Contents

1 Basic

1.1 BIT

```cpp
#define lowbit(k) (k & -k)

int n;
vector<int> B1, B2;

void add(vector<int> &tr, int id, int val) {
  for (; id <= n; id += lowbit(id)) {
    tr[id] += val;
  }
}
void range_add(int l, int r, int val) {
  add(B1, l, val);
  add(B1, r + 1, -val);
  add(B2, l, val * (l - 1));
  add(B2, r + 1, -val * r);
}
int sum(vector<int> &tr, int id) {
  int ret = 0;
  for (; id >= 1; id -= lowbit(id)) {
    ret += tr[id];
  }
  return ret;
}
int prefix_sum(int id) {
  return sum(B1, id) * id - sum(B2, id);
}
int range_sum(int l, int r) {
  return prefix_sum(r) - prefix_sum(l - 1);
}
```

1.2 Black Magic

```cpp
#include <bits/extc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
using namespace __gnu_pbds;
using set_t =
  tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update>;//紅黑樹(set)
using map_t =
  tree<int, int, less<int>, rb_tree_tag,
    tree_order_statistics_node_update>;//紅黑樹(map)
using heap_t =
  __gnu_pbds::priority_queue<int>;
using ht_t =
  gp_hash_table<int, int>;
int main() {
  //set-----------------------------------
  set_t st;
  st.insert(5); st.insert(6);
  st.insert(3); st.insert(1);

  // the smallest is (0), biggest is (n-1), kth small is (k-1)
  int num = *st.find_by_order(0);
  cout << num << '\n'; // print 1

  num = *st.find_by_order(st.size() - 1);
  cout << num << '\n'; // print 6

  // find the index
  int index = st.order_of_key(6);//在裡面第幾大
  cout << index << '\n'; // print 3

  // check if there exists x
  int x = 5;
  int check = st.erase(x);
  if (check == 0) printf("st not contain 5\n");
  else if (check == 1) printf("st contain 5\n");

  //tree policy like set
  st.insert(5); st.insert(5);
  cout << st.size() << '\n'; // print 4

  //map-----------------------------------
  map_t mp;
  mp[1] = 2;
  cout << mp[1] << '\n';
  auto tmp = *mp.find_by_order(0); // pair
  cout << tmp.first << " " << tmp.second << '\n';

  //heap----------------------------------
  heap_t h1, h2;
  h1.push(1); h1.push(3);
  h2.push(2); h2.push(4);
  h1.join(h2);
  cout << h1.size() << h2.size() << h1.top() << '\n';
  // 支援合併
  // 404

  //hash-table----------------------------
  ht_t ht;
  ht[85] = 5;
  ht[89975] = 234;
  for (auto i : ht) {
    cout << i.first << " " << i.second << '\n';
  }
  //比較強的unorder map
}
```

1.3 DJS

```cpp
const int MAXN = 1000;
int boss[MAXN];
void init(int n) {
  for (int i = 0; i < n; i++) {
    boss[i] = -1;
  }
}
int find(int x) {
  if (boss[x] < 0) {
    return x;
  }
  return boss[x] = find(boss[x]);
}
bool uni(int a, int b) {
  a = find(a);
  b = find(b);
  if (a == b) {
    return false;
  }
  if (boss[a] > boss[b]) {
    swap(a, b);
  }
  boss[a] += boss[b];
  boss[b] = a;
  return true;
}
```

1.4 DFS

```cpp
struct Edge {
    int bi,color;//a連接到的bi,通道顏色
    bool operator < (const Edge &other) const{
        return color < other.color;
    }
};
vector<Edge>G[maxn];

void DFS(int me,int mydad,int distance){
    if(dist[me] < distance) return;
    dist[me] = distance;
    for(int i = 0;i<G[me].size();i++){
        int v = G[me][i].bi;
        DFS(v,me,distance+1);
    }
}
```

1.5 BFS

```cpp
bool visit[maxn];//訪問過的
void BFS(int point){
    queue<int>q;
    q.push(point);
    while(!q.empty()){
        int u = q.front();
        if(visit[u]) continue;//訪問過就下一個
        visit[u] = true;
        for(int i = 0;i<edge[u][i];i++){//連出去的線丟到queue
            q.push(edge[u][i]);
        }
    }
}
```

