Contents

1 Basic

1.1 .vimrc

1.2 Increase Stack Size

```
//stack resize
asm( "mov %0,%%esp\n" ::"g"(mem+10000000) );
//change esp to rsp if 64-bit system

//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
  const rlim_t ks = 64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK, &rl);
  if(res==0){
    if(rl.rlim_cur<ks){
      rl.rlim_cur=ks;
      res=setrlimit(RLIMIT_STACK, &rl);
    }
  }
}
```

2 Graph

2.1 HLD

```
struct segment_tree{
    #define MAXN 100100
    #define right(x) x << 1 | 1
    #define left(x) x << 1
    int* arr;
    LL sum[4*MAXN];
    const int inf = 1e9;

    void pull(int ind) {
        sum[ind] = sum[right(ind)]+sum[left(ind)];
    };
    /// root => 1
    void build(int ind, int l, int r) {

        if( r - l == 1) {
            sum[ind] = 0;
            return;
        }
        int mid = (l+r)>>1;
        build( left(ind), l, mid );
        build( right(ind), mid, r );
        pull(ind);
    }
    LL query_sum(int ind, int L, int R, int ql, int qr)
         {
        if( L >= qr || R <= ql ) return 0;
        if( R <= qr && L >= ql ) {
            return sum[ind];
        }
        int mid = (L+R)>>1;
        return query_sum(left(ind), L, mid, ql, qr) +
            query_sum(right(ind), mid, R, ql, qr);
    }
    void modify(int ind, int L, int R, int ql, int qr,
        int x) {
        if( L >= qr || R <= ql ) return;
        if( R <= qr && L >= ql ) {
            sum[ind] = x;
            return;
        }
        int mid = (L+R)>>1;
        modify(left(ind), L, mid, ql, qr, x);
        modify(right(ind), mid, R, ql, qr, x);
        pull(ind);
    }
};

struct Tree{
    segment_tree seg;
    #define MAXN 100010
```

```
#define maxm (maxn<<1)
int n;
struct edge { int u, v; };
vector<edge> e;
void addedge(int x, int y) {
    G[x].pb( SZ(e) );
    G[y].pb( SZ(e) );
    e.pb( edge{x, y}  );
}
int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
/*size of subtree、index of max_son, parent index、
    depth*/
int link_top[MAXN],link[MAXN],Time;
/*chain top、index in segtree、time stamp*/
std::vector<int >G[MAXN];

void init(int N) {
    n = N;
    e.clear();
    for(int i = 1; i <= n; i++) G[i].clear();
}
void find_max_son(int x){
    siz[x]=1;
    max_son[x]=-1;
    for(int e_ind : G[x]) {
        int v = e[e_ind].u == x ? e[e_ind].v : e[
            e_ind].u ;
        if( v == pa[x] )continue;
        pa[v] = x; dep[v] = dep[x] + 1;
        find_max_son(v);
        if(max_son[x] == -1 || siz[v] > siz[max_son
            [x]])
            max_son[x] = v;
        siz[x] += siz[v];
    }
}
void build_link(int x,int top){
    link[x] = ++Time;/*記錄x點的時間戳*/
    link_top[x] = top;
    if(max_son[x] == -1)return;
    build_link( max_son[x], top);/*優先走訪最大孩子
        */
    for(int e_ind : G[x]) {
        int v = e[e_ind].u == x ? e[e_ind].v : e[
            e_ind].u ;
        if( v == max_son[x] || v == pa[x] )continue
            ;
        build_link(v, v);
    }
}
inline int lca(int a,int b){
    /*求LCA，可以在過程中對區間進行處理*/
    int ta=link_top[a],tb=link_top[b];
    while(ta != tb){
        if(dep[ta]<dep[tb]){
            std::swap(ta,tb);
            std::swap(a,b);
        }
        //interval [ link[ta], link[a] ]
        a = pa[ta];
        ta = link_top[a];
    }
    return dep[a] < dep[b] ? a:b;
}
int query(int a,int b){
    int ret = 0;
    int ta=link_top[a],tb=link_top[b];
    while(ta != tb){
        if(dep[ta]<dep[tb]){
            std::swap(ta,tb);
            std::swap(a,b);
        }
        //interval [ link[ta],link[a] ]
        a = pa[ta];
        ta = link_top[a];
    }
    if( a == b ) return ret;
    else {
        if(dep[a]>dep[b])
            swap(a,b);
        //interval [ link[a],link[b] ]
            // if operate on edges ==> [ link[ max_son[
                ta] ], link[b] ]
        }
    }
    /// Heavy Light Decomposition
    void HLD() {
        // root is indexed 1 here !
        find_max_son(1);
        build_link(1, 1);
    }
     void modify(int a, int b, int x) {
        // modify the path from a -> b to x
        //( which is [ link[a] .. link[b] ] on the
            segment tree)
        seg.modify(1, 1, n+1, link[a], link[b]+1, x);
        // this segment tree uses [ 1 ..n+1 )
    }
}tree;
```

