Contents

1 Basic

1.1 Mergesort

```
long long sol(int L, int R) {
  if (R - L <= 1)return 0;
  int M = (R + L) / 2;
  long long ans = sol(L, M) + sol(M, R);
  int i = L, j = M, k = L;
  while (i < M || j < R) {
    if (i >= M)
      buf[k] = arr[j++];
    else if (j >= R)
      buf[k] = arr[i++];
    else {
      if (arr[i]<=arr[j])
        buf[k] = arr[i++];
      else {
        buf[k] = arr[j++];
        ans += M - i;
      }
    }
    k++;
  }
  for (int k = L; k < R; k++) arr[k] = buf[k];
  return ans;
}
```

2 Data and Structure

2.1 Disjoint Set

```
int p[N];
void init(){for (int i = 0; i < N; i++)p[i] = i;}
int find(int x)return x == p[x] ? x : find(p[x]);}
void Union(int a, int b){p[find(a)] = find(b);}
```

2.2 Segment Tree

```
int bulit(int L,int R,int x) {
  if(L==R)return heap[x - 1]=arr[L];
  int M=(L+R)>>1;
  return heap[x-1]=bulit(L, M, (x << 1))+bulit(M + 1, R
      , (x << 1) + 1);
}
void modify(int L,int R,int x,int a,int b,int mo) {
    if(b<L||R<a)return;
  if(L==R){heap[x-1]+=mo; return;}
  int M=(L+R)>>1;
  modify(L,M,(x<<1),a,b,mo);
  modify(M+1,R,(x<<1)+1,a,b,mo);
  heap[x - 1] += mo;
  return;
}
int quest(int L,int R,int x,int a,int b) {
    if(b<L||R<a)return 0;
  if(a<=L&&R<=b)return heap[x - 1];
  int M=(L+R)>>1;
  return quest(L,M,(x<<1),a,b)+quest(M+1,R,(x<<1)+1,a,b
      );
}
```

2.3 Treap

```
struct Treap{
  Treap *l, *r;
  int val, key, pri;
  Treap(int _val, int _key) :
    val(_val), key(_key), l(NULL), r(NULL), pri(rand())
        {}
  Treap(){};
};
Treap* merge(Treap* a, Treap* b){
  if (!a || !b)return a ? a : b;
  if (a->pri > b->pri){
    a->r = merge(a->r, b);
    return a;
  }else{
    b->l = merge(a, b->l);
    return b;
  }
}
void split(Treap* t, int k, Treap *&a, Treap *&b){
  if (!t)a = b = NULL;
  else if (t->key <= k){
    a = t;
    split(t->r, k, a->r, b);
  }else {
    b = t;
    split(t->l, k, a, b->l);
  }
  return;
}
Treap* insert(Treap* t, int k){
  Treap *tl, *tr;
  split(t, k, tl, tr);
  return merge(tl, merge(new Treap(k, ti++), tr));
}
Treap* remove(Treap* t, int k){
  Treap *tl, *tr;
  split(t, k - 1, tl, t);
  split(t, k, t, tr);
  return merge(tl, tr);
}
```

3 DP

3.1 LIS

```
void print_lis(int v){
    if(pre[v])print_lis(pre[v]);
    cout<<a[v]<<'\n';
}
int main(){
    for(ai=0;cin>>a[++ai];);
    pre[1]=0; b[bi=1]=1;
    for(int i=2;i<=ai;i++){
        if(a[i]>a[b[bi]]){
            b[++bi]=i;
            pre[i]=b[bi-1];
        }else{
            int id=int(lower_bound(b,b+bi,a[i])-b);
            b[id]=i;
            pre[i]=b[id-1];
        }
    }
    cout<<bi<<"\n-\n";
    print_lis(b[bi]);
}
```

3.2 TSP

```
bool b[N];
int n,dis[N][N],dp[N][100000];
void btb(int &x){
  x=0;
  for(int i=0,j=1;i<n;i++,j*=2)x+=b[i]*j;
  return;
}
int main(){
  memset(dp,0,sizeof(dp));
    for(int i=1,st;i<=n;i++){//st:state
        for(int jj=0;jj<n;jj++)b[n-jj-1]=(jj<i);
        do{
            btb(st);
            for(int x=0;x<n;x++){
                if(!b[x])continue;
                if(i==1)dp[x][st]=dis[x][0];
                for(int y=0;y<n;y++){
                    if(x!=y&&b[y]&&(dp[x][st]==0||dp[x
                        ][st]>dp[y][st-(1<<x)]+dis[y][x
                        ])){
                        dp[x][st]=dp[y][st-(1<<x)]+dis[
                            y][x];
                    }
                }
            }
        }while(next_permutation(b,b+n));
    }
    cout<<dp[0][(1<<n)-1]<<'\n';
}
```

