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

```
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

```cpp
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

```java
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. 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
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

5.7.1   scanf/printf reference

5.7.2   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>
```

5.7.3   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>
```

5.7.4   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);
```

5.7.5   String

5.7.6   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);
```

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

```
// 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

```
// 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) {
```

```
 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    }
```

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

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.

9.3 Shortest Path

Time complexity notations: V = vertex, E = edge

9.3.1 Dijkatra

密集圖別用 priority queue!

```
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 Dijkatra (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
```

```cpp
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              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

```cpp
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!
```

```cpp
31                  return true;
32              }
33          }
34      }
35     }
36  }
37
38  return false;
39 }
```

9.3.4   Bellman-Ford $O(VE)$

```cpp
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
           exists
11         bool hasChange = false;
12         for (int j = 0; j < (int)edge.size(); j++) {
13             int u = edge[j].first.first, v = edge[j].first.second, w =
               edge[j].second;
14             if (d[u] != INT_MAX && d[u] + w < d[v]) {
15                 hasChange = true;
16                 d[v] = d[u] + w;
17             }
18         }
19
20         if (i == n - 1 && hasChange == true)
21             loop = true;
22         else if (hasChange == false)
23             break;
24     }
25 }
```

9.3.5   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 ← cycle .

```cpp
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)

```cpp
struct Edge {
    int to, cap, rev;
    Edge(int a, int b, int c) {
        to = a;
        cap = b;
        rev = c;
    }
};

const int INF = 0x3f3f3f3f;
const int MAX_V = 20000 + 10;
// vector<Edge> g[MAX_V];
vector< vector<Edge> > g(MAX_V);
int level[MAX_V];
int iter[MAX_V];

inline void add_edge(int u, int v, int cap) {
    g[u].push_back((Edge){v, cap, (int)g[v].size()});
    g[v].push_back((Edge){u, 0, (int)g[u].size() - 1});
}

void bfs(int s) {
    memset(level, -1, sizeof(level));
    queue<int> q;

    level[s] = 0;
    q.push(s);

    while (!q.empty()) {
        int v = q.front(); q.pop();
        for (int i = 0; i < int(g[v].size()); i++) {
            const Edge& e = g[v][i];
            if (e.cap > 0 && level[e.to] < 0) {
                level[e.to] = level[v] + 1;
                q.push(e.to);
            }
        }
    }
}

int dfs(int v, int t, int f) {
    if (v == t) return f;
    for (int& i = iter[v]; i < int(g[v].size()); i++) {
        Edge& e = g[v][i];
        if (e.cap > 0 && level[v] < level[e.to]) {
            int d = dfs(e.to, t, min(f, e.cap));
            if (d > 0) {
                e.cap -= d;
                g[e.to][e.rev].cap += d;
                return d;
            }
        }
    }
    return 0;
}

int max_flow(int s, int t) { // dinic
    int flow = 0;
    for (;;) {
        bfs(s);
        if (level[t] < 0) return flow;
        memset(iter, 0, sizeof(iter));
        int f;
        while ((f = dfs(s, t, INF)) > 0) {
            flow += f;
        }
    }
}
```

10.2 Min Cost Flow

```cpp
#define st first
#define nd second

typedef pair<double, int> pii;
const double INF = 1e10;

struct Edge {
    int to, cap;
    double cost;
    int rev;
};

const int MAX_V = 2 * 100 + 10;
int V;
vector<Edge> g[MAX_V];
double h[MAX_V];
double d[MAX_V];
int prevv[MAX_V];
int preve[MAX_V];
// int match[MAX_V];

void add_edge(int u, int v, int cap, double cost) {
    g[u].push_back((Edge){v, cap, cost, (int)g[v].size()});
    g[v].push_back((Edge){u, 0, -cost, (int)g[u].size() - 1});
}

double min_cost_flow(int s, int t, int f) {
```

```cpp
        double res = 0;
        fill(h, h + V, 0);
        fill(match, match + V, -1);
        while (f > 0) {
            // dijkstra 找最小成本增廣路徑
            // without h will reduce to SPFA = O(V*E)
            fill(d, d + V, INF);
            priority_queue< pii, vector<pii>, greater<pii> > pq;

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

            while (!pq.empty()) {
                pii p = pq.top(); pq.pop();
                int v = p.nd;
                if (d[v] < p.st) continue;
                for (size_t i = 0; i < g[v].size(); i++) {
                    const Edge& e = g[v][i];
                    if (e.cap > 0 && d[e.to] > d[v] + e.cost + h[v] - h[e.to]) {
                        d[e.to] = d[v] + e.cost + h[v] - h[e.to];
                        prevv[e.to] = v;
                        preve[e.to] = i;
                        pq.push(pii(d[e.to], e.to));
                    }
                }
            }

            // 找不到增廣路徑
            if (d[t] == INF) return -1;

            // 維護 h[v]
            for (int v = 0; v < V; v++)
                h[v] += d[v];

            // 找瓶頸
            int bn = f;
            for (int v = t; v != s; v = prevv[v])
                bn = min(bn, g[prevv[v]][preve[v]].cap);

            // // find match
            // for (int v = prevv[t]; v != s; v = prevv[prevv[v]]) {
            //     int u = prevv[v];
            //     match[v] = u;
            //     match[u] = v;
            // }

            // 更新剩餘圖
            f -= bn;
            res += bn * h[t]; // SPFA: res += bn * d[t]
            for (int v = t; v != s; v = prevv[v]) {
                Edge& e = g[prevv[v]][preve[v]];
                e.cap -= bn;
                g[v][e.rev].cap += bn;
            }
        }
    }
    return res;
}
```

