Contents

1   Math
1.1   快速幂

```cpp
// 根據費馬小定
    理，若 a p 互質，a^(p-2) 為 a 在 mod p 時的乘法逆元
int fast_pow(int a, int b, int mod)
{
    // a^b % mod
    int res = 1;
    while(b)
    {
        if(b & 1) res = (res * a) % mod;
        a = (a * a) % mod;
        b >>= 1;
    }
    return res;
}
```

1.2   擴展歐幾里得

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

int lcm(int a, int b)
{
    return a * b / gcd(a, b);
}

pair<int, int> ext_gcd
    (int a, int b) //擴展歐幾里德 ax+by = gcd(a,b)
{
    if (b == 0)
        return {1, 0};
    if (a == 0)
        return {0, 1};
    int x, y;
    tie(x, y) = ext_gcd(b % a, a);
    return make_pair(y - b * x / a, x);
}
```

2   Graph
2.1   Tarjan SCC

```cpp
class tarjan{
    //  1-base
    int time = 1;
    int id = 1;
    stack<int> s;
    vector<int> low;
    vector<int> dfn;
    vector<bool> in_stack;
    void dfs(int node, vector<vector<int>> &graph){
        in_stack[node] = true;
        s.push(node);
        dfn[node] = low[node] = time++;
        for(auto &j : graph[node]){
            if(dfn[j] == 0){
                dfs(j, graph);
                // 看看往下有沒有辦法回到更上面的點
                low[node] = min(low[node], low[j]);
            }
            else if(in_stack[j]){
                low[node] = min(low[node], low[j]);
            }
        }
        vector<int> t; // 儲存這個強連通分量
        if(dfn[node] == low[node]){
            while(s.top() != node){
                t.push_back(s.top());
                in_stack[s.top()] = false;
                scc_id[s.top()] = id;
                s.pop();
            }
            t.push_back(s.top());
            scc_id[s.top()] = id;
            in_stack[s.top()] = false;
            s.pop();
            id++;
        }
        if(!t.empty()) ans.push_back(t);
    }
    public:
    vector<int> scc_id;
    vector<vector<int>> ans;
    // ans ans[i] 代表第 i 個強連通分量裡面包涵的點
    // scc_id[i] 代表第 i 個點屬於第幾個強連通分量
    vector
        <vector<int>> scc(vector<vector<int>> &graph){
        int num = graph.size();
        scc_id.resize(num, -1);
        dfn.resize(num, 0);
        low.resize(num, 0);
        in_stack.resize(num, false);
        for(int i = 1; i < num; i++){
            if(dfn[i] == 0) dfs(i, graph);
        }
        return ans;
    }
};
```

2.2   Max flow

```cpp
#define int long long

// Edmonds-Karp Algorithm Time: O(VE^2) 實際上會快一點

class edge{
    public:
    int next;
    int capicity;
    int rev;
    edge(int _n, int _c
        , int _r) : next(_n), capicity(_c), rev(_r){};
};

vector<vector<edge>> graph;

void add_edge(int a, int b, int capacity){
    graph[
        a].push_back(edge(b, capacity, graph[b].size()));
    graph[b].push_back(edge(a, 0, graph[a].size() - 1));
}

int dfs(int now, int end
    , int flow, vector<pair<int, int>> &path, int idx){
    if(now == end) return flow;
    auto &e = graph[now][path[idx + 1].second];
    if(e.capicity > 0){
        auto ret = dfs(e.next
            , end, min(flow, e.capicity), path, idx + 1);
        if(ret > 0){
            e.capicity -= ret;
            graph[e.next][e.rev].capicity += ret;
            return ret;
        }
    }
    return 0;
}

vector<pair<int, int>> search_path(int start, int end){
    vector<pair<int, int>> ans;
    queue<int> q;
    vector
        <pair<int, int>> parent(graph.size(), {-1, -1});
    q.push(start);
    while(!q.empty()){
        int now = q.front();
        q.pop();
        for(int i = 0; i < (int)graph[now].size(); i++){
            auto &e = graph[now][i];
            if(e.
                capicity > 0 and parent[e.next].first == -1){
                parent[e.next] = {now, i};
                if(e.next == end) break;
                q.push(e.next);
            }
        }
```

