目录

[1. MATH 2](#_Toc427624227)

[a) euler function 2](#_Toc427624228)

[b) exgcd 2](#_Toc427624229)

[c) fermat’s little 2](#_Toc427624230)

[d) Chinese remainder 3](#_Toc427624231)

[e) fft 4](#_Toc427624232)

[f) else 6](#_Toc427624233)

[2. GRATH 6](#_Toc427624234)

[a) Prim 6](#_Toc427624235)

[b) Kruskal 7](#_Toc427624236)

[c) Dijstra 7](#_Toc427624237)

[d) Spfa 8](#_Toc427624238)

[e) Two-sat 9](#_Toc427624239)

[f) MCMF 10](#_Toc427624240)

[g) Max-flow 12](#_Toc427624241)

[h) LCA 13](#_Toc427624242)

[i) Hungarian-dfs 15](#_Toc427624243)

[3. DATA STRUCTURE 16](#_Toc427624244)

[a) Kd-tree 16](#_Toc427624245)

[b) Tree array 17](#_Toc427624246)

[c) Sa array 18](#_Toc427624247)

[4. GEOMETRY 19](#_Toc427624248)

[5. ALGORITHM 21](#_Toc427624249)

[a) RMQ 21](#_Toc427624250)

[b) Manacher 21](#_Toc427624251)

[c) Ac automatic 22](#_Toc427624252)

[d) Kmp 24](#_Toc427624253)

[6. STL 25](#_Toc427624254)

1. MATH
   1. euler function

1 int euler(int x){

2 int ans = x, m = n;

3 for(int i = 2; i \* i <= x; i++){

4 if(x % i == 0){

5 ans = ans / i \* (i - 1);

6 while(x % i == 0){

7 x /= i;

8 }

9 }

10 }

11 if(x > 1){

12 ans = ans / x \* (x - 1);

13 }

* 1. exgcd

1 int gcd(int a, int b, int& x, int& y)

2 {

3 if(b == 0){

4 x = 1;

5 y = 0;

6 return a;

7 }

8 else{

9 int temp = gcd(b, a % b, x, y);

10 int t = y;

11 y = x - y \* (a / b);

12 x = t;

13 return temp;

14 }

15 }

* 1. fermat’s little

if gcd(a,p)=1，then a(p-1) ≡1(mod p)

1 #include <iostream>

2 #include <cstdio>

3 #define MAXN 2000001

4 #define MOD 1000000007

5 using namespace std;

6

7 long long f[MAXN + 5];

8 long long gcd(long long a, long long b, long long& x, long long& y)

9 {

10 if(b == 0){

11 x = 1;

12 y = 0;

13 return a;

14 }

15 else{

16 long long temp = gcd(b, a % b, x, y);

17 long long t = y;

18 y = x - y \* (a / b);

19 x = t;

20 return temp;

21 }

22 }

23

24 int main()

25 {

26 f[1] = 1;

27 for(long long i = 2; i <= MAXN; i++){

28 f[i] = (f[i - 1] \* (4 \* i - 2)) % MOD;

29 gcd(i + 1, MOD, x, y);

30 if(x < 0){

31 x = MOD - (-x) % MOD;

32 }

33 x = x % MOD;

34 f[i] = (f[i] \* x) % MOD;

35 }

* 1. Chinese remainder

1 #include <iostream>

2 #include <cstdio>

3 #define M 11

4 using namespace std;

5

6 long long a[M], b[M];

7 long long x, y;

8 bool flag;

9 long long gcd(long long a, long long b)

10 {

11 if (b == 0){

12 return a;

13 }

14 return gcd(b, a % b);

15 }

16

17 long long exgcd(long long a, long long b, long long & x, long long & y)

18 {

19 if (b == 0){

20 x = 1;

21 y = 0;

22 return a;

23 }

24 else{

25 long long temp = exgcd(b, a % b, x, y);

26 long long t = y;

27 y = x - y \* (a / b);

28 x = t;

29 return temp;

30 }

31 }

32 int main()

33 {

34 long long T, d, n, m, t, A, B;

35 scanf("%I64d", &T);

36 for (int cas = 1; cas <= T; cas++){

37 flag = false;

38 scanf("%I64d%I64d", &n, &m);

39 for (int i = 1; i <= m; i++){

40 scanf("%d", &a[i]);

41 //t = (t \* a[i]) / gcd(t, a[i]);

42 }

43 for (int i = 1; i <= m; i++){

44 scanf("%I64d", &b[i]);

45 }

46 A = a[1], B = b[1];

47 for (int i = 2; i <= m; i++){

48 d = exgcd(A, a[i], x, y);

49 if ((b[i] - B) % d != 0){

50 flag = true;

51 break;

52 }

53 x = (b[i] - B) / d \* x;