1.6 Segment Tree

```cpp
#include <./basic/Template.h>
const int INF = 1e9;
const int MAXN = ;

int n;
int a[MAXN], tr[MAXN << 1];

// !!! remember to call this function
void build() {
  for (int i = 0; i < n; i++) {
    tr[i + n] = a[i];
  }
  for (int i = n - 1; i > 0; i--) {
    tr[i] = max(tr[i << 1], tr[i << 1 | 1]);
  }
}
void update(int id, int val) {
  for (tr[id += n] = val; id > 1; id >>= 1) {
    tr[id >> 1] = max(tr[id], tr[id ^ 1]);
  }
}
int query(int l, int r) { // [l, r)
  int ret = -INF;
  for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
    if (l & 1) {
      ret = max(ret, tr[l++]);
    }
    if (r & 1) {
      ret = max(ret, tr[--r]);
    }
  }
  return ret;
}
```

1.7 Binary Serach

```cpp
lower_bound(a, a + n, k);      //最左邊 ≥ k 的位置
upper_bound(a, a + n, k);      //最左邊 > k 的位置
upper_bound(a, a + n, k) - 1;  //最右邊 ≤ k 的位置
lower_bound(a, a + n, k) - 1;  //最右邊 < k 的位置
[lower_bound, upper_bound)     //等於 k 的範圍
equal_range(a, a + n, k);
```

1.8 Template

```cpp
#pragma GCC optimize("O2")
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
using ULL = unsigned long long;
using PII = pair<int, int>;
using PLL = pair<LL, LL>;
using VI = vector<int>;
using VVI = vector<vector<int>>;
using dvt = double;
const int INF = 1e9;
const int MXN = 0;
const int MXV = 0;
const double EPS = 1e-9;
const int MOD = 1e9 + 7;
typedef long long ll;
typedef vector<int> vi;
typedef vector<string> vs;
typedef pair<int, int> pii;
typedef vector<pii> vpii;
#define MP make_pair
#define SORT(a) sort(a.begin(), a.end())
#define REVERSE(a) reverse(a.begin(), a.end())
#define ALL(a) a.begin(), a.end()
#define PI acos(-1)
#define ms(x, y) memset(x, y, sizeof(x))
#define inf 1e9
#define INF 1e16
#define pb push_back
#define MAX 100005
#define debug(a, b) cout << a << ": " << b << endl
```

```
#define Debug cout << "Reached here" << endl
#define prnt(a) cout << a << "\n"
#define mod 1000000007LL
#define FOR(i, a, b) for (int i = (a); i < (b); i++)
#define FORr(i, a, b) for (int i = (a); i >= (b); i--)
#define itrALL(c, itr) for (__typeof((c).begin()) itr = (c).begin(); itr != (c).end(); itr++)
#define lc ((node) << 1)
#define rc ((node) << 1 | 1)
#define VecPrnt(v)           \
    FOR(J, 0, v.size())      \
        cout << v[J] << " "; \
    cout << endl
#define endl "\n"
#define PrintPair(x) cout << x.first << " " << x.second << endl
#define EPS 1e-9
#define ArrPrint(a, st, en)        \
    for (int J = st; J <= en; J++) \
        cout << a[J] << " ";       \
    cout << endl;
#define MP make_pair
#define PB push_back
#define Fi first
#define Se second
#define FOR(i, L, R) for (int i = L; i < (int)R; ++i)
#define FORD(i, L, R) for (int i = L; i > (int)R; --i)
#define IOS            \
    cin.tie(nullptr);  \
    cout.tie(nullptr); \
    ios_base::sync_with_stdio(false);

int main()
{
    // ios_base::sync_with_stdio(0);
    // cin.tie(NULL); cout.tie(NULL);
    // freopen("in.txt","r",stdin);
    IOS;
}

/* Direction Array */

// int fx[]={1,-1,0,0};
// int fy[]={0,0,1,-1};
// int fx[]={0,0,1,-1,-1,1,-1,1};
// int fy[]={-1,1,0,0,1,1,-1,-1};

/***************** END OF HEADER *****************/
```

