(), out.begin());
7 int count(v.begin(), v.end(), int val); // number of val in v
8 it unique(v.begin(), v.end()); // it - v.begin() = size
    // after calling, v[nth] will be n-th smallest elem in v
9 void nth_element(v.begin(), nth_it, bin_comp);
10 void merge(in1.begin(), in1.end(), in2.begin(), in2.end(), out.begin(),
    comp);
11 // include union, intersection, difference, symmetric_difference(xor)
12 void set_union(in1.begin(), in1.end(), in2.begin(), in2.end(), out.
    begin(), comp);
13 bool next_permutation(v.begin(), v.end());
14 // v1, v2 need sorted already, whether v1 includes v2
15 bool inclues(v1.begin(), v1.end(), v2.begin(), v2.end());
16 it find(v.begin(), v.end(), T val); // = v.end() if not found
17 it search(v1.begin(), v1.end(), v2.begin(), v2.end());
18 it lower_bound(v.begin(), v.end(), T val);
19 it upper_bound(v.begin(), v.end(), T val);
20 bool binary_search(v.begin(), v.end(), T val); // exist in v ?
21 void sort(v.begin(), v.end(), comp);
22 void stable_sort(v.begin(), v.end(), comp);
```

5.7.5 String

5.7.6 Priority Queue

```
1 bool cmp(ii a, ii b)
2 {
3     if(a.first == b.first)
4         return a.second > b.second;
5     return b.first > a.first;
6 }
7
8 priority_queue< ii, vector<ii>, function<bool(ii, ii)> > pq(cmp);
```

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)

7 Basic data structure

7.1 1D BIT

```
1 // BIT is 1-based
2 const int MAX_N = 20000; //這個記得改!
3 ll bit[MAX_N + 1];
4
5 ll sum(int i) {
6     int s = 0;
7     while (i > 0) {
8         s += bit[i];
9         i -= (i & -i);
10    }
11    return s;
12 }
13
14 void add(int i, ll x) {
15     while (i <= MAX_N) {
16         bit[i] += x;
17         i += (i & -i);
18    }
19 }
```

7.2 2D BIT

```
1 // BIT is 1-based
2 const int MAX_N = 20000, MAX_M = 20000; //這個記得改!
3 ll bit[MAX_N + 1][MAX_M + 1];
4
```

```

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

```

```

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

8.2 Tree Centroid

8.3 Treap

8.4 Merge Tree

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

Time complexity notations: V = vertex, E = edge

9.3.1 Dijkstra

```

1 #define st first
2 #define nd second
3
4 typedef pair<int, int> pii; // <d, v>
5 struct Edge {
6     int to, w;
7 };
8
9 const int MAX_V = ...;
10 const int INF = 0x3f3f3f3f;
11
12 int V, S; // V, Source
13 vector<Edge> g[MAX_V];
14 int d[MAX_V];
15 int cnt[MAX_V];
16
17 bool spfa() { // 回傳有無負環
18     fill(d, d + V, INF);
19     fill(cnt, cnt + V, 0);
20     priority_queue<pii, vector<pii>, greater<pii> > pq;
21
22     d[S] = 0;
23     pq.push(pii(0, S));
24     cnt[S] = 1;
25
26     while (!pq.empty()) {
27         pii top = pq.top(); pq.pop();
28         int u = top.nd;
29

```

```

30     if (d[u] < top.st) continue;
31
32     // for (const Edge& e : g[u]) {
33     for (size_t i = 0; i < g[u].size(); i++) {
34         const Edge& e = g[u][i];
35         if (d[e.to] > d[u] + e.w) {
36             d[e.to] = d[u] + e.w;
37             pq.push(pii(d[e.to], e.to));
38
39             cnt[e.to]++;
40             if (cnt[e.to] >= V)
41                 return true;
42         }
43     }
44 }
45
46 return false;
47 }