54 y = a[i] / d;

55 x = (x % y + y) % y;

56 B = x \* A + B;

57 A = (A \* a[i]) / d;

58 B = (B % A + A) % A;

59 }

60 if (B > n || flag){

61 printf("0\n");

62 }

63 else{

64 t = 1 + (n - B) / A;

65 if (B == 0){

66 --t;

67 }

68 printf("%I64d\n", t);

69 }

70 }

71 }

* 1. fft

1 #include <iostream>

2 #include <stdio.h>

3 #include <cmath>

4 #include <algorithm>

5 #include <cstring>

6 #include <vector>

7 using namespace std;

8 #define N 50500\*2

9 const double PI = acos(-1.0);

10 struct Vir

11 {

12 double re, im;

13 Vir(double \_re = 0., double \_im = 0.) :re(\_re), im(\_im){}

14 Vir operator\*(Vir r) { return Vir(re\*r.re - im\*r.im, re\*r.im + im\*r.re); }

15 Vir operator+(Vir r) { return Vir(re + r.re, im + r.im); }

16 Vir operator-(Vir r) { return Vir(re - r.re, im - r.im); }

17 };

18 void bit\_rev(Vir \*a, int loglen, int len)

19 {

20 for (int i = 0; i < len; ++i)

21 {

22 int t = i, p = 0;

23 for (int j = 0; j < loglen; ++j)

24 {

25 p <<= 1;

26 p = p | (t & 1);

27 t >>= 1;

28 }

29 if (p < i)

30 {

31 Vir temp = a[p];

32 a[p] = a[i];

33 a[i] = temp;

34 }

35 }

36 }

37 void FFT(Vir \*a, int loglen, int len, int on)

38 {

39 bit\_rev(a, loglen, len);

40

41 for (int s = 1, m = 2; s <= loglen; ++s, m <<= 1)

42 {

43 Vir wn = Vir(cos(2 \* PI\*on / m), sin(2 \* PI\*on / m));

44 for (int i = 0; i < len; i += m)

45 {

46 Vir w = Vir(1.0, 0);

47 for (int j = 0; j < m / 2; ++j)

48 {

49 Vir u = a[i + j];

50 Vir v = w\*a[i + j + m / 2];

51 a[i + j] = u + v;

52 a[i + j + m / 2] = u - v;

53 w = w\*wn;

54 }

55 }

56 }

57 if (on == -1)

58 {

59 for (int i = 0; i < len; ++i) a[i].re /= len, a[i].im /= len;

60 }

61 }

62 char a[N \* 2], b[N \* 2];

63 Vir pa[N \* 2], pb[N \* 2];

64 int ans[N \* 2];

65 int main()

66 {

67 while (scanf("%s%s", a, b) != EOF)

68 {

69 int lena = strlen(a);

70 int lenb = strlen(b);

71 int n = 1, loglen = 0;

72 while (n < lena + lenb) n <<= 1, loglen++;

73 for (int i = 0, j = lena - 1; i < n; ++i, --j)

74 pa[i] = Vir(j >= 0 ? a[j] - '0' : 0., 0.);

75 for (int i = 0, j = lenb - 1; i < n; ++i, --j)

76 pb[i] = Vir(j >= 0 ? b[j] - '0' : 0., 0.);

77 for (int i = 0; i <= n; ++i) ans[i] = 0;

78

79 FFT(pa, loglen, n, 1);

80 FFT(pb, loglen, n, 1);

81 for (int i = 0; i < n; ++i)

82 pa[i] = pa[i] \* pb[i];

83 FFT(pa, loglen, n, -1);

84

85 for (int i = 0; i < n; ++i) ans[i] = pa[i].re + 0.5;

86 for (int i = 0; i<n; ++i) ans[i + 1] += ans[i] / 10, ans[i] %= 10;

87

88 int pos = lena + lenb - 1;

89 for (; pos>0 && ans[pos] <= 0; --pos);

90 for (; pos >= 0; --pos) printf("%d", ans[pos]);

91 puts("");

92 }

93 return 0;

94 }

* 1. else

1. GRATH
   1. Prim

1 int prim() {

2 memset(vis,0,sizeof(vis));

3 int i;

4 int maxedge=0;

5 for (i = 1; i <= n; i++) {

6 dis[i]= value[1][i];

7 }

8 dis[1] = 0;

9 vis[1] = true;

10 for (i = 2; i <= n; i++) {

11 int temp = inf;

12 int mark;

13 for (int j = 1; j <= n; j++) {

14 if (!vis[j] && dis[j] < temp) {

15 temp = dis[j];

16 mark = j;

17 }

18 }

19 if(dis[mark]>maxedge)

20 maxedge=dis[mark];

21 vis[mark]=true;

