Contents

Basic

/.vimrc

```
set nu ai si cin ts=4 sw=4 sts=4

nmap #2 :! gedit %<.in %<*.in &<CR>
nmap #4 :! date > %<.pt; cat -n % > %<.pt; lpr %<.pt <
    CR>
nmap #9 :! clear ; g++ -std=c++11 -O2 -D AC -o %<.out %
     ; for i in %<*.in; do echo $i; ./%<.out < $i; echo
     ""; done <CR>
nmap #0 :! clear ; g++ -std=c++11 -O2 -D AC -o %<.out %
     ; ./%<.out <CR>
nmap <C-I> :! read -p "CASE:" CASE; gedit %<_$CASE.in <
    CR>
```

default code

```
#include <bits/stdc++.h>
using namespace std;

int main(){
#ifndef AC
  freopen("","r",stdin);
#endif
  ios_base::sync_with_stdio(0);
  cin.tie(0);
}
```

debug list

```
模板要記得 init
priority_queue 要清空
把邊界條件都加入測資
邊界條件 (過程溢位, 題目數據範圍), 會不會爆 long long
是否讀錯題目, 想不到時可以自己讀一次題目
環狀or凸包問題一定要每種都算n次
比較容易有問題的地方換人寫
注意公式有沒有推錯或抄錯
精度誤差 sqrt(大大的東西) + EPS
測試 %lld or %I64d
喇分 random_suffle 隨機演算法
```

Flow

Dinic

```
(a) Bounded Maxflow Construction:
1. add two node ss, tt
2. add_edge(ss, tt, INF)
3. for each edge u -> v with capacity [l, r]:
        add_edge(u, tt, l)
        add_edge(ss, v, l)
        add_edge(u, v, r-l)
4. see (b), check if it is possible.
5. answer is maxflow(ss, tt) + maxflow(s, t)
-------------------------------------------------------
(b) Bounded Possible Flow:
1. same construction method as (a)
2. run maxflow(ss, tt)
3. for every edge connected with ss or tt:
        rule: check if their rest flow is exactly 0
4. answer is possible if every edge do satisfy the rule
    ;
5. otherwise, it is NOT possible.
-------------------------------------------------------
(c) Bounded Minimum Flow:
1. same construction method as (a)
2. answer is maxflow(ss, tt)
-------------------------------------------------------
(d) Bounded Minimum Cost Flow:
* the concept is somewhat like bounded possible flow.
1. same construction method as (a)
2. answer is maxflow(ss, tt) + (∑ l * cost for every
    edge)
-------------------------------------------------------
(e) Minimum Cut:
1. run maxflow(s, t)
2. run cut(s)
3. ss[i] = 1: node i is at the same side with s.
-------------------------------------------------------

const long long INF = 1LL<<60;
struct Dinic {  //O(VVE), with minimum cut
    static const int MAXN = 5003;
    struct Edge{
        int u, v;
        long long cap, rest;
    };

    int n, m, s, t, d[MAXN], cur[MAXN];
    vector<Edge> edges;
    vector<int> G[MAXN];

    void init(){
        edges.clear();
        for ( int i = 0 ; i < MAXN ; i++ ) G[i].clear()
            ;
    }

    // min cut start
    bool side[MAXN];
    void cut(int u) {
        side[u] = 1;
        for ( int i : G[u] ) {
            if ( !side[ edges[i].v ] && edges[i].rest )
                cut(edges[i].v);
        }
    }
    // min cut end

    void add_edge(int u, int v, long long cap){
        edges.push_back( {u, v, cap, cap} );
        edges.push_back( {v, u, 0, 0LL} );
        m = edges.size();
        G[u].push_back(m-2);
        G[v].push_back(m-1);
    }

    bool bfs(){
        memset(d, -1, sizeof(d));
        queue<int> que;
        que.push(s); d[s]=0;
        while (!que.empty()){
            int u = que.front(); que.pop();
            for (int ei : G[u]){
                Edge &e = edges[ei];
                if (d[e.v] < 0 && e.rest > 0){
                    d[e.v] = d[u] + 1;
                    que.push(e.v);
                }
            }
        }
        return d[t] >= 0;
    }

    long long dfs(int u, long long a){
        if ( u == t || a == 0 ) return a;
        long long flow = 0, f;
```