```

9.3.2 Dijkstra (next-to-shortest path)

```

1 struct Edge {
2     int to, cost;
3 };
4
5 typedef pair<int, int> P; // <d, v>
6 const int INF = 0x3f3f3f3f;
7
8 int N, R;
9 vector<Edge> g[5000];
10
11 int d[5000];
12 int sd[5000];
13
14 int solve() {
15     fill(d, d + N, INF);
16     fill(sd, sd + N, INF);
17     priority_queue< P, vector<P>, greater<P> > pq;
18
19     d[0] = 0;
20     pq.push(P(0, 0));
21
22     while (!pq.empty()) {
23         P p = pq.top(); pq.pop();
24         int v = p.second;
25
26         if (sd[v] < p.first) // 比次短距離還大，沒用，跳過
27             continue;
28
29         for (size_t i = 0; i < g[v].size(); i++) {
30             Edge& e = g[v][i];
31             int nd = p.first + e.cost;
32             if (nd < d[e.to]) { // 更新最短距離
33                 swap(d[e.to], nd);
34                 pq.push(P(d[e.to], e.to));
35             }
36         }
37     }
38 }

```

```

36     if (d[e.to] < nd && nd < sd[e.to]) { // 更新次短距離
37         sd[e.to] = nd;
38         pq.push(P(sd[e.to], e.to));
39     }
40 }
41
42
43 return sd[N-1];
44 }

```

9.3.3 SPFA

```

1 typedef pair<int, int> ii;
2 vector< ii > g[N];
3
4 bool SPFA()
5 {
6     vector<ll> d(n, INT_MAX);
7     d[0] = 0; // origin
8
9     queue<int> q;
10    vector<bool> inqueue(n, false);
11    vector<int> cnt(n, 0);
12    q.push(0);
13    inqueue[0] = true;
14    cnt[0]++;
15
16    while(q.empty() == false) {
17        int u = q.front();
18        q.pop();
19        inqueue[u] = false;
20
21        for(auto i : g[u]) {
22            int v = i.first, w = i.second;
23            if(d[u] + w < d[v]) {
24                d[v] = d[u] + w;
25                if(inqueue[v] == false) {
26                    q.push(v);
27                    inqueue[v] = true;
28                    cnt[v]++;
29
30                    if(cnt[v] == n) { // loop!
31                        return true;
32                    }
33                }
34            }
35        }
36    }
37
38    return false;
39 }

```

9.3.4 Bellman-Ford $O(VE)$

```

1 vector<pair<ii, int>> edge; // store graph by edge: ((u, v), w)
2
3 void BellmanFord()
4 {
5     ll d[n]; // n: total nodes
6     fill(d, d + n, INT_MAX);
7     d[0] = 0; // src is 0
8     bool loop = false;
9     for (int i = 0; i < n; i++) {
10         // Do n - 1 times. If the n-th time still has relaxation, loop
11         exists
12         bool hasChange = false;
13         for (int j = 0; j < (int)edge.size(); j++) {
14             int u = edge[j].first.first, v = edge[j].first.second, w =
15             edge[j].second;
16             if (d[u] + w < d[v]) {
17                 hasChange = true;
18                 d[v] = d[u] + w;
19             }
20         }
21         if (i == n - 1 && hasChange == true)
22             loop = true;
23         else if (hasChange == false)
24             break;
25     }
26 }

```

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

The graph is stored using adjacency matrix. The initial state is *diagonal* = 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]);

```

9.4 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

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         }
53     }
54     return 0;
55 }

```



```

56 int max_flow(int s, int t) { // dinic
57     int flow = 0;
58     for (;;) {
59         bfs(s);
60         if (level[t] < 0) return flow;
61         memset(iter, 0, sizeof(iter));
62         int f;
63         while ((f = dfs(s, t, INF)) > 0) {
64             flow += f;
65         }
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     while (!pq.empty()) {
40         pii p = pq.top(); pq.pop();
41         int v = p.nd;
42         if (d[v] < p.st) continue;
43         for (size_t i = 0; i < g[v].size(); i++) {
44             const Edge& e = g[v][i];
45             if (e.cap > 0 && d[e.to] > d[v] + e.cost + h[v] - h[e.
46 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[preve[v]][v].cap);
66
67     // // find match
68     // for (int v = prevv[t]; v != s; v = prevv[preve[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[preve[v]][v];
79         e.cap -= bn;
80         g[v][e.rev].cap += bn;
81     }
82 }
83 return res;
84 }