22 for (int j = 1; j <= n; j++) {

23 if (!vis[j]&&dis[j]>value[mark][j])

24 dis[j] = value[mark][j];

25 }

26 }

27 return maxedge;

28 }

* 1. Kruskal
  2. Dijstra

1 struct Edge

2 {

3 int from, to, dist;

4 Edge(int from, int to, int dist):from(from), to(to), dist(dist){};

5 };

6 struct HeapNode

7 {

8 int d, u;

9 HeapNode(int d, int u):d(d), u(u){};

10 bool operator <(const HeapNode& rhs) const{

11 return d > rhs.d;

12 }

13 };

14 struct Dijstra

15 {

16 int n, m;

17 vector<Edge> edges;

18 vector<int> G[MAXN];

19 bool done[MAXN];

20 int d[MAXN];

21 int p[MAXN];

22

23 void init(int n){

24 this->n = n;

25 for(int i = 0; i <= n; i++){

26 G[i].clear();

27 road[i].clear();

28 }

29 edges.clear();

30 }

31

32 void AddEdge(int from, int to, int dist){

33 edges.push\_back(Edge(from, to, dist));

34 m = edges.size();

35 G[from].push\_back(m - 1);

36 }

37

38 void dijstra(int s){

39 priority\_queue<HeapNode> Q;

40 for(int i = 0; i <= n; i++){

41 d[i] = INF;

42 }

43 d[s] = 0;

44 memset(done, 0, sizeof(done));

45 Q.push(HeapNode(0, s));

46 while(!Q.empty()){

47 HeapNode x = Q.top();

48 Q.pop();

49 int u = x.u;

50 if(done[u]) continue;

51 done[u] = true;

52 for(int i = 0; i < G[u].size(); i++){

53 Edge& e = edges[G[u][i]];

54 if(d[e.to] > d[u] + e.dist){

55 d[e.to] = d[u] + e.dist;

56 p[e.to] = G[u][i];

57 Q.push(HeapNode(d[e.to], e.to));

58 }

59 }

60 }

61 }

62 };

* 1. Spfa

1 int spfa(int s)

2 {

3 queue <int> q;

4 memset(d, INF, sizeof(d));

5 d[s] = 0;

6 memset(cnt, 0, sizeof(cnt));

7 memset(vis, 0, sizeof(vis));

8 q.push(s);

9 vis[s] = 1;

10 while (!q.empty())

11 {

12 int x;

13 x = q.front();

14 q.pop();

15 while (no[x]){

16 x = q.front();

17 q.pop();

18 }

19 vis[x] = 0;

20 for (int i = 0; i < G[x].size(); i++)

21 {

22 int y = G[x][i].v;

23 if (d[x] + G[x][i].w < d[y])

24 {

25 d[y] = d[x] + G[x][i].w;

26 if (!vis[y])

27 {

28 vis[y] = 1;

29 q.push(y);

30 }

31 }

32 }

33 }

34 }

* 1. Two-sat

1 struct TwoSat{

2 int n;

3 vector<int> G[MAXN\*2];

4 bool mark[MAXN\*2];

5 int S[MAXN\*2], c;

6

7 bool dfs(int x){

8 if(mark[x^1]) return false;

9 if(mark[x]) return true;

10 mark[x] = true;

11 S[c++] = x;

12 for(int i = 0; i < G[x].size(); i++){

13 if(!dfs(G[x][i])) return false;

14 }

15 return true;

16 }

17

18 void init(int n){

19 this->n = n;

20 for(int i = 0; i < n \* 2; i++){

21 G[i].clear();

22 }

23 memset(mark, 0, sizeof(mark));

24 }

25

26 void add\_clause(int x, int xval, int y, int yval){

27 x = x \* 2 + xval;

28 y = y \* 2 + yval;

29 G[x^1].push\_back(y);

30 G[y^1].push\_back(x);

31 }

32

33 bool solve(){

34 for(int i = 0; i < n \* 2; i += 2){

35 if(!mark[i] && !mark[i + 1]){

36 c = 0;

37 if(!dfs(i)){

38 while(c > 0){

39 mark[S[--c]] = false;

40 }

41 if(!dfs(i + 1)){

42 return false;

43 }

44 }

45 }

46 }

47 return true;

48 }

49 };

* 1. MCMF

1 #include <iostream>

2 #include <string.h>

3 #include <stdio.h>

4 #include <algorithm>

5 #include <queue>

6 #define V 10100

7 #define E 1000100

8 #define inf 99999999

9 using namespace std;

10 int vis[V];

11 int dist[V];

12 int pre[V];

13

14 struct Edge{

15 int u,v,c,cost,next;

16 }edge[E];

17 int head[V],cnt;

18

19 void init(){

20 cnt=0;

21 memset(head,-1,sizeof(head));