2.2 Hungarian

```
// edge and node index starting from 0
// dfs  version below
/* to do
#define ___maxNodes
num_left = ?
*/
struct Edge {
    int from;
    int to;
    int weight;
    Edge(int f, int t, int w):from(f), to(t), weight(w)
        {}
};
vector<int> G[___maxNodes]; /* G[i] 存储顶点 i 出发的边
    的编号 */
vector<Edge> edges;
int num_nodes;
int num_left;
int num_right;
int num_edges;
int matching[___maxNodes]; /* matching result */
int check[___maxNodes];

bool dfs(int u) {
    for (auto i = G[u].begin(); i != G[u].end(); ++i) {
        // 对 u 的每个邻接点
        int v = edges[*i].to;
        if (!check[v]) {     // 要求不在交替路中
            check[v] = true; // 放入交替路
            if (matching[v] == -1 || dfs(matching[v]))
                {
                // 如果是未盖点，说明交替路为增广路，则
                    交换路径，并返回成功
                matching[v] = u;
                matching[u] = v;
                return true;
            }
        }
    }
    return false; // 不存在增广路，返回失败
}
int hungarian() {
    int ans = 0;
    memset(matching, -1, sizeof(matching));
    for (int u=0; u < num_left; ++u) {
        if (matching[u] == -1) {
            memset(check, 0, sizeof(check));
            if (dfs(u)) ++ans;
        }
    }
    return ans;
}
```

2.3 KM

```
// 最小帶權匹配~ km算法
```

```
//http://acm.csie.org/ntujudge/contest_view.php?id=836&
    contest_id=449
#include <bits/stdc++.h>
using namespace std;

struct bipartite {
    #define maxn 602
    #define INF 0xfffffff
    int sx[maxn], sy[maxn], mat[maxn][maxn];
    int x[maxn], y[maxn], link[maxn];
    int N, M, slack;

    int DFS(int t) {
        int tmp;
        sx[t] = 1;
        for (int i = 0; i < M; i++) {
            if (!sy[i]) {
                tmp = x[t] + y[i] - mat[t][i];
                if (tmp == 0) {
                    sy[i] = 1;
                    if (link[i] == -1 || DFS(link[i]))
                        {
                        link[i] = t;
                        return 1;
                    }
                }
                else if (tmp < slack) slack = tmp;
            }
        }
        return 0;
    }
    int KM() {
        for (int i = 0; i < N; i++) {
            x[i] = 0;
            for (int j = 0; j < M; j++) {
                if (mat[i][j] > x[i]) x[i] = mat[i][j];
            }
        }
        for (int j = 0; j < M; j++) { y[j] = 0; }
        memset(link, -1, sizeof(link));
        for (int i = 0; i < N; i++) {
            while (1) {
                memset(sx, 0, sizeof(sx));
                memset(sy, 0, sizeof(sy));
                slack = INF;
                if (DFS(i)) break;
                for (int j = 0; j < N; j++) {
                    if (sx[j]) x[j] -= slack;
                }
                for (int j = 0; j < M; j++) {
                    if (sy[j]) y[j] += slack;
                }
            }
        }

        int ans = 0;
        int cnt = 0;
        int t;
        for (int i = 0; i < M; i++)
        {
            t = link[i];
            if (t >= 0 && mat[t][i] != -INF)
            {
                cnt ++;
                ans += mat[t][i];
            }
        }
        // 最大權 : 沒有負號
        return -ans;
    }
    void init(int n,int m) {
        N = n, M = m;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < M; j++)
                mat[i][j] = -INF;
    }
    void input() {
        for(int i = 0; i < N; i++)
            for(int j =0;j<M;j++) {
                // fill in mat[i][j]
                // stands for the weighting , but
                    negative sign !
                // if 最大權 : 沒有負號
            }

    }
}km;

int main(){
    int n,E;
    while (scanf("%d", &n) != EOF)
    {
        km.init(n, n);
        km.input();
        cout<< km.KM() <<endl;
    }
    return 0;
}
```