```

10.3 Bipartite Matching

```

1  const int MAX_V = ...;
2  int V;
3  vector<int> g[MAX_V];
4  int match[MAX_V];

```

```

5 bool used[MAX_V];
6
7 void add_edge(int u, int v) {
8     g[u].push_back(v);
9     g[v].push_back(u);
10 }
11
12 // 回傳有無找到從 v 出發的增廣路徑
13 // (首尾都為未匹配點的交錯路徑)
14 // [待確認] 每次遞迴都找一個未匹配點 v 及匹配點 u
15 bool dfs(int v) {
16     used[v] = true;
17     for (size_t i = 0; i < g[v].size(); i++) {
18         int u = g[v][i], w = match[u];
19         // 尚未配對或可從 w 找到增廣路徑 (即路徑繼續增長)
20         if (w < 0 || (!used[w] && dfs(w))) {
21             // 交錯配對
22             match[v] = u;
23             match[u] = v;
24             return true;
25         }
26     }
27     return false;
28 }
29
30 int bipartite_matching() { // 匈牙利演算法
31     int res = 0;
32     memset(match, -1, sizeof(match));
33     for (int v = 0; v < V; v++) {
34         if (match[v] == -1) {
35             memset(used, false, sizeof(used));
36             if (dfs(v)) {
37                 res++;
38             }
39         }
40     }
41     return res;
42 }

```

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`

```

1 #define N 1000100
2 #define B 137
3 #define M 1000000007
4
5 typedef long long ll;
6
7 char inp[N];
8 int len;
9 ll p[N], h[N];
10

```

```

11 void init()
12 { // build polynomial table and hash value
13     p[0] = 1; // b to the ith power
14     for (int i = 1; i <= len; i++) {
15         h[i] = (h[i - 1] * B % M + inp[i - 1]) % M; // hash value
16         p[i] = p[i - 1] * B % M;
17     }
18 }
19
20 ll get_hash(int l, int r) // [l, r] of the inp string array
21 {
22     return ((h[r + 1] - (h[l] * p[r - l + 1])) % M + M) % M;
23 }

```

11.2 KMP

```

1 void fail()
2 {
3     int len = strlen(pat);
4
5     f[0] = 0;
6     int j = 0;
7     for (int i = 1; i < len; i++) {
8         while (j != 0 && pat[i] != pat[j])
9             j = f[j - 1];
10
11         if (pat[i] == pat[j])
12             j++;
13
14         f[i] = j;
15     }
16 }
17
18 int match()
19 {
20     int res = 0;
21     int j = 0, plen = strlen(pat), tlen = strlen(text);
22
23     for (int i = 0; i < tlen; i++) {
24         while (j != 0 && text[i] != pat[j])
25             j = f[j - 1];
26
27         if (text[i] == pat[j]) {
28             if (j == plen - 1) { // find match
29                 res++;
30                 j = f[j];
31             } else {
32                 j++;
33             }
34         }
35     }
36
37     return res;
38 }

```

11.3 Z Algorithm

11.4 Trie

注意 count 的擺放位置，視題意可以擺在迴圈外

```

1 struct Node {
2     int cnt;
3     Node* nxt[2];
4     Node() {
5         cnt = 0;
6         fill(nxt, nxt + 2, nullptr);
7     }
8 };
9
10 const int MAX_Q = 200000;
11 int Q;
12
13 int NN = 0;
14 Node data[MAX_Q * 30];
15 Node* root = &data[NN++];
16
17 void insert(Node* u, int x) {
18     for (int i = 30; i >= 0; i--) {
19         int t = ((x >> i) & 1);
20         if (u->nxt[t] == nullptr) {
21             u->nxt[t] = &data[NN++];
22         }
23
24         u = u->nxt[t];
25         u->cnt++;
26     }
27 }
28
29 void remove(Node* u, int x) {
30     for (int i = 30; i >= 0; i--) {
31         int t = ((x >> i) & 1);
32         u = u->nxt[t];
33         u->cnt--;
34     }
35 }
36
37 int query(Node* u, int x) {
38     int res = 0;
39     for (int i = 30; i >= 0; i--) {
40         int t = ((x >> i) & 1);
41         // if it is possible to go the another branch
42         // then the result of this bit is 1
43         if (u->nxt[t ^ 1] != nullptr && u->nxt[t ^ 1]->cnt > 0) {
44             u = u->nxt[t ^ 1];
45             res |= (1 << i);
46         }
47         else {
48             u = u->nxt[t];
49         }
50     }
51     return res;
52 }