22 }

23 void addedge(int u,int v,int c,int cost)

24 {

25 edge[cnt].u=u;edge[cnt].v=v;edge[cnt].cost=cost;

26 edge[cnt].c=c;edge[cnt].next=head[u];head[u]=cnt++;

27

28 edge[cnt].u=v;edge[cnt].v=u;edge[cnt].cost=-cost;

29 edge[cnt].c=0;edge[cnt].next=head[v];head[v]=cnt++;

30 }

31

32 bool spfa(int begin,int end){

33 int u,v;

34 queue<int> q;

35 for(int i=0;i<=end+2;i++){

36 pre[i]=-1;

37 vis[i]=0;

38 dist[i]=inf;

39 }

40 vis[begin]=1;

41 dist[begin]=0;

42 q.push(begin);

43 while(!q.empty()){

44 u=q.front();

45 q.pop();

46 vis[u]=0;

47 for(int i=head[u];i!=-1;i=edge[i].next){

48 if(edge[i].c>0){

49 v=edge[i].v;

50 if(dist[v]>dist[u]+edge[i].cost){

51 dist[v]=dist[u]+edge[i].cost;

52 pre[v]=i;

53 if(!vis[v]){

54 vis[v]=true;

55 q.push(v);

56 }

57 }

58 }

59 }

60 }

61 return dist[end]!=inf;

62 }

63

64 int MCMF(int begin,int end){

65 int ans=0,flow;

66 int flow\_sum=0;

67 while(spfa(begin,end)){

68 flow=inf;

69 for(int i=pre[end];i!=-1;i=pre[edge[i].u])

70 if(edge[i].c<flow)

71 flow=edge[i].c;

72 for(int i=pre[end];i!=-1;i=pre[edge[i].u]){

73 edge[i].c-=flow;

74 edge[i^1].c+=flow;

75 }

76 ans+=dist[end];

77 flow\_sum += flow;

78 }

79 //cout << flow\_sum << endl;

80 return ans;

81 }

82

83 int main()

84 {

85 //freopen("in.txt","r",stdin);

86 int n,m,a,b,c;

87 while(scanf("%d%d",&n,&m)!=EOF){

88 init();

89 addedge(0,1,2,0);

90 addedge(n,n+1,2,0);

91 for(int i=1;i<=m;i++){

92 scanf("%d%d%d",&a,&b,&c);

93 addedge(a,b,1,c);

94 addedge(b,a,1,c);

95 }

96 printf("%d\n",MCMF(0,n+1));

97 }

98 return 0;

99 }

* 1. Max-flow

1 struct Edge{

2 int from, to, cap, flow;

3 //Edge(int u, int v, int c, int f) :from(u), to(v), cap(c), flow(f){};

4 };

5 bool comp(const Edge& a, const Edge& b){

6 return (a.from < b.from || (a.from == b.from && a.to < b.to));

7 }

8 struct Dinic{

9 int n, m, i, s, t;

10 Edge e;

11 vector<Edge> edges;

12 vector<int> G[MAXN];

13 int d[MAXN], cur[MAXN];

14 bool vis[MAXN];

15 void init(int n){

16 this->n = n;

17 for (i = 0; i <= n; i++){

18 G[i].clear();

19 }

20 edges.clear();

21 }

22 void AddEdge(int from, int to, int cap){

23 edges.push\_back(Edge{ from, to, cap, 0 });

24 edges.push\_back(Edge{ to, from, 0, 0 });

25 m = edges.size();

26 G[from].push\_back(m - 2);

27 G[to].push\_back(m - 1);

28 }

29 bool BFS(){

30 memset(vis, 0, sizeof(vis));

31 queue<int> Q;

32 Q.push(s);

33 d[s] = 0;

34 vis[s] = 1;

35 while (!Q.empty()){

36 int x = Q.front();

37 Q.pop();

38 for (i = 0; i < G[x].size(); i++){

39 Edge& e = edges[G[x][i]];

40 if (!vis[e.to] && e.cap > e.flow){

41 vis[e.to] = true;

42 d[e.to] = d[x] + 1;

43 Q.push(e.to);

44 }

45 }

46 }

47 return vis[t];

48 }

49 int DFS(int x, int a){

50 if (x == t || a == 0) return a;

51 int flow = 0, f;

52 for (int& i = cur[x]; i < G[x].size(); i++){

53 Edge& e = edges[G[x][i]];

54 if (d[x] + 1 == d[e.to] && (f = DFS(e.to, min(a, e.cap - e.flow))) > 0){

55 e.flow += f;

56 edges[G[x][i] ^ 1].flow -= f;

57 flow += f;

58 a -= f;

59 if (a == 0) break;

60 }

61 }

62 return flow;

63 }

64 int MaxFlow(int s, int t, int need){

65 int flow = 0;