2 Data Structure

2.1 Range Sum Query

```
#include <./basic/Template.h>
int a[MAX + 7], tree[4 * MAX + 7], lazy[4 * MAX + 7];
void build(int node, int l, int r)
{
    if (l == r)
    {
        tree[node] = a[l];
        return;
    }
    if (l >= r)
        return;
    int mid = (l + r) / 2;
    build(node * 2, l, mid);
    build(node * 2 + 1, mid + 1, r);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}
void upd(int node, int l, int r, int v)
{
    lazy[node] += v;
    tree[node] += (r - l + 1) * x;
}
void pushDown(int node, int l, int r) //passing update information to the children
{
    int mid = (l + r) / 2;
    upd(node * 2, l, mid, lazy[node]);
    upd(node * 2 + 1, mid + 1, r, lazy[node]);
    lazy[node] = 0;
}
void update(int node, int l, int r, int x, int y, int v)
{
    if (x > r || y < l)
        return;
    if (x >= l && r <= y)
    {
        upd(node, l, r, v);
        return;
    }
    pushDown(node, l, r);
    int mid = (l + r) / 2;
    update(node * 2, l, mid, x, y, v);
    update(node * 2 + 1, mid + 1, r, x, y, v);
    tree[node] = tree[node * 2] + tree[node * 2 + 1];
}
```

3 DP

3.1 LCS

```
int LCS(string s1, string s2) {
  int n1 = s1.size(), n2 = s2.size();
  vector<vector<int>> dp(n1 + 1, vector<int>(n2 + 1, 0));
  for (int i = 1; i <= n1; i++) {
    for (int j = 1; j <= n2; j++) {
      if (s1[i - 1] == s2[j - 1]) {
        dp[i][j] = dp[i - 1][j - 1] + 1;
      } else {
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
      }
    }
  }
  return dp[n1][n2];
}
```

3.2 LIS

```
int LIS(vector<int> &a) {
  vector<int> s;
  for (int i = 0; i < a.size(); i++) {
    if (s.empty() || s.back() < a[i]) {
      s.push_back(a[i]);
    } else {
      *lower_bound(s.begin(), s.end(), a[i],
        [](int x, int y) {return x < y;}) = a[i];
    }
  }
  return s.size();
}
```

3.3 迴文

```
bool isPalindrome[100][100];
// Find the palindromes of a string in O(n^2)

int main()
{
  ios_base::sync_with_stdio(0);
  // freopen("in.txt","r",stdin);
  string s;
```

```cpp
  cin>>s;
  int len=s.size();
  for(int i=0; i<len; i++)
    isPalindrome[i][i]=true;

  for(int k=1; k<len; k++){
    for(int i=0; i+k<len; i++){
      int j=i+k;
      isPalindrome[i][j]=(s[i]==s[j]) &&
      (isPalindrome[i+1][j-1] || i+1>=j-1);
    }
  }
  return 0;
}
```

3.4 2DMaxSubArray

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

using namespace std;

#define size 4

int arr[size][size];

int maxSubArr()
{

    int b[size];
    int MAX = -11111111;

    for (int i = 0; i < size; i++)
    {

        memset(b, 0, sizeof(b));
        for (int j = i; j < size; j++)
        {

            int s = 0;
            for (int k = 0; k < size; k++)
            {

                b[k] += arr[j][k];
                s += b[k];
                if (s <= 0)
                    s = b[k];
                if (s > MAX)
                    MAX = s;
            }
        }
    }
    return MAX;
}

int main()
{

#ifdef DBG
    freopen("1.in", "r", stdin);
    freopen("2.out", "w", stdout);
#endif

    for (int i = 0; i < size; i++)
        for (int j = 0; j < size; j++)
            cin >> arr[i][j];

    maxSubArr();

    return 0;
}
```

4 Geometry

4.1 Convex hull

```cpp
#include <./basic/Template.h>
struct PT
{
    int x, y;
    PT() {}
    PT(int x, int y) : x(x), y(y) {}
    bool operator<(const PT &P) const
    {
        return x < P.x || (x == P.x && y < P.y);
    }
};

ll cross(const PT p, const PT q, const PT r)
{
    return (ll)(q.x - p.x) * (ll)(r.y - p.y) - (ll)(q.y - p.y) * (ll)(r.x - p.x);
}

vector<PT> Points, Hull;

void findConvexHull()
{
    int n = Points.size(), k = 0;

    SORT(Points);

    // Build lower hull

    FOR(i, 0, n)
    {
        while (Hull.size() >= 2 && cross(Hull[Hull.size() - 2], Hull.back(), Points[i]) <= 0)
        {
            Hull.pop_back();
            k--;
        }
        Hull.pb(Points[i]);
        k++;
    }