```

11.5 Suffix Array

12 Matrix

12.1 高斯消去法

12.2 高斯喬登

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

$= 0$: $fabs \leq eps$
 < 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     // Equal sign for -----| case
78     // non equal sign => proper intersection
79     if (ccw(a.x, a.y, b.x) * ccw(a.x, a.y, b.y) <= 0 &&
80         ccw(b.x, b.y, a.x) * ccw(b.x, b.y, a.y) <= 0)
81         return true;
82     return false;
83 }
84
85 double area(vector<pt> pts)
86 {
87     double res = 0;
88     int n = pts.size();
89

```

```

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

```

```

143     ll farx = 111111 * (slope >= 0 ? 1 : -1);
144     ll fary = farx * slope + offset;
145     seg a = seg(pt(inner[i].x, inner[i].y), pt(farx, fary));
146     for (int j = 0; j < (int)outer.size(); j++) {
147         seg b = seg(outer[j], outer[(j + 1) % (int)outer.size()]);
148
149         if ((b.x.x * slope + offset == b.x.y) ||
150             (b.y.x * slope + offset == b.y.y)) { // on-line
151             i--;
152             break;
153         }
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)

$$\begin{aligned}
 a &= p^2 - q^2 \\
 b &= 2pq \text{ (always even)} \\
 c &= p^2 + q^2
 \end{aligned}$$