66 this->s = s;

67 this->t = t;

68 while (BFS()){

69 memset(cur, 0, sizeof(cur));

70 flow += DFS(s, INF);

71 if (flow > need) return flow;

72 }

73 return flow;

74 }

75 bool checkFull(int s){

76 for (int i = 0; i < G[s].size(); i++){

77 if (edges[G[s][i]].flow != edges[G[s][i]].cap){

78 return false;

79 }

80 }

81 return true;

82 }

83 };

* 1. LCA

1 #include <iostream>

2 #include <cstdio>

3 #include <algorithm>

4 #define LL long long

5 #define MAXN 10005

6 #define MAXM 30005

7 using namespace std;

8

9 int euler[MAXM], deep[MAXM], pos[MAXN];

10 int f[20][MAXN];

11 vector<int>G[MAXN];

12 bool vis[MAXN];

13 int top;

14 int cnt[MAXN];

15 void dfs(int t, int x)

16 {

17 if (pos[x] == -1)

18 pos[x] = top;

19 deep[top] = t;

20 euler[top++] = x;

21

22 for (int i = 0; i < G[x].size(); i++)

23 {

24 dfs(t + 1, G[x][i]);

25 deep[top] = t;

26 euler[top++] = x;

27 }

28 }

29

30 void rmq(int n)

31 {

32 for (int i = 1; i <= n; i++)

33 f[0][i] = deep[i];

34 for (int j = 1; j <= (int)(log((double)n) / log(2.0)); j++){

35 for (int i = 1; i <= n - (1 << j) + 1; i++){

36 f[j][i] = min(f[j - 1][i], f[j - 1][i + (1 << (j - 1))]);

37 }

38 }

39 }

40

41 int get(int x, int y)

42 {

43 if (x > y){

44 swap(x, y);

45 }

46 int k = (int)(log((double)(y - x + 1.0)) / log(2.0));

47 int temp = min(f[k][x], f[k][y - (1 << k) + 1]);

48 for (int i = x; i <= y; i++)

49 if (deep[i] == temp)

50 return euler[i];

51 }

52

53 int main()

54 {

55 int n;

56 int a, num, b;

57 int root;

58 int m, x, y;

59 int T;

60 scanf("%d", &T);

61 for (int cas = 1; cas <= T; cas++){

62 scanf("%d", &n);

63 top = 1;

64 memset(pos, -1, sizeof(pos));

65 memset(cnt, 0, sizeof(cnt));

66 memset(vis, 0, sizeof(vis));

67 for (int i = 1; i <= n; i++)

68 G[i].clear();

69 for (int i = 1; i < n; i++)

70 {

71 scanf("%d %d", &x, &y);

72 vis[y] = true;

73 G[x].push\_back(y);

74 }

75

76 for (int i = 1; i <= n; i++){

77 if (!vis[i]){

78 root = i;

79 break;

80 }

81 }

82 dfs(0, root);

83 rmq(2 \* n - 1);

84 scanf("%d %d", &x, &y);

85 printf("%d\n", get(pos[x], pos[y]));

86 }

87 return 0;

88 }

* 1. Hungarian-dfs

1 bool dfs(int u){

2 for(int i = 1; i <= n; i++){

3 if(a[u][i] && !visit[i]){

4 visit[i] = true;

5 if(match[i] == -1 || dfs(match[i])){

6 match[i] = u;

7 }

8 return true;

9 }r

10 }

11 return false;

12 }

* 1. Tarjan

1 int pre[MAXN], isbridge[MAXM], low[MAXN];

2 vector<Edge> G[MAXN];

3 int dfs\_clock;

4 int dfs(int u, int father){

5 int lowu = pre[u] = ++dfs\_clock;

6 //int child = 0;

7 for (int i = 0; i < G[u].size(); i++){

8 int v = G[u][i].to;

9 if (!pre[v]){

10 //child++;

11 int lowv = dfs(v, G[u][i].pos);

12 lowu = min(lowu, lowv);

13 if (lowv > pre[u]){

14 isbridge[G[u][i].pos] = true;

15 }

16 }

17 else if (pre[v] < pre[u] && G[u][i].pos != father){

18 lowu = min(lowu, pre[v]);

19 }

20 }

21 low[u] = lowu;

22 return lowu;

1. DATA STRUCTURE
   1. Kd-tree

1 #include <iostream>

2 #include <cstdio>

3 #define LL long long

4 #define eps 1e-8

5 #define INF 0x3f3f3f3f

6 #define MAXN 100005

7 using namespace std;

8 int sum[MAXN \* 3], add[MAXN \* 3];

9

10 void pushup(int t){

11 sum[t] = sum[t << 1] + sum[t << 1 | 1];