3 Data Structure

3.1 Disjoint Set

4 Math

4.1 Prime Table

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

    int prime[1000000]={2,3,5,7};
    int sz=4;
    // biggest prime < ub
    int ub=(1<<20);

    int check(int num){
        int k = 0;
        for(k = 0; k < sz && prime[k]*prime[k] <= num;
            k++){
            if( num % prime[k]==0)  return 0;
        }
        return 1;
    }
    void buildprime(){
        int currentPrime=7;
        int j=4;
        for(sz=4,j=4; currentPrime<ub; sz++, j=6-j){
             currentPrime=currentPrime+j;
             if (check(currentPrime)) {
                prime[sz] = currentPrime;
             }
             else{
                sz--;
             }
        }
     }
}ptable;
```

4.2 Miller Rabin Prime Test

```
#include <cstdio>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;


long long mul(unsigned long long a, unsigned long long
    b, unsigned long long mod) {
    long long ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1,
        a = a >= mod ? a - mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod) ret -= mod;
```

```
        }
    }
    return ret;
}

long long mpow2(long long x, long long y, long long mod
    ) {
    long long ret = 1;
    while (y) {
        if (y&1)
            ret = mul(ret, x, mod);
        y >>= 1, x = mul(x, x, mod);
    }
    return ret % mod;
}
int isPrime(long long p, int it) { // implements by
    miller-babin
    if (p < 2)  return 0;
    if (p == 2) return 1;
    if (!(p&1)) return 0;
    long long q = p-1, a, t;
    int k = 0, b = 0;
    while (!(q&1))  q >>= 1, k++;

    while(it--) {
        a = rand()%(p-4) + 2;
        t = mpow2(a, q, p);
        b = (t == 1) || (t == p-1);
        for (int i = 1; i < k && !b; i++) {
            t = mul(t, t, p);
            if (t == p-1)
                b = 1;
        }
        if (b == 0)
            return 0;
    }

    return 1;
}


int main() {

    int testcase;
    scanf("%d", &testcase);
    while (testcase--) {
        long long n;
        scanf("%lld", &n);
        puts(isPrime(n, 1000)?"YES":"NO");
    }
    return 0;
}
```

4.3 Extended Euclidean Algorithm

```
/** normal gcd function using recursion **/
int gcd(int a, int b){
    if(b == 0) return a;
    return gcd(b, a%b);
}
// Find solution of ax + by = gcd(a, b)
// ps : x, y may be negative
int extgcd(int a, int b, int& x, int& y){
    int d = a;
    if(b != 0) {
        d = extgcd(b, a%b, y, x);
        y -= (a/b) * x;
    }else {
        x = 1, y = 0;
    }
    return d;
}
```

4.4 Gauss Elimination

```
// solving linear equations with gauss elimination
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

void print(vector< vector<double> > A) {
    int n = A.size();
    for (int i=0; i<n; i++) {
        for (int j=0; j<n+1; j++) {
            cout << A[i][j] << "\t";
            if (j == n-1) {
                cout << "| ";
            }
        }
        cout << "\n";
    }
    cout << endl;
}

vector<double> gauss(vector< vector<double> > A) {
    int n = A.size();

    for (int i=0; i<n; i++) {
        // Search for maximum in this column
        double maxEl = abs(A[i][i]);
        int maxRow = i;
        for (int k=i+1; k<n; k++) {
            if (abs(A[k][i]) > maxEl) {
                maxEl = abs(A[k][i]);
                maxRow = k;
            }
        }

        // Swap maximum row with current row (column by
            column)
        for (int k=i; k<n+1;k++) {
            double tmp = A[maxRow][k];
            A[maxRow][k] = A[i][k];
            A[i][k] = tmp;
        }

        // Make all rows below this one 0 in current
            column
        for (int k=i+1; k<n; k++) {
            double c = -A[k][i]/A[i][i];
            for (int j=i; j<n+1; j++) {
                if (i==j) {
                    A[k][j] = 0;
                } else {
                    A[k][j] += c * A[i][j];
                }
            }
        }
    }