14.2 Difference between two consecutive numbers' square is odd

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

14.3 Summation

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

14.4 FFT

```

1  typedef unsigned int ui;
2  typedef long double ldb;
3  const ldb pi = atan2(0, -1);
4
5  struct Complex {
6      ldb real, imag;
7      Complex(): real(0.0), imag(0.0) {};
8      Complex(ldb a, ldb b) : real(a), imag(b) {};
9      Complex conj() const {
10         return Complex(real, -imag);
11     }
12     Complex operator + (const Complex& c) const {
13         return Complex(real + c.real, imag + c.imag);
14     }
15     Complex operator - (const Complex& c) const {
16         return Complex(real - c.real, imag - c.imag);
17     }
18     Complex operator * (const Complex& c) const {
19         return Complex(real*c.real - imag*c.imag, real*c.imag + imag*c.
20         real);
21     }
22     Complex operator / (ldb x) const {
23         return Complex(real / x, imag / x);
24     }
25     Complex operator / (const Complex& c) const {
26         return *this * c.conj() / (c.real * c.real + c.imag * c.imag);
27     }
28 };
29
30 inline ui rev_bit(ui x, int len){
31     x = ((x & 0x55555555u) << 1) | ((x & 0xAAAAAAAAu) >> 1);
32     x = ((x & 0x33333333u) << 2) | ((x & 0xCCCCCCCCu) >> 2);
33     x = ((x & 0x0F0F0F0Fu) << 4) | ((x & 0xF0F0F0F0u) >> 4);
34     x = ((x & 0x00FF00FFu) << 8) | ((x & 0xFF00FF00u) >> 8);
35     x = ((x & 0x0000FFFFu) << 16) | ((x & 0xFFFF0000u) >> 16);
36     return x >> (32 - len);
37 }
38
39 // flag = -1 if ifft else +1
40 void fft(vector<Complex>& a, int flag = +1) {
41     int n = a.size(); // n should be power of 2
42
43     int len = __builtin_ctz(n);
44     for (int i = 0; i < n; i++) {
45         int rev = rev_bit(i, len);
46
47         if (i < rev)
48             swap(a[i], a[rev]);
49     }
50
51     for (int m = 2; m <= n; m <= 1) { // width of each item
52         auto wm = Complex(cos(2 * pi / m), flag * sin(2 * pi / m));
53         for (int k = 0; k < n; k += m) { // start idx of each item
54             auto w = Complex(1, 0);
55             for (int j = 0; j < m / 2; j++) { // iterate half
56                 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.

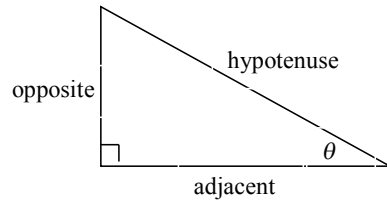
Trig Cheat Sheet

Definition of the Trig Functions

Right triangle definition

For this definition we assume that

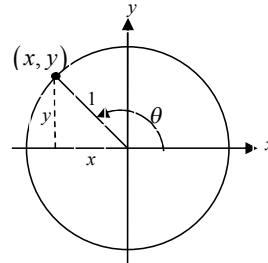
$$0 < \theta < \frac{\pi}{2} \text{ or } 0^\circ < \theta < 90^\circ.$$



$$\begin{aligned}\sin \theta &= \frac{\text{opposite}}{\text{hypotenuse}} & \csc \theta &= \frac{\text{hypotenuse}}{\text{opposite}} \\ \cos \theta &= \frac{\text{adjacent}}{\text{hypotenuse}} & \sec \theta &= \frac{\text{hypotenuse}}{\text{adjacent}} \\ \tan \theta &= \frac{\text{opposite}}{\text{adjacent}} & \cot \theta &= \frac{\text{adjacent}}{\text{opposite}}\end{aligned}$$

Unit circle definition

For this definition θ is any angle.



$$\begin{aligned}\sin \theta &= \frac{y}{1} = y & \csc \theta &= \frac{1}{y} \\ \cos \theta &= \frac{x}{1} = x & \sec \theta &= \frac{1}{x} \\ \tan \theta &= \frac{y}{x} & \cot \theta &= \frac{x}{y}\end{aligned}$$

Facts and Properties

Domain

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

$$\begin{aligned}\sin \theta, \quad \theta &\text{ can be any angle} \\ \cos \theta, \quad \theta &\text{ can be any angle} \\ \tan \theta, \quad \theta &\neq \left(n + \frac{1}{2}\right)\pi, \quad n = 0, \pm 1, \pm 2, \dots \\ \csc \theta, \quad \theta &\neq n\pi, \quad n = 0, \pm 1, \pm 2, \dots \\ \sec \theta, \quad \theta &\neq \left(n + \frac{1}{2}\right)\pi, \quad n = 0, \pm 1, \pm 2, \dots \\ \cot \theta, \quad \theta &\neq n\pi, \quad n = 0, \pm 1, \pm 2, \dots\end{aligned}$$