12 }

13 void pushdown(int t, int x){

14 if (add[t]){

15 add[t << 1] += add[t];

16 add[t << 1 | 1] += add[t];

17 sum[t << 1] += ((x + 1) >> 1)\* add[t];

18 sum[t << 1 | 1] += (x >> 1) \* add[t];

19 add[t] = 0;

20 }

21 }

22 void update(int L, int R, int t, int p, int q, int x){

23 if (p <= L && q >= R){

24 sum[t] += (R - L + 1) \* x;

25 add[t] += x;

26 return;

27 }

28

29 pushdown(t, R - L + 1);

30 int mid = (L + R) >> 1;

31 if (p <= mid){

32 update(L, mid, t << 1, p, q, x);

33 }

34 if (q > mid){

35 update(mid + 1, R, t << 1 | 1, p, q, x);

36 }

37 pushup(t);

38 }

39 int query(int L, int R, int t, int p, int q){

40 if (p <= L && q >= R){

41 return sum[t];

42 }

43 pushdown(t, R - L + 1);

44 int mid = (L + R) >> 1;

45 int res = 0;

46 if (p <= mid){

47 res += query(L, mid, t << 1, p, q);

48 }

49 if (q > mid){

50 res += query(mid + 1, R, t << 1 | 1, p, q);

51 }

52 return res;

53 }

54 int main()

55 {

56 int n;

57 while (~scanf("%d", &n) && n){

58 memset(sum, 0, sizeof(sum));

59 memset(add, 0, sizeof(add));

60 int x, y;

61 for (int i = 1; i <= n; i++){

62 scanf("%d%d", &x, &y);

63 update(1, n, 1, x, y, 1);

64 }

65 for (int i = 1; i < n; i++){

66 printf("%d ", query(1, n, 1, i, i));

67 }

68 printf("%d\n", query(1, n, 1, n, n));

69 }

70 }

* 1. Tree array

1 int lowbit(int x)

2 {

3 return x & (-x);

4 }

5 void modify(int x,int add)//一维

6 {

7 while(x<=MAXN)

8 {

9 a[x]+=add;

10 x+=lowbit(x);

11 }

12 }

13 int get\_sum(int x)

14 {

15 int ret=0;

16 while(x!=0)

17 {

18 ret+=a[x];

19 x-=lowbit(x);

20 }

21 return ret;

22 }

23 void modify(int x,int y,int data)//二维

24 {

25 for(int i=x;i<MAXN;i+=lowbit(i))

26 for(int j=y;j<MAXN;j+=lowbit(j))

27 a[i][j]+=data;

28 }

29 int get\_sum(int x,int y)

30 {

31 int res=0;

32 for(int i=x;i>0;i-=lowbit(i))

33 for(int j=y;j>0;j-=lowbit(j))

34 res+=a[i][j];

35 return res;

36 }

* 1. Sa array

1 #include<iostream>

2 #include<stdio.h>

3 #include<string.h>

4 using namespace std;

5 #define min(x,y) x>y? y:x

6 #define N 200010

7 int dp[N][33];

8 int wa[N], wb[N], wsf[N], wv[N], sa[N];

9 int ra[N], height[N], s[N];

10 char str[N], str1[N];

11 int cmp(int \*r, int a, int b, int k)

12 {

13 return r[a] == r[b] && r[a + k] == r[b + k];

14 }

15 void getsa(int \*r, int \*sa, int n, int m)

16 {

17 int i, j, p, \*x = wa, \*y = wb, \*t;

18 for (i = 0; i < m; i++) wsf[i] = 0;

19 for (i = 0; i < n; i++) wsf[x[i] = r[i]]++;

20 for (i = 1; i < m; i++) wsf[i] += wsf[i - 1];

21 for (i = n - 1; i >= 0; i--) sa[--wsf[x[i]]] = i;

22 p = 1;

23 j = 1;

24 for (; p < n; j \*= 2, m = p)

25 {

26 for (p = 0, i = n - j; i < n; i++) y[p++] = i;

27 for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;

28 for (i = 0; i < n; i++) wv[i] = x[y[i]];

29 for (i = 0; i < m; i++) wsf[i] = 0;

30 for (i = 0; i < n; i++) wsf[wv[i]]++;

31 for (i = 1; i < m; i++) wsf[i] += wsf[i - 1];

32 for (i = n - 1; i >= 0; i--) sa[--wsf[wv[i]]] = y[i];

33 t = x;

34 x = y;

35 y = t;

36 x[sa[0]] = 0;

37 for (p = 1, i = 1; i < n; i++)

38 x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;

39 }

40 }

41 void getheight(int \*r, int n)

42 {

43 int i, j, k = 0;

44 for (i = 1; i <= n; i++) ra[sa[i]] = i;

45 for (i = 0; i < n; i++)