    // Build upper hull

    for (int i = n - 2, t = k + 1; i >= 0; i--)
    {
        while (Hull.size() >= t && cross(Hull[Hull.size() - 2], Hull.back(), Points[i]) <= 0)
        {
            Hull.pop_back();
            k--;
        }
        Hull.pb(Points[i]);
        k++;
    }

    Hull.resize(k);
}
```

5 Graph

5.1 Bellman Ford

```cpp
#include <./basic/Template.h>
bool bellman(int src)
{
    // Nodes are indexed from 1
  for (int i = 1; i <= n; i++)
    dist[i] = INF;
  dist[src] = 0;
    for(int i = 2; i <= n; i++)
    {
        for (int j = 0; j < edges.size(); j++)
        {
            int u = edges[j].first;
            int v = edges[j].second;
            ll weight = adj[u][v];
            if (dist[u]!=INF && dist[u] + weight < dist[v])
                dist[v] = dist[u] + weight;
        }
    }
  for (int i = 0; i < edges.size(); i++)
    {
    int u = edges[i].first;
    int v = edges[i].second;
    ll weight = adj[u][v];
        // True if neg-cylce exists
    if (dist[u]!=INF && dist[u] + weight < dist[v])
      return true;
    }
  return false;
}
```

5.2 Dijk

```
#include <./basic/Template.h>
const long long int INF = 1e18;
const int MAXN = 1000000;
struct Edge {
  int to;
  long long int cost;
  Edge(int v, long long int c) : to(v), cost(c) {}
  bool operator < (const Edge &other) const {
    return cost > other.cost;
  }
};

int n;
long long int dis[MAXN];
vector<Edge> G[MAXN];

void init() {
  for (int i = 0; i < n; i++) {
    G[i].clear();
    dis[i] = INF;
  }
}
void Dijkstra(int st, int ed = -1) {
  priority_queue<Edge> pq;
  pq.emplace(st, 0);
  dis[st] = 0;
  while (!pq.empty()) {
    auto now = pq.top();
    pq.pop();
    if (now.to == ed) {
      return;
    }
    if (now.cost > dis[now.to]) {
      continue;
    }
    for (auto &e : G[now.to]) {
      if (dis[e.to] > now.cost + e.cost) {
        dis[e.to] = now.cost + e.cost;
        pq.emplace(e.to, dis[e.to]);
      }
    }
  }
}
```

5.3 Edges

```
struct Edge
{
    int from, to, w;
    bool operator<(const Edge& rhs) // optional
    {
        return w < rhs.w;
    }
};
```

5.4 Floyd

```
const LL INF = 1e18;
const int MAXN = ;

int n;
LL G[MAXN][MAXN];

void init() {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      G[i][j] = INF;
    }
    G[i][i] = 0;
  }
}
void floyd() {
  for (int k = 0; k < n; k++) {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        if (G[i][k] != INF && G[k][j] != INF) {
          G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
        }
      }
    }
  }
}
```

5.5 KM

```
const int INF = 1e9;
const int MAXN = ;
struct KM { //1-base
  int n, G[MAXN][MAXN];
  int lx[MAXN], ly[MAXN], my[MAXN];
  bool vx[MAXN], vy[MAXN];
  void init(int _n) {
    n = _n;
    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
        G[i][j] = 0;
      }
    }
  }
  bool match(int i) {
    vx[i] = true;
    for (int j = 1; j <= n; j++) {
      if (lx[i] + ly[j] == G[i][j] && !vy[j]) {
        vy[j] = true;
        if (!my[j] || match(my[j])) {
          my[j] = i;
          return true;
        }
      }
    }
    return false;
  }
  void update() {
    int delta = INF;
    for (int i = 1; i <= n; i++) {
      if (vx[i]) {
        for (int j = 1; j <= n; j++) {
          if (!vy[j]) {
            delta = min(delta, lx[i] + ly[j] - G[i][j]);
```

```
     }
    }
   }
  }
  for (int i = 1; i <= n; i++) {
   if (vx[i]) {
    lx[i] -= delta;
   }
   if (vy[i]) {
    ly[i] += delta;
   }
  }
 }
 int run() {
  for (int i = 1; i <= n; i++) {
   lx[i] = ly[i] = my[i] = 0;
   for (int j = 1; j <= n; j++) {
    lx[i] = max(lx[i], G[i][j]);
   }
  }
  for (int i = 1; i <= n; i++) {
   while (true) {
    for (int i = 1; i <= n; i++) {
     vx[i] = vy[i] = 0;
    }
    if (match(i)) {
     break;
    } else {
     update();
    }
   }
  }
  int ans = 0;
  for (int i = 1; i <= n; i++) {
   ans += lx[i] + ly[i];
  }
  return ans;
 }
};
```

5.6 Global Minimum Cut

