目次

1 準備

1.1 init.el

linum は emacs24 のみ

```
;key
(keyboard-translate ?\C-h ?\C-?)
(global-set-key "\M-g" 'goto-line)

;tab
(setq-default indent-tabs-mode nil)
(setq-default tab-width 4)
(setq indent-line-function 'insert-tab)

;line number
(global-linum-mode t)
(setq linum-format "%4d ")
```

1.2 tpl.cpp

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <complex>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <map>
#include <queue>
#include <set>
#include <stack>
#include <string>
#include <vector>

#define rep(i,a) for(int i = 0;i < (a); i++)
#define repi(i,a,b) for(int i = (a); i < (b); i++)
#define repd(i,a,b) for(int i = (a); i >= (b); i--)
#define repit(i,a) for(__typeof((a).begin()) i = (a).begin(); i != (a).end(); i++)
#define all(u) (u).begin(),(u).end()
#define rall(u) (u).rbegin(),(u).rend()
#define UNIQUE(u) (u).erase(unique(all(u)),(u).end())
#define pb push_back
#define mp make_pair
#define INF 1e9
#define EPS 1e-10
#define PI acos(-1.0)

using namespace std;

typedef long long ll;
typedef vector<int> vi;

int main(){
}
```

2 文字列

2.1 Aho-Corasick 法

$O(N + M)$

```cpp
struct PMA{
    PMA* next[256];    //0 is failure link
    vector<int> matched;
    PMA(){memset(next, 0, sizeof(next));}
    ~PMA(){rep(i,256) if(next[i]) delete next[i];}
};
vector<int> set_union(const vector<int> &a,const vector<int> &b){
    vector<int> res;
    set_union(all(a), all(b), back_inserter(res));
    return res;
}
// patternからパターンマッチングオートマトンの生成
PMA *buildPMA(vector<string> pattern){
    PMA *root = new PMA, *now;
    root->next[0] = root;
    rep(i, patter.size()){
        now = root;
        rep(j, pattern[i].size()){
            if(now->next[(int)pattern[i][j]] == 0)
                now->next[(int)pattern[i][j]] = new PMA;
            now = now->next[(int)pattern[i][j]];
        }
        now->matched.push_back(i);
    }
    queue<PMA*> que;
    repi(i,1,256){
        if(!root->next[i]) root->next[i] = root;
        else {
            root->next[i]->next[0] = root;
            que.push(root->next[i]);
        }
    }
    while(!que.empty()){
        now = que.front(); que.pop();
        repi(i,1,256){
            if(now->next[i]){
                PMA *next = now->next[0];
                while(!next->next[i]) next = next->next[0];
                now->next[i]->next[0] = next->next[i];
                now->next[i]->matched = set_union(now->next[i]->matched, next->next[i]->
                    matched);
                que.push(now->next[i]);
            }
        }
    }
    return root;
}
void match(PMA* &pma, const string s, vector<int> &res){
    rep(i,s.size()){
        int c = s[i];
        while(!pma->next[c])
            pma = pma->next[0];
        pma = pma->next[c];
        rep(j,pma->matched.size())
            res[pma->matched[j]] = true;
    }
}
```