46 {

47 if (k)

48 k--;

49 else

50 k = 0;

51 j = sa[ra[i] - 1];

52 while (r[i + k] == r[j + k])

53 k++;

54 height[ra[i]] = k;

55 }

56 }

57 int main()

58 {

59 while (cin >> str)

60 {

61 cin >> str1;

62 int n = 0, len = strlen(str);

63 for (int i = 0; i < len; i++)

64 s[n++] = str[i] - 'a' + 1;

65 s[n++] = 28;

66 len = strlen(str1);

67 for (int i = 0; i < len; i++)

68 s[n++] = str1[i] - 'a' + 1;

69 s[n] = 0;

70 getsa(s, sa, n + 1, 30);

71 getheight(s, n);

72 int max = 0, pos = 0;

73 len = strlen(str);

74 for (int i = 2; i<n; i++)

75 if (height[i]>max)

76 {

77 if (0 <= sa[i - 1] && sa[i - 1] < len&&len < sa[i])

78 max = height[i];

79 if (0 <= sa[i] && sa[i] < len&&len < sa[i - 1])

80 max = height[i];

81 }

82 cout << max << endl;

83 }

84 return 0;

85 }

1. GEOMETRY

1 #define eps 1e-8

2 int dcmp(double x){

3 if (fabs(x) < eps) return 0;

4 return x < 0 ? -1 : 1;

5 }

6 struct Point{

7 double x, y;

8 Point(double p = 0, double q = 0){

9 x = p;

10 y = q;

11 }

12 };

13 struct Node{

14 int p;

15 Point a, b;

16 Node(Point a1, Point a2, int t){

17 a = a1;

18 b = a2;

19 p = t;

20 }

21 };

22

23 typedef Point Vector;

24

25 Vector operator + (Vector A, Vector B){

26 return Vector(A.x + B.x, A.y + B.y);

27 }

28 Vector operator - (Vector A, Vector B){

29 return Vector(A.x - B.x, A.y - B.y);

30 }

31 Vector operator \* (Vector A, double p){

32 return Vector(A.x \* p, A.y \* p);

33 }

34 Vector operator / (Vector A, double p){

35 return Vector(A.x / p, A.y / p);

36 }

37 bool operator == (Vector A, Vector B){

38 return dcmp(A.x - B.x) == 0 && dcmp(A.y - B.y) == 0;

39 }

40 bool operator > (Vector A, Vector B){

41 return A.x > B.x && A.y > B.y;

42 }

43 bool operator <(Vector A, Vector B){

44 return A.x < B.x && A.y < B.y;

45 }

46 //点积

47 double Dot(Vector A, Vector B){

48 return A.x \* B.x + A.y \* B.y;

49 }

50 //模

51 double Length(Vector A){

52 return sqrt(Dot(A, A));

53 }

54 //夹角

55 double Angle(Vector A, Vector B){

56 return acos(Dot(A, B) / Length(A) / Length(B));

57 }

58 //叉积

59 double Cross(Vector A, Vector B){

60 return A.x \* B.y - A.y\*B.x;

61 }

62 //三角形面积

63 double Area2(Point A, Point B, Point C){

64 return Cross(B - A, C - A);

65 }

66 //点在直线上投影

67 Point GetLineProjection(Point P, Point A, Point B){

68 Vector v = B - A;

69 return A + v \* (Dot(v, P - A) / Dot(v, v));

70 }

71 //线段相交(不含端点)

72 bool SegmentProperIntersection(Point a1, Point a2, Point b1, Point b2){

73 double c1 = Cross(a2 - a1, b1 - a1);

74 double c2 = Cross(a2 - a1, b2 - a1);

75 double c3 = Cross(b2 - b1, a1 - b1);

76 double c4 = Cross(b2 - b1, a2 - b1);

77 return dcmp(c1) \* dcmp(c2) < 0 && dcmp(c3) \* dcmp(c4) < 0;

78 }

79 //点在直线上(不含端点)

80 bool OnSegment(Point p, Point a1, Point a2){

81 return dcmp(Cross(a1 - p, a2 - p)) == 0 && dcmp(Dot(a1 - p, a2 - p)) < 0;

82 }

1. ALGORITHM
   1. RMQ

1 void rmq(int n)

2 {

3 for (int i = 1; i <= n; i++)

4 f[0][i] = deep[i];

5 for (int j = 1; j <= (int)(log((double)n) / log(2.0)); j++){

6 for (int i = 1; i <= n - (1 << j) + 1; i++){

7 f[j][i] = min(f[j - 1][i], f[j - 1][i + (1 << (j - 1))]);

8 }

9 }

10 }

* 1. Manacher

1 void manacher(){

2 int res = 0, id = 0;

