

Contents

1	DP	1
1.1	LCS	1
1.2	LIS $O(n^2)$	1
1.3	LIS $O(n \log n)$	1
1.4	LIS $O(n \log n)$	1
2	Prime	1
2.1	質數篩 CPP	1
2.2	質數篩 PY	2
2.3	單一質數	2
2.4	egcd CPP	2
2.5	egcd PY	2
3	Graph	2
3.1	Floyd Warshall	2
3.2	Bellman Ford	2
3.3	SPFA	2
3.4	Dijkstra	3
3.5	Kruskal's Algorithm	3
3.6	Prim's Algorithm	3
3.7	LCA	3
3.8	Topological Sort	3
4	RMQ	4
4.1	Segment Tree	4
4.2	BIT	5
4.3	Sparse Table	5
5	Uncategorized	6
5.1	快速幂	6
5.2	矩陣快速幂	6
5.3	快速計算費氏數列	6

1 DP

1.1 LCS

```

1 #include <bits/stdc++.h>
2 #define IOS
3 ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 string s1, s2;
6 int dp[505][505];
7 int main(){
8     IOS
9     cin >> s1 >> s2;
10    memset(dp, 0, sizeof(dp));
11    int l1 = s1.size(), l2 = s2.size();
12    for(int i = 1; i <= l1; i++){
13        for(int j = 1; j <= l2; j++){
14            if(s1[i-1] == s2[j-1]) dp[i][j] =
15                dp[i-1][j-1] + 1;
16            else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
17        }
18    }
19    cout << dp[l1][l2] << '\n';
20    return 0;
21 }
```

1.2 LIS $O(n^2)$

```

1 #include <bits/stdc++.h>
2 #define IOS
3 ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 typedef long long ll;
6 int main(){
7     IOS
8     int arr[100];
9     int n;
10    cin >> n;
11    for(int i = 0; i < n; i++) cin >> arr[i];
12    int dp[100];
```

```

13    for(int i = 0; i < n; i++) dp[i] = 1;
14    for(int i = 0; i < n; i++){
15        for(int j = 0; j < i; j++){
16            if(arr[i] > arr[j])
17                dp[i] = max(dp[j] + 1, dp[i]);
18        }
19    }
20    int ans = 1;
21    for(int i = 0; i < n; i++) ans = max(ans, dp[i]);
22    cout << ans << '\n';
23    return 0;
24 }
```

1.3 LIS $O(n \log n)$

```

1 class Solution {
2 public:
3     int lengthOfLIS(vector<int>& nums) {
4         vector<int> v;
5         int n = nums.size();
6         for(int i = 0; i < n; i++){
7             int p = lower_bound(v.begin(), v.end(),
8                 nums[i]) - v.begin();
9             if(p == v.size()) v.push_back(nums[i]);
10            else v[p] = nums[i];
11        }
12        return v.size();
13    };
14 }
```

1.4 LIS $O(n \log n)$

```

1 for(int i=0;i<num.size();i++){
2     if(lis.empty()||lis.back()<num[i]){
3         lis.push_back(num[i]);
4         dp[i]=lis.size();
5     }
6     else{
7         auto iter=lower_bound(all(lis),num[i]);
8         dp[i]=iter-lis.begin()+1;
9         *iter=num[i];
10    }
11 }
12 int length=lis.size();
13 for(int i=num.size()-1;i>=0;i--){
14     if(dp[i]==length){
15         ans.push_back(num[i]);
16         length--;
17     }
18 }
```

2 Prime

2.1 質數篩 CPP

```

1 bitset<MAXN> prime_bool;
2 vector<ll> prime;
3 void find_prime(){
4     prime_bool.set();
5     for(int i=2;i<MAXN;i++){
6         if(prime_bool[i]){
7             prime.push_back(i);
8         }
9         for(auto j:prime){
10            if(j*i>=MAXN)
11                break;
12            prime_bool[j*i]=0;
13            if(i%j==0)
14                break;
```

```

15     }
16 }
17 }