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 $0xdefaced$

```c
#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;
}
```

11.2 KMP

```c
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
```

```c
                res++;
                j = f[j];
            } else {
                j++;
            }
        }
    }

    return res;
}
```

11.3 Z Algorithm

```c
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

```c
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;
```

```
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 Gauss Jordan

```
1    typedef long long ll;
2    typedef vector<ll> vec;
3    typedef vector<vec> mat;
4
5    vec gauss_jordan(mat A) {
6        int n = A.size(), m = A[0].size();
7        for (int i = 0; i < n; i++) {
8            // float: find j s.t. A[j][i] is max
9            // mod: find min j s.t. A[j][i] is not 0
```

```
10           int pivot = i;
11           for (int j = i; j < n; j++) {
12               // if (fabs(A[j][i]) > fabs(A[pivot])) {
13               //     pivot = j;
14               // }
15               if (A[pivot][i] != 0) {
16                   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];
```

```cpp
            for (int k = 0; k < n; k++)
                m[a][k] = m[a][k] - m[b][k] * q;
            swap(a, b);
        }

        if (a != i) {
            swap(m[i], m[j]);
            det = -det;
        }
    }

    if (m[i][i] == 0)
        return 0;
    else
        det *= m[i][i];
}
return det;
}
```

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

```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);
}
```

```cpp
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
        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
```

```
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);
            for (int j = 0; j < m / 2; j++) { // iterate half
                Complex t = w * a[k + j + m / 2];
                Complex u = a[k + j];
                a[k + j] = u + t;
                a[k + j + m / 2] = u - t;
                w = w * wm;
            }
        }
    }

    if (flag == -1) { // if it's ifft
        for (int i = 0; i < n; i++)
            a[i].real /= n;
    }
}

vector<int> mul(const vector<int>& a, const vector<int>& b) {
    int n = int(a.size()) + int(b.size()) - 1;
    int nn = 1;
    while (nn < n)
        nn <<= 1;

    vector<Complex> fa(nn, Complex(0, 0));
    vector<Complex> fb(nn, Complex(0, 0));
    for (int i = 0; i < int(a.size()); i++)
        fa[i] = Complex(a[i], 0);
    for (int i = 0; i < int(b.size()); i++)
        fb[i] = Complex(b[i], 0);

    fft(fa, +1);
    fft(fb, +1);
    for (int i = 0; i < nn; i++) {
        fa[i] = fa[i] * fb[i];
    }
    fft(fa, -1);

    vector<int> c;
    for(int i = 0; i < nn; i++) {
        int val = int(fa[i].real + 0.5);
        if (val) {
            while (int(c.size()) <= i)
                c.push_back(0);
            c[i] = 1;
        }
    }

    return c;
}
```

14.5 Combination

14.5.1 Pascal triangle

```cpp
#define N 210
ll C[N][N];

void Combination() {
    for(ll i=0; i<N; i++) {
        C[i][0] = 1;
        C[i][i] = 1;
    }

    for(ll i=2; i<N; i++) {
        for(ll j=1; j<=i; j++) {
            C[i][j] = (C[i-1][j] + C[i-1][j-1])%M; // if needed, mod it
        }
    }
}
```

14.5.2 線性

```cpp
ll binomialCoeff(ll n, ll k)
{
    ll 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;
}
```