```
#include <./basic/Template.h>
/*Given an undirected graph G = (V, E), we define a cut of G to be a partition
of V into two non-empty sets A and B. Earlier, when we looked at network
flows, we worked with the closely related definition of an s-t cut: there, given
a directed graph G = (V, E) with distinguished source and sink nodes s and t,
an s-t cut was defined to be a partition of V into sets A and B such that s ∈ A
and t ∈ B. Our definition now is slightly different, since the underlying graph
is now undirected and there is no source or sink.
This problem can be solved by max-flow. First we remove undirected edges and replace
them by two opposite directed edge. Now we fix a node s. Then we consider each of
the n nodes as t and run max-flow. The minimum of those values is the answer.
This is O(n^3).
*/

struct Stoer_Wagner
{
  vector<vl> weights;
  Stoer_Wagner(ll N)
  {
    weights.resize(N, vl(N, 0));
  }
  void AddEdge(ll from, ll to, ll cap)
  {
    weights[from][to] += cap;
    weights[to][from] += cap;
  }
  pair<ll, vl> GetMinCut()
  {
    ll N = weights.size();
    vl used(N), cut, best_cut;
    ll best_weight = -1;

    for (ll phase = N - 1; phase >= 0; phase--)
    {
      vl w = weights[0];
      vl added = used;
      ll prev, last = 0;
      for (ll i = 0; i < phase; i++)
      {
        prev = last;
        last = -1;
        for (ll j = 1; j < N; j++)
          if (!added[j] && (last == -1 || w[j] > w[last]))
            last = j;
        if (i == phase - 1)
        {
          for (ll j = 0; j < N; j++)
            weights[prev][j] += weights[last][j];
          for (ll j = 0; j < N; j++)
            weights[j][prev] = weights[prev][j];
          used[last] = true;
          cut.push_back(last);
          if (best_weight == -1 || w[last] < best_weight)
          {
            best_cut = cut;
            best_weight = w[last];
          }
        }
        else
        {
          for (ll j = 0; j < N; j++)
            w[j] += weights[last][j];
          added[last] = true;
        }
      }
    }
    return make_pair(best_weight, best_cut);
  }
};

int main()
{
  ll T;
  sl(T);
  f(t, 1, T + 1)
  {
    ll N, M;
    sll(N, M);
    Stoer_Wagner SW(N);
    f(i, 0, M)
    {
      ll a, b, c;
      slll(a, b, c);
      SW.AddEdge(a - 1, b - 1, c);
    }
    pf("Case #%lld: ", t);
    pfl(SW.GetMinCut().x);
  }
}
```

5.7 Krushal

```
#include <./basic/Template.h>
struct edge
{
  int u, v, w;
  bool operator<(const edge &p) const
```

```
    {
        return w < p.w;
    }
};
edge get;
int parent[100];
vector<edge> e;
int find(int r)
{
    if (parent[r] == r)
        return r;
    return parent[r] = find(parent[r]);
}
int mst(int n)
{
    sort(e.begin(), e.end());
    for (int i = 1; i <= n; i++)
        parent[i] = i;
    int cnt = 0, s = 0;
    for (int i = 0; i < (int)e.size(); i++)
    {
        int u = find(e[i].u);
        int v = find(e[i].v);
        if (u != v)
        {
            parent[u] = v;
            cnt++;
            s += e[i].w;
            if (cnt == n - 1)
                break;
        }
    }
}
```

5.8 K-th Shortest Path Length

```
#include <./basic/Template.h>
int n, m, x, y, k, a, b, c;
vi Graph[103], Cost[103];
vector<priority_queue<int>> d(103);
priority_queue<pii> Q;

void goDijkstra()
{

    // Here, elements are sorted in decreasing order of the first elements
    // of the pairs and then the second elements if equal first element.
    // d[i] is the priority_queue of the node i where the best k path length
    // will be stored in decreasing order. So, d[i].top() has the longest of the
    // first k shortest path.
    d[x].push(0);
    Q.push(MP(x, 0));
    // Q contains the nodes in the increasing order of their cost
    // Since the priority_queue sorts the pairs in decreasing order of their
    // first element and then second element, to sort it in increasing order
    // we will negate the cost and push it.

    while (!Q.empty())
    {
        pii t = Q.top();
        Q.pop();
        int u = t.first, costU = -t.second;
        // Since the actual cost was negated.