4 Graph

4.1 Articulation Point

```
vector<int>v[N],bcc[N];//clear
LL dep[N],low[N],bccno[N],time_cnt,bcc_cnt;//set dep
    low -1 else 0
bitset<N>is_AP;//0
struct Edge{int s,t;};
stack<Edge>st;//clear
int dfs(int s,int fa){
    int child=0;
    dep[s]=low[s]=time_cnt++;
    for(auto t:v[s]){
        Edge e=(Edge){s,t};
        if(dep[t]==-1){
            st.push(e);
            child++;
            dfs(t,s);
            low[s]=min(low[s],low[t]);
            if(dep[s]<=low[t]){
                is_AP[s]=1;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                while(1){
                    Edge x=st.top(); st.pop();
                    if(bccno[x.s]!=bcc_cnt){
                        bcc[bcc_cnt].push_back(x.s);
                        bccno[x.s]=bcc_cnt;
                    }
                    if(bccno[x.t]!=bcc_cnt){
                        bcc[bcc_cnt].push_back(x.t);
                        bccno[x.t]=bcc_cnt;
                    }
                    if(x.s==s&&x.t==t)break;
                }
            }
        }else if(low[s]>dep[t]){
            st.push(e);
            low[s]=dep[t];
        }
    }
    if(fa<0&&child==1)is_AP[s]=0;
    return low[s];
}
```

4.2 Convex Hull

```
struct loc {
  int x, y;
  loc() {};
  loc(int x, int y): x(x), y(y) {}
  bool operator <(const loc& b)const {return x != b.x ?
       x < b.x : y < b.y;}
  bool operator ==(const loc& b)const {return x == b.x
      && y == b.y;}
  loc operator -(const loc& b)const {return loc(x - b.x
      , y - b.y);}
  int cross(const loc& b)const {return x * b.y - y * b.
      x;}
  int dis(loc a, loc b) {return (x - b.x) * (x - b.x) +
       (y - b.y) * (y - b.y);}
};
vector<loc>p, p1;
int n;
void convexhull() {
  sort(p.begin(), p.end());
  p.erase(unique(p.begin(), p.end()), p.end());
  p1.clear();
  p1.resize(p.size());
  int m = 0;
  for (int i = 0; i < p.size(); i++) {
    while (m > 1 && (p1[m - 1] - p1[m - 2]).cross(p[i]
        - p1[m - 2]) <= 0)m--;
    p1[m++] = p[i];
  }
  int k = m;
  for (int i = p.size() - 2; i >= 0; i--) {
    while (m > k && (p1[m - 1] - p1[m - 2]).cross(p[i]
        - p1[m - 2]) <= 0)m--;
    p1[m++] = p[i];
  }
  if (n > 1)m--;
  p1.resize(m);
}
```

4.3 Dinic

```
struct dinic{
  struct Edge{int v,f,re;}; //residual flow
  int n, s, t, level[M], now[M];
  vector<Edge> e[M];
```

```
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i = 0; i <= n; i++)e[i].clear();
  }
  void add_edge(int u, int v, int f){
    e[u].push_back({ v, f, e[v].size() });
    e[v].push_back({ u, f, e[u].size() - 1 });
  }
  bool bfs(){
    fill(level, level + n + 1, -1);
    queue<int> q;
    q.push(s); level[s] = 0;
    while (!q.empty()){
      int u = q.front(); q.pop();
      for (auto it : e[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u] + 1;
          q.push(it.v);
        }
      }
    }
    return level[t] != -1;
  }
  int dfs(int u, int nf){
    if (u == t)return nf;
    int res = 0;
    while (now[u] < e[u].size()){
      Edge &it = e[u][now[u]];
      if (it.f>0 && level[it.v] == level[u] + 1){
        int tf = dfs(it.v, min(nf, it.f));
        res += tf; nf -= tf; it.f -= tf;
        e[it.v][it.re].f += tf;
        if (nf == 0)return res;
      }
      else now[u]++;
    }
    if (!res)level[u] = -1;
    return res;
  }
  int flow(int res = 0){
    while (bfs()){
      int temp;
      memset(now, 0, sizeof(now));
      while (temp = (dfs(s, INF))){
        res += temp;
      }
    }
    return res;
  }
};
```

4.4 Longest Common Ancestor

```
void preprocess() {
  for (int i = 1; i <= 25; i++) {
    for (int j = 1; j <= n; j++) {
      if (par[j][i - 1] == -1 || par[par[j][i - 1]][i -
          1] == -1)continue;
      par[j][i] = par[par[j][i - 1]][i - 1];
    }
  }
}
```