14.6 重複組合

14.7 Chinese remainder theorem

```cpp
typedef long long ll;

struct Item {
    ll m, r;
};

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;
    }
}

Item extcrt(const vector<Item> &v)
{
    ll m1 = v[0].m, r1 = v[0].r, x, y;

    for (int i = 1; i < int(v.size()); i++) {
        ll m2 = v[i].m, r2 = v[i].r;
        ll g = extgcd(m1, m2, x, y); // now x = (m/g)^(-1)

        if ((r2 - r1) % g != 0)
            return {-1, -1};

        ll k = (r2 - r1) / g * x % (m2 / g);
        k = (k + m2 / g) % (m2 / g); // for the case k is negative

        ll m = m1 * m2 / g;
        ll r = (m1 * k + r1) % m;

        m1 = m;
        r1 = (r + m) % m; // for the case r is negative
    }

    return (Item) {
        m1, r1
    };
}
```

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

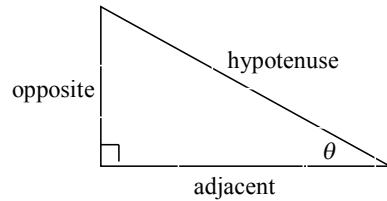1. 如果 $\frac{b}{a}$ 是最簡分數, 則 $1 - \frac{b}{a}$ 也是
2.

Trig Cheat Sheet

Definition of the Trig Functions

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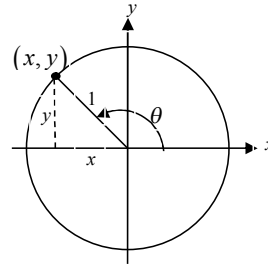For this definition we assume that
$0 < \theta < \dfrac{\pi}{2}$ or $0° < \theta < 90°$.

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

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

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

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

Range
The range is all possible values to get out of the function.
$-1 \le \sin\theta \le 1 \qquad \csc\theta \ge 1 \text{ and } \csc\theta \le -1$
$-1 \le \cos\theta \le 1 \qquad \sec\theta \ge 1 \text{ and } \sec\theta \le -1$
$-\infty < \tan\theta < \infty \qquad -\infty < \cot\theta < \infty$

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
$$\sin(2\theta) = 2\sin\theta\cos\theta$$
$$\cos(2\theta) = \cos^2\theta - \sin^2\theta$$
$$\qquad\;\;\; = 2\cos^2\theta - 1$$
$$\qquad\;\;\; = 1 - 2\sin^2\theta$$
$$\tan(2\theta) = \frac{2\tan\theta}{1 - \tan^2\theta}$$

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

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

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

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

Sum to Product Formulas
$$\sin\alpha + \sin\beta = 2\sin\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right)$$
$$\sin\alpha - \sin\beta = 2\cos\left(\frac{\alpha+\beta}{2}\right)\sin\left(\frac{\alpha-\beta}{2}\right)$$
$$\cos\alpha + \cos\beta = 2\cos\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right)$$
$$\cos\alpha - \cos\beta = -2\sin\left(\frac{\alpha+\beta}{2}\right)\sin\left(\frac{\alpha-\beta}{2}\right)$$
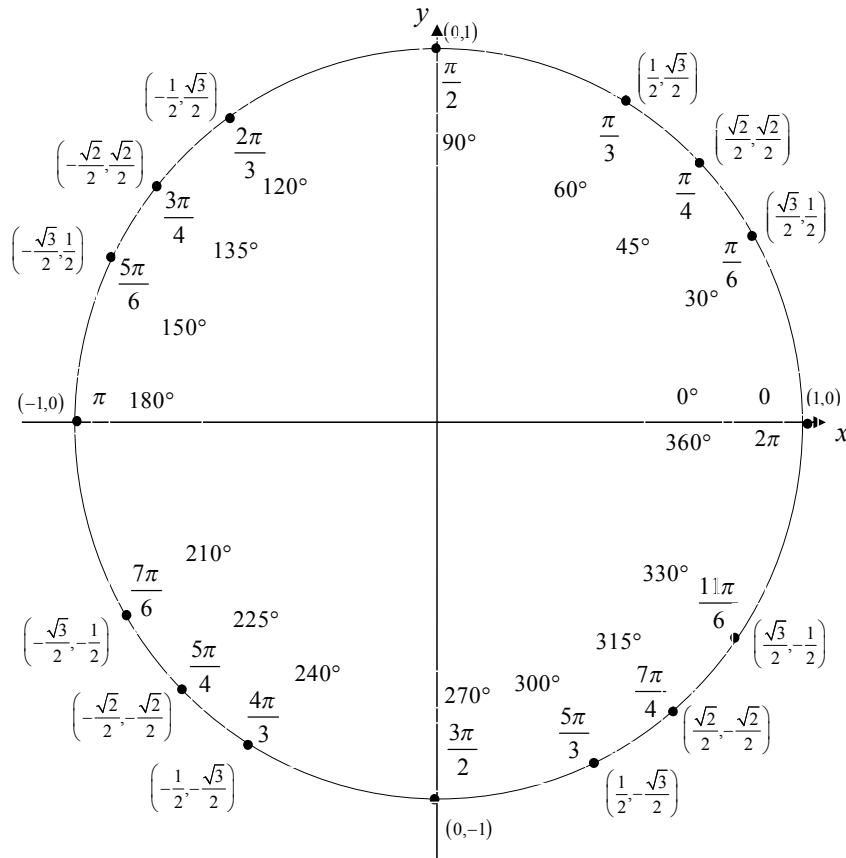
Cofunction Formulas
$$\sin\left(\frac{\pi}{2}-\theta\right) = \cos\theta \qquad \cos\left(\frac{\pi}{2}-\theta\right) = \sin\theta$$
$$\csc\left(\frac{\pi}{2}-\theta\right) = \sec\theta \qquad \sec\left(\frac{\pi}{2}-\theta\right) = \csc\theta$$
$$\tan\left(\frac{\pi}{2}-\theta\right) = \cot\theta \qquad \cot\left(\frac{\pi}{2}-\theta\right) = \tan\theta$$