        FOR(j, 0, Graph[u].size())
        {
            int v = Graph[u][j];

            // prnt(v); prnt(d[v].size());

            // Have we already got k shortest paths? Or is the longest path can be made better?
            if (d[v].size() < k || d[v].top() > costU + Cost[u][j])
            {
                int temp = costU + Cost[u][j];
                d[v].push(temp);
                Q.push(MP(v, -temp));
            }
            if (d[v].size() > k)
                d[v].pop();
            // If we have more than k shortest path for the current node, we can pop
            // the worst ones.
        }
    }

    if (d[y].size() < k)
        prnt(-1);
    // We have not found k shortest path for our destination.
    else
        prnt(d[y].top());
}

int main()
{
    // ios_base::sync_with_stdio(0);
    // cin.tie(NULL); cout.tie(NULL);
    // freopen("in.txt","r",stdin);

    while (scanf("%d%d", &n, &m) && n + m)
    {
        scanf("%d%d%d", &x, &y, &k);

        FOR(i, 0, m)
        {
            scanf("%d%d%d", &a, &b, &c);

            Graph[a].pb(b);
            Cost[a].pb(c);
        }

        goDijkstra();

        FOR(i, 0, 103)
        Graph[i].clear(),
            Cost[i].clear();
        FOR(i, 0, 103)
        {
            while (!d[i].empty())
                d[i].pop();
        }

        while (!Q.empty())
            Q.pop();
    }

    return 0;
}
```

5.9 SPFA

```

#include <./basic/Template.h>
#define MAXN 1000000
struct Edge
{
    int at;
    long long cost;
};
int n;
long long dis[MAXN];
vector<Edge> G[MAXN];
```

```
void init()
{
    for (int i = 0; i < n; i++)
    {
        G[i].clear();
        dis[i] = INF;
    }
}
bool SPFA(int st)
{
    vector<int> cnt(n, 0);
    vector<bool> inq(n, false);
    queue<int> q;

    q.push(st);
    dis[st] = 0;
    inq[st] = true;
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        inq[now] = false;
        for (auto &e : G[now])
        {
            if (dis[e.at] > dis[now] + e.cost)
            {
                dis[e.at] = dis[now] + e.cost;
                if (!inq[e.at])
                {
                    cnt[e.at]++;
                    if (cnt[e.at] > n)
                    {
                        // negative cycle
                        return false;
                    }
                    inq[e.at] = true;
                    q.push(e.at);
                }
            }
        }
    }
    return true;
}
```

5.10 BipartiteMatch

```
int n, m, Left[maxn], G[maxn][maxn];
bitset<maxn> used;

bool dfs(int s)
{
    for (int i = 1; i <= m; i++)
    {
        if (!G[s][i] || used[i])
        {
            continue;
        }
        used[i] = true;
        if (Left[i] == -1 || dfs(Left[i]))
        {
            Left[i] = s;
            return true;
        }
    }
    return false;
}

int sol()
{
    int ret = 0;
    memset(Left, -1, sizeof(Left));
    for (int i = 1; i <= n; i++)
    {
        used.reset();
        if (dfs(i))
        {
            ret++;
        }
    }
    return ret;
}
```

6 Math

6.1 GCDhjackh

```
int extgcd(int a, int b, int c, int &x, int &y) {
  if (b == 0) {
    x = c / a;
    y = 0;
    return a;
  }
  int d = extgcd(b, a % b, c, x, y);
  int tmp = x;
  x = y;
  y = tmp - (a / b) * y;
  return d;
}
```

6.2 Prime

```
const int maxn = ;
int arr[maxn];
int prime[maxn];
void init(){
  for (int i = 0; i < maxn; ++i){
    arr[i] = 0;
  }
}
void find(){
  int num = 0;
  for(int i = 2;i<maxn;i++){
    if(arr[i] == 0){
      prime[num] = 0;
      num++;
      for(int j = i*i;j<maxn;j+=i){
        arr[j] = 1;
      }
    }
  }
}
```

6.3 Gauss Elimination