    // Solve equation Ax=b for an upper triangular
        matrix A
    vector<double> x(n);
    for (int i=n-1; i>=0; i--) {
        x[i] = A[i][n]/A[i][i];
        for (int k=i-1;k>=0; k--) {
            A[k][n] -= A[k][i] * x[i];
        }
    }
    return x;
}

int main() {
    int n;
    cin >> n;

    vector<double> line(n+1,0);
    vector< vector<double> > A(n,line);

    // Read input data
    for (int i=0; i<n; i++) {
        for (int j=0; j<n; j++) {
            cin >> A[i][j];
        }
    }

    for (int i=0; i<n; i++) {
        cin >> A[i][n];
```

```
    }

    // Print input
    print(A);

    // Calculate solution
    vector<double> x(n);
    x = gauss(A);

    // Print result
    cout << "Result:\t";
    for (int i=0; i<n; i++) {
        cout << x[i] << " ";
    }
    cout << endl;
}
```

4.5 FFT

```
typedef long double ld;
/* N must be 2^k and greater than array.size()
 * FFT( a );
 * FFT( b );
 * for(int i = 0; i<N; ++i) c[i] = conj(a[i] * b[i]);
 * FFT( c );
 * for(int i = 0; i<N; ++i) c[i] = conj(c[i]);
 * for(int i = 0; i<N; ++i) c[i] /= N;
 */
void FFT(vector< complex<ld> >& v) {
    int N = v.size();
    for(int i = 1, j = 0; i<N; ++i) {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1) {
        ld w = -2.0*pi/k;
        complex<ld> deg(cos(w),sin(w));
        for(int j = 0; j<N; j+=k) {
            complex<ld> theta(1,0);
            for(int i = j; i<j+k/2; ++i) {
                complex<ld> a = v[i];
                complex<ld> b = v[i+k/2]*theta;
                v[i] = a+b;
                v[i+k/2] = (a-b);
                theta *= deg;
            }
        }
    }
}
```

4.6 NNT

```
/*
NTT( a );
NTT( b );
for(int i = 0; i<N: ++i)
    c[i] = (long long) a[i] * b[i] % mod;
NTT( c, true );
for(int i = 0; i<N; ++i)
    c[i] = (786433LL-12) * c[i] % mod;
*/

constexpr int mod = 786433;
constexpr int N = 65536;

void NTT(vector< int >& v, bool flag = false)
{
    for(int i = 1, j = 0; i<N; ++i)
    {
        for(int k = N>>1; !((j^=k)&k); k>>=1);
        if(i>j) swap(v[i],v[j]);
    }
    for(int k = 2; k<=N; k<<=1)
    {
        int deg = mypow(flag ? 524289 : 3, N / k);
        for(int j = 0; j<N; j+=k)
        {
            int theta = 1;
            for(int i = j; i<j+k/2; ++i)
            {
                int a = v[i];
                int b = (long long) v[i+k/2]*theta%mod;
                v[i] = (a+b) % mod;
                v[i+k/2] = (a-b+mod)%mod;
                theta = (long long) theta * deg % mod;
            }
        }
    }
}
```

5 string

5.1 Palindromic Tree

```
//coppied from internet's template
#include<vector>
struct palindromic_tree{
  struct node{
    int next[26],fail,len;/*這些是必要的元素*/
    int cnt,num;/*這些是額外維護的元素*/
    node(int l=0):fail(0),len(l),cnt(0),num(0){
      for(int i=0;i<26;++i)next[i]=0;
    }
  };
  std::vector<node >St;
  std::vector<char >s;
  int last,n;
  palindromic_tree():St(2),last(1),n(0){
    St[0].fail=1;
    St[1].len=-1;
    s.push_back(-1);
  }
  inline void clear(){
    St.clear();
    s.clear();
    last=1;
    n=0;
    St.push_back(0);
    St.push_back(-1);
    St[0].fail=1;
    s.push_back(-1);
  }
  inline int get_fail(int x){
    while(s[n-St[x].len-1]!=s[n])x=St[x].fail;
    return x;
  }
  inline void add(int c){
    s.push_back(c-='a');
    ++n;
    int cur=get_fail(last);
    if(!St[cur].next[c]){
      int now=St.size();
      St.push_back(St[cur].len+2);
      St[now].fail=St[get_fail(St[cur].fail)].next[c];
      /*不用擔心會找到空節點，由證明的過程可知*/
      St[cur].next[c]=now;
      St[now].num=St[St[now].fail].num+1;
    }
    last=St[cur].next[c];
    ++St[last].cnt;
  }
  inline void count(){/*cnt必須要在構造完後呼叫count()
      去計算*/
    std::vector<node>::reverse_iterator i=St.rbegin();
    for(;i!=St.rend();++i){
      St[i->fail].cnt+=i->cnt;
    }
  }
  inline int size(){/*傳回其不同的回文子串個數*/
    return St.size()-2;
  }
};
#endif
```

5.2 Suffix Array