Unit Circle

$$\left(-\tfrac{1}{2},\tfrac{\sqrt{3}}{2}\right)$$

$$\left(-\tfrac{\sqrt{2}}{2},\tfrac{\sqrt{2}}{2}\right)$$

$$\left(-\tfrac{\sqrt{3}}{2},\tfrac{1}{2}\right)$$

$$(0,1)$$

$$\left(\tfrac{1}{2},\tfrac{\sqrt{3}}{2}\right)$$

$$\left(\tfrac{\sqrt{2}}{2},\tfrac{\sqrt{2}}{2}\right)$$

$$\left(\tfrac{\sqrt{3}}{2},\tfrac{1}{2}\right)$$

$$y$$ $$\tfrac{\pi}{2}$$ $$90°$$ $$\tfrac{\pi}{3}$$ $$60°$$ $$\tfrac{\pi}{4}$$ $$45°$$ $$\tfrac{\pi}{6}$$ $$30°$$

$$\tfrac{2\pi}{3}\quad 120°$$ $$\tfrac{3\pi}{4}\quad 135°$$ $$\tfrac{5\pi}{6}\quad 150°$$

$$(-1,0)\quad \pi\quad 180°\qquad 0°\quad 0\quad (1,0)$$

$$360°\quad 2\pi\quad x$$

$$210°\quad \tfrac{7\pi}{6}$$ $$225°\quad \tfrac{5\pi}{4}$$ $$240°\quad \tfrac{4\pi}{3}$$ $$270°\quad \tfrac{3\pi}{2}$$ $$300°\quad \tfrac{5\pi}{3}$$ $$315°\quad \tfrac{7\pi}{4}$$ $$330°\quad \tfrac{11\pi}{6}$$

$$\left(-\tfrac{\sqrt{3}}{2},-\tfrac{1}{2}\right)$$

$$\left(-\tfrac{\sqrt{2}}{2},-\tfrac{\sqrt{2}}{2}\right)$$

$$\left(-\tfrac{1}{2},-\tfrac{\sqrt{3}}{2}\right)$$

$$(0,-1)$$

$$\left(\tfrac{1}{2},-\tfrac{\sqrt{3}}{2}\right)$$

$$\left(\tfrac{\sqrt{2}}{2},-\tfrac{\sqrt{2}}{2}\right)$$

$$\left(\tfrac{\sqrt{3}}{2},-\tfrac{1}{2}\right)$$

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 $x = \tan y$

Inverse Properties

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

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

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

Domain and Range

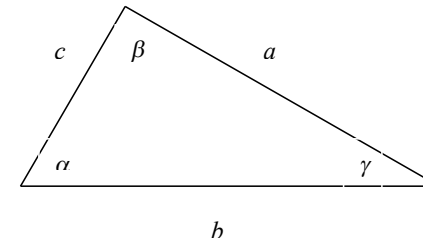| Function	Domain	Range
$y = \sin^{-1} x$	$-1 \le x \le 1$	$-\frac{\pi}{2} \le y \le \frac{\pi}{2}$
$y = \cos^{-1} x$	$-1 \le x \le 1$	$0 \le y \le \pi$
$y = \tan^{-1} x$	$-\infty < x < \infty$	$-\frac{\pi}{2} < y < \frac{\pi}{2}$

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