```
#include <./basic/Template.h>
const int MAXN = 300;
const double EPS = 1e-8;
int n;
double A[MAXN][MAXN];
void Gauss()
{
    for (int i = 0; i < n; i++)
    {
        bool ok = 0;
        for (int j = i; j < n; j++)
        {
            if (fabs(A[j][i]) > EPS)
            {
                swap(A[j], A[i]);
                ok = 1;
                break;
            }
        }
        if (!ok)
            continue;
        double fs = A[i][i];
        for (int j = i + 1; j < n; j++)
        {
            double r = A[j][i] / fs;
            for (int k = i; k < n; k++)
            {
                A[j][k] -= A[i][k] * r;
            }
        }
    }
}
```

6.4 Matrix

```
template <typename T, int N = 2>
struct Mat
{ // Matrix
    unsigned long long v[N][N];
    Mat operator*(Mat b) const
    {
        Mat val;
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                val.v[i][j] = 0;
                for (int k = 0; k < N; k++)
                {
                    val.v[i][j] += v[i][k] * b.v[k][j];
                }
            }
        }
        return val;
    }
};
```

6.5 Josephus

```
int josephus(int n, int k){ // 有n個人圍成一圈，每k個一次
  return n > 1 ? (josephus(n-1,k)+k)%n : 0;
}// 回傳最後一人的編號, 0 index
```

7 String

7.1 KMP

```
void failure(string s, int len, int *f)
{
    f[ 0 ] = -1;
    for(int i = 1; i < len; i++)
    {
        int k = f[ i-1 ];

        while(s[i] != s[k+1] && k >= 0)
            k = f[k];

        if(s[i] == s[k+1])f[i] = k+1;
        else f[i] = -1;
    }
}

int compare(string big, string little, int *f)
{
    int Blen = big.length(), Llen = little.length();
    int i = 0, j = 0;

    while(i < Blen && j < Llen)
    {
        if(big[i] == little[j])
        {
            i++;
            j++;
        }
        else if(j == 0)i++;
        else j = f[j-1] + 1;
    }

    if(j == Llen)return 1;
    else return 0;
}
```

7.2 Trie

```
#include <./basic/Template.h>
struct Node
{
    char ch;
    int v;
    Node *next[26];
    Node()
    {
        v = 0;
        FOR(i, 0, 26)
        next[i] = NULL;
    }
};

void insert(Node *root, string s)
{
    FOR(i, 0, s.size())
    {
        int v = s[i] - 'a';
        if (root->next[v] == NULL)
        {
            root->next[v] = new Node();
        }
        root = root->next[v];
        ++root->v;
        root->ch = s[i];
    }
    return;
}
void search(Node *root, string s)
{
    FOR(i, 0, s.size())
    {
        int v = s[i] - 'a';
        root = root->next[v];
        if (root->v == 1)
        {
            cout << s << ' ' << s.substr(0, i + 1) << '\n';
            return;
        }
    }
    cout << s << ' ' << s << '\n';
}

int main()
{
    vector<string> v;
    string s;
    Node *root = new Node();
    while (cin >> s)
    {
        insert(root, s);
        v.push_back(s);
    }
    FOR(i, 0, v.size()) { search(root, v[i]); }
}
```

8 Python

8.1 Model

```
### EOF
while True:
  try:
    pass
  except EOFError:
    break
###math
import math

math.ceil(x)#上高斯
math.floor(x)#下高斯
math.factorial(x)#接乘
math.fabs(x)#絕對值
math.fsum(arr)#跟sum一樣但更精確(小數點問題)
math.gcd(x, y)#bj4
math.exp(x)#e^x
math.log(x, base)
math.log2(x)#2為底
math.log10(x)#10為底
math.sqrt(x)
math,pow(x, y)#精確些(float型態)
math.sin(x)# cos tan asin acos atan atan2(弧度) sinh cosh tanh acosh asinh atanh
math.hypot(x, y)#歐幾里德範數
math.degrees(x)#x從弧度轉角度
math.radians(x)#x從角度轉弧度
math.gamma(x)#x的gamma函數
math.pi#常數
math.e#常數
math.inf

### ascii

ord(x)#char to asc
chr(x)#asc to char

x.encode().hex()#string to hex
### reverse string
string = "abc"
string_reverse = string[::-1]
```