```
    }
  }
  if(parent[end].first == -1) return ans;
  int now = end;
  while(now != start){
    auto [node, idx] = parent[now];
    ans.emplace_back(node, idx);
    now = node;
  }
  ans.emplace_back(start, -1);
  reverse(ans.begin(), ans.end());
  return ans;
}

int maxflow(int start, int end, int node_num){
  int ans = 0;
  while(1){
    vector<bool> visited(node_num + 1, false);
    auto tmp = search_path(start, end);
    if(tmp.size() == 0) break;
    auto flow = dfs(start, end, 1e9, tmp, 0);
    ans += flow;
  }
  return ans;
}
```

3 String
3.1 Hash

```
vector<int> Pow(int num){
  int p = 1e9 + 7;
  vector<int> ans = {1};
  for(int i = 0; i < num; i++)
    ans.push_back(ans.back() * b % p);
  return ans;
}

vector<int> Hash(string s){
  int p = 1e9 + 7;
  vector<int> ans = {0};
  for(char c:s){
    ans.push_back((ans.back() * b + c) % p);
  }
  return ans;
}

// 閉區間[l, r]
int query
    (vector<int> &vec, vector<int> &pow, int l, int r){
  int p = 1e9 + 7;
  int length = r - l + 1;
  return
      (vec[r + 1] - vec[l] * pow[length] % p + p) % p;
}
```

3.2 Zvalue

```
vector<int> z_func(string s1){
  int l = 0, r = 0, n = s1.size();
  vector<int> z(n, 0);
  for(int i = 1; i < n; i++){
    if(i
        <= r and z[i - l] < r - i + 1) z[i] = z[i - l];
    else{
      z[i] = max(z[i], r - i + 1);
      while(i + z
          [i] < n and s1[i + z[i]] == s1[z[i]]) z[i]++;
    }
    if(i + z[i] - 1 > r){
      l = i;
      r = i + z[i] - 1;
    }
  }
  return z;
}
```

4 Geometry
4.1 Static Convex Hull

```
#define mp(a, b) make_pair(a, b)
#define pb(a) push_back(a)
#define F first
#define S second
```

```
template<typename T>
pair<T, T> operator-(pair<T, T> a, pair<T, T> b){
    return mp(a.F - b.F, a.S - b.S);
}

template<typename T>
T cross(pair<T, T> a, pair<T, T> b){
    return a.F * b.S - a.S * b.F;
}

template<typename T>
vector<pair
    <T, T>> getConvexHull(vector<pair<T, T>>& pnts){
    sort(pnts.begin
        (), pnts.end(), [](pair<T, T> a, pair<T, T> b)
    { return
        a.F < b.F || (a.F == b.F && a.S < b.S); });
    auto cmp = [&](pair<T, T> a, pair<T, T> b)
    { return a.F == b.F && a.S == b.S; };
    pnts.erase(unique
        (pnts.begin(), pnts.end(), cmp), pnts.end());
    if(pnts.size()<=1)
        return pnts;
    int n = pnts.size();
    vector<pair<T, T>> hull;
    for(int i = 0; i < 2; i++){
        int t = hull.size();
        for(pair<T, T> pnt : pnts){
            while(hull.size() - t >= 2 &&
                cross(hull.back() - hull[hull.size() -
                2], pnt - hull[hull.size() - 2]) <= 0){
                hull.pop_back();
            }
            hull.pb(pnt);
        }
        hull.pop_back();
        reverse(pnts.begin(), pnts.end());
    }
    return hull;
}
```