3 for(int i = 1; i <= m; i++) {

4 if(res > i){

5 p[i] = min(p[2 \* id - i], res - i);

6 }

7 else{

8 p[i] = 1;

9 }

10 //p[i] = mx > i? min(mp[2\*id-i], mx-i): 1;

11 while(s[i + p[i]] == s[i - p[i]]){

12 p[i]++;

13 }

14 //while(s[i+mp[i]] == s[i-mp[i]]) mp[i]++;

15 if(i + p[i] > res) {

16 res = i + p[i];

17 id = i;

18 }

19 }

20 }

* 1. Ac automatic

1 #include <iostream>

2 #include <cstdio>

3 #include <algorithm>

4 #include <cstring>

5 #define LL long long

6 #define INF 0x3f3f3f3f

7

8 #define MAXM 1000005

9 using namespace std;

10

11 int cnt[200];

12 char s[MAXM];

13 char words[160][100];

14 int n, ans;

15

16 struct Node

17 {

18 int count, id;

19 struct Node \*next[26];

20 struct Node \*fail;

21 void init(){

22 int i;

23 for (int i = 0; i < 26; i++){

24 next[i] = NULL;

25 }

26 count = -1;

27 fail = NULL;

28 id = -1;

29 }

30 };

31 Node \*root, \*d[MAXM];

32

33

34 void insert(char \*s, int id){

35 int len, k;

36 Node \*p = root;

37 len = strlen(s);

38 for (k = 0; k < len; k++){

39 int pos = s[k] - 'a';

40 if (p->next[pos] == NULL){

41 p->next[pos] = new Node;

42 p->next[pos]->init();

43 p = p->next[pos];

44 }

45 else

46 p = p->next[pos];

47 }

48 p->count = id;

49 }

50

51 void build(Node \*root){

52 int head, tail, i;

53 Node \*p, \*temp;

54 head = 0;

55 tail = 0;

56 root->fail = NULL;

57 d[head] = root;

58 while (head <= tail){

59 temp = d[head++];

60 for (int i = 0; i < 26; i++){

61 if (temp->next[i] == NULL) continue;

62 if (temp == root){

63 temp->next[i]->fail = root;

64 }

65 else{

66 p = temp->fail;

67 while (p != NULL){

68 if (p->next[i] != NULL){

69 temp->next[i]->fail = p->next[i];

70 break;

71 }

72 p = p->fail;

73 }

74 if (p == NULL){

75 temp->next[i]->fail = root;

76 }

77 }

78 d[++tail] = temp->next[i];

79 }

80 }

81 }

82

83 void query(){

84 int len = strlen(s);

85 Node \*p, \*temp;

86 p = root;

87 for (int i = 0; i < len; i++){

88 int pos = s[i] - 'a';

89 while (!p->next[pos] && p != root) p = p->fail;

90 p = p->next[pos];

91 if (!p) p = root;

92 temp = p;

93 while (temp != root){

94 if (temp->count >= 0){

95 cnt[temp->count]++;

96 }

97 temp = temp->fail;

98 }

99 }

100 }

101

102

103 int main()

104 {

105 while (~scanf("%d", &n)){

106 if (n == 0) break;

107 memset(cnt, 0, sizeof(cnt));

108 root = new Node;

109 root->init();

110 for (int i = 0; i < n; i++){

111 scanf("%s", &words[i]);

112 insert(words[i], i);

113 }

114 build(root);

115 scanf("%s", s);

116 query();

117 ans = -1;

118 for (int i = 0; i < n; i++){

119 if (cnt[i] >ans){

120 ans = cnt[i];

121 }

122 }

123 printf("%d\n", ans);

124 for (int i = 0; i < n; i++){

125 if (cnt[i] == ans){

126 printf("%s\n", words[i]);

127 }

128 }

129 }

130 }

* 1. Kmp

1 #include <iostream>

2 #include <cstdio>

3 #define MAXN 1000005

4 using namespace std;

5 int n, last[MAXN], j, m = 0;

6 char s[MAXN];

7 int main()

8 {

9 while (~scanf("%d", &n)){

10 if (n == 0) break;

11 memset(last, 0, sizeof(last));

12 scanf("%s", s + 1);

13

14 int k = 0;

15 last[1] = 0;

16 for (int i = 2; i <= n; i++){

17 while (s[k + 1] != s[i] && k > 0){

18 k = last[k];

19 }

20 if (s[k + 1] == s[i]){

21 k++;

22 }

23 last[i] = k;

24 }

25 printf("Test case #%d\n", ++m);

26 for (int i = 2; i <= n; i++){

27 j = i - last[i];

28 if (i % j == 0 && i > j){

29 printf("%d %d\n", i, i / j);

30 }

31 }

32 printf("\n");

33 }

34 }

1. DP
   1. LIS