```

2.2 質數篩 PY

```

1 is_prime = n * [1]
2 is_prime[0] = is_prime[1] = 0
3
4 for i in range(2, n):
5     if is_prime[i]:
6         for j in range(2, n):
7             if i * j >= n:
8                 break
9             is_prime[i * j] = 0

```

2.3 單一質數

```

1 bool prime(int n){
2     if(n < 2) return false;
3     if(n <= 3) return true;
4     if(!(n % 2) || !(n % 3)) return false;
5     for(int i = 5; i * i <= n; i += 6)
6         if(!(n % i) || !(n % (i + 2))) return false;
7     return true;
8 }

```

2.4 egcd CPP

```

1 int exgcd(int a, int b, int &x, int &y){
2     if(b==0){
3         x=1, y=0;
4         return a;
5     }
6     int gcd=exgcd(b, a%b, y, x);
7     y-=a/b*x;
8     return gcd;
9 }

```

2.5 egcd PY

```

1 def egcd(a: int, b: int) -> Tuple[int, int, int]:
2     """return (g, x, y) such that a*x + b*y = g =
3         gcd(a, b)"""
4     """x % y"""
5     if a == 0:
6         return (b, 0, 1)
7     else:
8         b_div_a, b_mod_a = divmod(b, a)
9         g, x, y = egcd(b_mod_a, a)
10        return (g, y - b_div_a * x, x)

```

3 Graph

3.1 Floyd Warshall

```

1 int n, rd, l, r, v;
2 cin >> n >> rd;
3 vector<vector<int>> dp(n + 1, vector<int>(n + 1,
4     1e9));
5 for(int i = 0; i < rd; i++){
6     cin >> l >> r >> v;
7     dp[l][r] = dp[r][l] = v;
8     //每條路皆雙向
9 }

```

```

10 //以下即 Floyd-Warshall
11 for(int k = 1; k <= n; k++){
12     for(int i = 1; i <= n; i++){
13         for(int j = 1; j <= n; j++){
14             dp[i][j] = min(dp[i][k] + dp[k][j],
15                 dp[i][j]);
16             //窮舉所有鬆弛可能
17         }
18     }
19 cin >> l >> r;
20 cout << dp[l][r];

```

3.2 Bellman Ford

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 struct Edge{
8     int x, y, t;
9 };
10 int dis[1005];
11 int main(){
12     IOS
13     int c;
14     cin >> c;
15     while(c--){
16         vector<Edge> edge;
17         int n, m;
18         cin >> n >> m;
19         for(int i = 0; i <= n; i++) dis[i] = INF;
20         dis[0] = 0;
21         for(int i = 0; i < m; i++){
22             int x, y, t;
23             cin >> x >> y >> t;
24             edge.push_back({x, y, t});
25         }
26         for(int i = 0; i < n - 1; i++){
27             for(int j = 0; j < m; j++){
28                 if(dis[edge[j].x] + edge[j].t <
29                     dis[edge[j].y]){
30                     dis[edge[j].y] = dis[edge[j].x] + edge[j].t;
31                 }
32             }
33         }
34         bool judge = true;
35         for(auto e : edge){
36             if(dis[e.x] + e.t < dis[e.y]){
37                 judge = false;
38                 break;
39             }
40         }
41         cout << (judge ? "not possible" : "possible") <<
42             '\n';
43     }
44     return 0;
45 }

```

3.3 SPFA

```

1 #define mem(x) memset(x, 0, sizeof(x))
2 struct road{
3     int r, val;
4 };
5 int main(){
6     int n, e, l, r, v;
7     cin >> n >> e;
8     vector<int> dp(n + 1, 1e9);
9     vector<road> rd[n + 1];

```