```
//http://www.cnblogs.com/yefeng1627/p/3233611.html
/** template **/
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cassert>
// including iostream can't compile !??
//#include <iostream>
using namespace std;
/**
DA(倍增)算法求 SA[N] 与 Rank[N] (时间O(NlogN),空间O(N))

sa[i] : 表示 排在第i位的后缀 起始下标
Rank[i] : 表示后缀 suffix(i)排在第几
height[i] : 表示 suffix ( sa[i-1] ) 与 suffix ( sa[i] ) 的LCP 值
h[i]: 表示 suffix(i)与其排名前一位的 LCP值 => not included in template
LCP : longest common prefix
**/

const int N = 5*10001;
int cmp(int *r,int a,int b,int l){
    return (r[a]==r[b]) && (r[a+l]==r[b+l]);
}
// 用于比较第一关键字与第二关键字,
// 比较特殊的地方是,预处理的时候,r[n]=0(小于前面出现过的字符)

int wa[N],wb[N],ws[N],wv[N];
int Rank[N],height[N];
void DA(int *r,int *sa,int n,int m){ //此处N比输入的N要多1, 为人工添加的一个字符, 用于避免CMP时越界
    int i,j,p,*x=wa,*y=wb;
    for(i=0;i<m;i++) ws[i]=0;
    for(i=0;i<n;i++) ws[x[i]=r[i]]++;
    for(i=1;i<m;i++) ws[i]+=ws[i-1];
    for(i=n-1;i>=0;i--) sa[--ws[x[i]]] = i;
    for(j=1,p=1;p<n;j*=2,m=p)
    {
        for(p=0,i=n-j;i<n;i++) y[p++]=i;
        for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
        for(i=0;i<m;i++) ws[i]=0;
        for(i=0;i<n;i++) wv[i]=x[y[i]];
        for(i=0;i<n;i++) ws[wv[i]]++;
        for(i=1;i<m;i++) ws[i]+=ws[i-1];
        for(i=n-1;i>=0;i--) sa[--ws[wv[i]]]=y[i];
        for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;i++)
            x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
}

void calheight(int *r,int *sa,int n){ // 此处N为实际长度
    int i,j,k=0;          // height[]的合法范围为 1-N, 其中0是结尾加入的字符
    for(i=1;i<=n;i++) Rank[sa[i]]=i;  // 根据SA求Rank
    for(i=0;i<n; height[Rank[i++]] = k ) // 定义: h[i] = height[ Rank[i] ]
    for(k?k--:0,j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k++);
        //根据 h[i] >= h[i-1]-1 来优化计算height过程
}

char str[N];
int sa[N], r[N];
int main() {
    scanf("%s",str);
    int n = strlen(str);
    for(int i = 0;i < n; i++)
        r[i] = (int)str[i];
    r[n]=0;
    DA(r,sa,n+1,128);  //注意区分此处为n+1,因为添加了一个结尾字符用于区别比较
    calheight(r,sa,n);
//    /** demonstrate
    assert(sa[0] == n);
    for(int i = 0; i <= n; i++) printf("%d ", sa[i]);
    printf("\n");
    assert(Rank[n] == 0);
    for(int i = 0; i <= n; i++) printf("%d ", Rank[i]);
    printf("\n");
    //height[0] 沒有意義
    assert(height[1] == 0); //since sa[0] is 空字串
    printf("  ");
    for(int i = 1; i <= n; i++) printf("%d ", height[i]);
    printf("\n");
//    **/
}
```