4.5 KM

```
bool match(int i) {
  vx[i] = true;
  for (int j = 1; j <= n; j++) {
    if ((fabs(Lx[i] + Ly[j] - w[i][j]) < 1e-9) && !vy[j
        ]) {
      vy[j] = 1;
      if (!Left[j] || match(Left[j])) {
        Left[j] = i;
        return true;
      }
    }
  }
  return false;
}
void update() {
  double a = 1e30;
  for (int i = 1; i <= n; i++) {
    if (vx[i])for (int j = 1; j <= n; j++) {
        if (!vy[j])a = min(a, Lx[i] + Ly[j] - w[i][j]);
      }
  }
  for (int i = 1; i <= n; i++) {
    if (vx[i])Lx[i] -= a;
    if (vy[i])Ly[i] += a;
  }
}
void KM() {//reset lx ly left
  for (int i = 1; i <= n; i++) {
    while (1) {
      vx.reset(); vy.reset();
      if (match(i))break;
      update();
    }
  }
}
```

5 Number

5.1 Catalan

```
long long f[N]={1},i,t,p;
int main(){
    for(int i=1;i<=100;i++){
        f[i]=f[i-1]*(4*i-2)%mod;
        for(t=i+1,p=mod-2;p;t=(t*t)%mod,p>>=1LL){
            if(p&1){f[i]*=t;f[i]%=mod;}
        }
    }
}
```

5.2 Extend Euclidean.cpp

```
int extgcd(int a,int b,int &x,int &y){
    int d=a;
    if(b){d=extgcd(b,a%b,y,x),y-=(a/b)*x;}
    else x=1,y=0;
    return d;
}//ax+by=1 ax同餘 1 mod b
```

5.3 GaussElimination

```
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;
      }
    }
  }
}
```

5.4 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;
  }
};
```

5.5 Prime table

```
bitset<N>is_notp;
vector<int>p;//prime
void PrimeTable(){
    is_notp.reset();
  is_notp[0] = is_notp[1] = 1;
  for (int i = 2; i < N; i++){
    if (is_notp[i])continue;
    p.push_back(i);
    for (int j=0;i*p[j]<N&&j<p.size();j++){
      is_notp[i*p[j]] = 1;
      if(i%p[j]==0)break;
    }
  }
}
```

6 String

6.1 KMP

```
void bulid_fail_funtion(string B, int *fail){
  int len = B.length(), current_pos;
  current_pos = fail[0] = -1;
  for (int i = 1; i<len; i++){
    while (current_pos != -1 && B[current_pos + 1] != B
        [i]){
      current_pos = fail[current_pos];
    }
    if (B[current_pos + 1] == B[i])current_pos++;
    fail[i] = current_pos;
  }
}
void match(string A, string B, int *fail){
  int lenA = A.length(), lenB = B.length();
  int current_pos = -1;
  for (int i = 0; i<lenA; i++){
    while (current_pos != -1 && B[current_pos + 1] != A
        [i]){
      current_pos = fail[current_pos];
    }
    if (B[current_pos + 1] == A[i])current_pos++;
    if (current_pos == lenB - 1){//match! A[i-lenB+1,i
        ]=B
      current_pos = fail[current_pos];
    }
  }
}
int main(){
  int t, i;
  string s;
  for (i = 0, cin >> t; i<t; i++){
    cin >> s;
    int fail[N];
    bulid_fail_funtion(s, fail);
    int p = s.length() - 1;
    if (fail[p] != -1 && (p + 1) % (p - fail[p]) == 0)
        printf("%d\n", p - fail[p]);
    else printf("%d\n", p + 1);
  }
}
```

6.2 Trie

```
//init sz=1 trie[0]=0
void insert(string s){
    int u=0,v;
    for(int i=0;i<r.size();i++){
        v=r[i]-'a';
        if(!trie[u][v]){
            memset(trie[sz],0,sizeof(trie[sz]));
            val[sz]=0;
            trie[u][v]=sz++;
        }
        u=trie[u][v];
    }
    val[u]=1;
    return;
}
void search(string s,int i){
    int u=0,v;
    dp[i]=0;
    for(int j=i;j<s.size();j++){
        v=s[j]-'a';
        if(!trie[u][v])return;
        u=trie[u][v];
        if(val[u])dp[i]=(dp[i]+dp[j+1])%MOD;
    }
    return;
}
```

6.3 Zvalue

```
void z_value(){
  int lens = s.size(), l = 0, r = 0;
  z[0] = 0;
  for (int i = 1; i < lens; i++){
    if (i>r)z[i] = 0;
    else{
      int ip = i - l;
      if (ip + z[ip] < z[l])z[i] = z[ip];
      else z[i] = r - l + 1;
    }
    while (i + z[i] < lens&&s[i + z[i]] == s[z[i]])z[i
        ]++;
    if (i + z[i] - 1 > r){
      l = i;
      r = l + z[i] - 1;
    }
  }
}
```