Range

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

$$\begin{aligned}-1 \leq \sin \theta \leq 1 & \quad \csc \theta \geq 1 \text{ and } \csc \theta \leq -1 \\ -1 \leq \cos \theta \leq 1 & \quad \sec \theta \geq 1 \text{ and } \sec \theta \leq -1 \\ -\infty < \tan \theta < \infty & \quad -\infty < \cot \theta < \infty\end{aligned}$$

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.

$$\begin{aligned}\sin(\omega\theta) &\rightarrow T = \frac{2\pi}{\omega} \\ \cos(\omega\theta) &\rightarrow T = \frac{2\pi}{\omega} \\ \tan(\omega\theta) &\rightarrow T = \frac{\pi}{\omega} \\ \csc(\omega\theta) &\rightarrow T = \frac{2\pi}{\omega} \\ \sec(\omega\theta) &\rightarrow T = \frac{2\pi}{\omega} \\ \cot(\omega\theta) &\rightarrow T = \frac{\pi}{\omega}\end{aligned}$$

Formulas and Identities

Tangent and Cotangent Identities

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

Reciprocal Identities

$$\begin{aligned}\csc \theta &= \frac{1}{\sin \theta} & \sin \theta &= \frac{1}{\csc \theta} \\ \sec \theta &= \frac{1}{\cos \theta} & \cos \theta &= \frac{1}{\sec \theta} \\ \cot \theta &= \frac{1}{\tan \theta} & \tan \theta &= \frac{1}{\cot \theta}\end{aligned}$$

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

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

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

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

Periodic Formulas

If n is an integer.

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

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

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

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)

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

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

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

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

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

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

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

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

Sum to Product Formulas

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

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

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

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

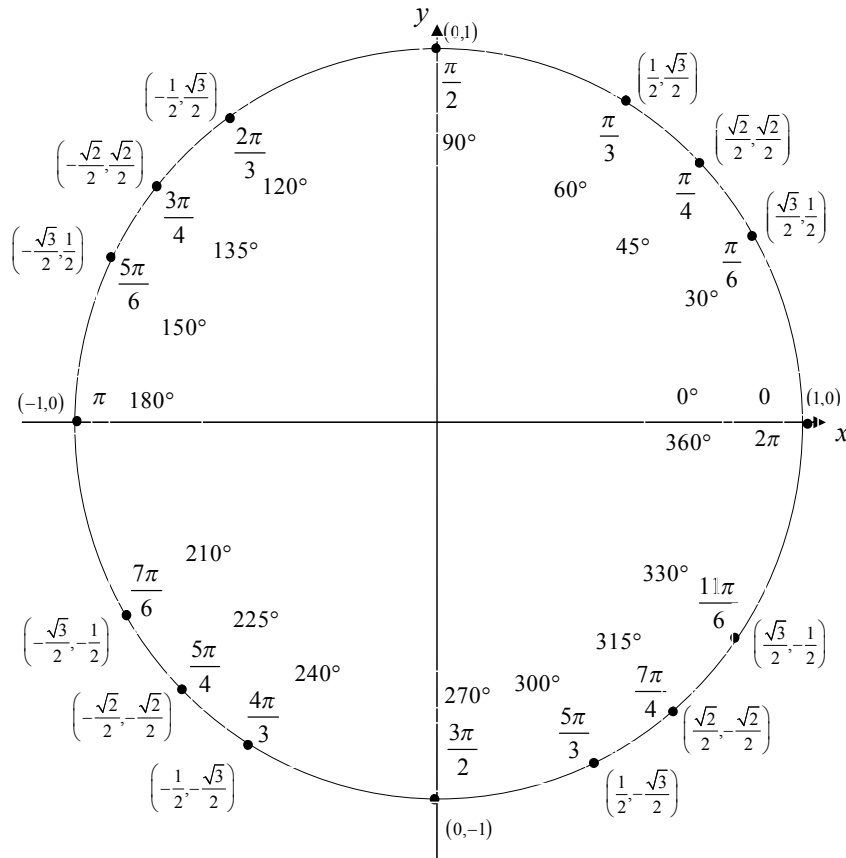
Cofunction Formulas

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

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

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

Unit Circle



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

Example

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

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 $x = \tan y$

Inverse Properties

$$\cos(\cos^{-1}(x)) = x \quad \cos^{-1}(\cos(\theta)) = \theta$$

$$\sin(\sin^{-1}(x)) = x \quad \sin^{-1}(\sin(\theta)) = \theta$$

$$\tan(\tan^{-1}(x)) = x \quad \tan^{-1}(\tan(\theta)) = \theta$$

Domain and Range

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

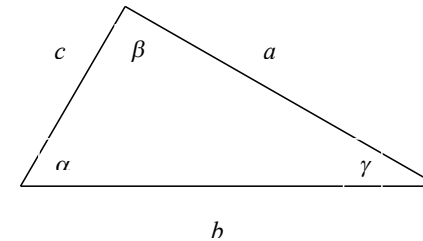
Alternate Notation

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

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

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

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

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

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

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

15 Dynamic Programming - Problems collection

好题收集