```

10 for(int i = 0; i < e; i++){
11     cin >> l >> r >> v;
12     rd[l].push_back({r, v});
13     rd[r].push_back({l, v});
14 }
15 cin >> l >> r;
16 dp[l] = 0;
17 queue<int> que;
18 que.push(l);
19 bool check[n + 1]; mem(check);
20 int cnt[n + 1]; mem(cnt);
21 while(!que.empty()){
22     int tmp = que.front(); que.pop();
23     check[tmp] = 0, cnt[tmp]++;
24     if(cnt[tmp] >= n) {cout << "neg cycle\n"; break;}
25     for(auto & i : rd[tmp]){
26         if(dp[i.r] > dp[tmp] + i.val){
27             dp[i.r] = dp[tmp] + i.val;
28             if(!check[i.r]) check[i.r] = 1, que.push(i.r);
29         }
30     }
31 }
32 for(auto & i : dp) cout << i << ' ';
33 return 0;
34 }

```

3.4 Dijkstra

```

1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <queue>
5 #define IOS
6     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
7 #define INF 2147483647
8 using namespace std;
9 int n, m;
10 vector<pair<int, int>> adj[100005];
11 bool visited[100005] = {false};
12 priority_queue<pair<int, int>> pq;
13 int dis[100005], parent[100005];
14 void solve(){ // Dijkstra
15     dis[0] = 0;
16     for(int i = 1; i < n; i++) dis[i] = INF;
17     pq.push(make_pair(0, 0));
18     while(!pq.empty()){
19         auto node = pq.top();
20         pq.pop();
21         int v = node.second; // parent
22         if(visited[v]) continue;
23         visited[v] = true;
24         for(auto i : adj[v]){
25             int vertex = i.first, weight = i.second;
26             if(visited[vertex]) continue;
27             if(dis[v] + weight < dis[vertex]){
28                 dis[vertex] = dis[v] + weight;
29                 parent[vertex] = v;
30                 pq.push(make_pair(-dis[vertex], vertex));
31             }
32         }
33     }
34     int maxd = -1, cnt = 0;
35     for(int i = 0; i < n; i++){
36         if(dis[i] < INF){
37             if(dis[i] > maxd) maxd = dis[i];
38             else cnt++;
39         }
40     }
41     cout << maxd << '\n' << cnt << '\n';
42 }
43 int main(){
44     IOS
45     cin >> n >> m;
46     for(int i = 0; i < m; i++){
47         int u, v, w;

```

```

47         cin >> u >> v >> w;
48         adj[u].push_back(make_pair(v, w));
49         adj[v].push_back(make_pair(u, w));
50     }
51     solve();
52
53     return 0;
54 }

```

3.5 Kruskal's Algorithm

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 int parent[10005];
6 struct Edge{
7     int u, v, w;
8     bool operator < (Edge &b){
9         return w < b.w;
10    }
11 };
12 int query(int a){
13     if(parent[a] == -1) return a;
14     return parent[a] = query(parent[a]);
15 }
16 bool merge(int a, int b){
17     int r1 = query(a);
18     int r2 = query(b);
19     if(r1 == r2) return false;
20     if(parent[r1] < parent[r2]) parent[r2] = r1;
21     else parent[r1] = r2;
22     return true;
23 }
24 int main(){
25     IOS
26     int n, m;
27     memset(parent, -1, sizeof(parent));
28     cin >> n >> m;
29     vector<Edge> adj;
30     for(int i = 0; i < m; i++){
31         int u, v, w;
32         cin >> u >> v >> w;
33         adj.push_back({u, v, w});
34     }
35     sort(adj.begin(), adj.end());
36     // for(int i = 0; i < m; i++) cout << adj[i].w << '
37     '
38     int cost = 0, n_edge = 0;
39     for(Edge e : adj){
40         if(merge(e.u, e.v)){
41             cost += e.w;
42             n_edge++;
43         }
44     }
45     if(n_edge == n - 1) cout << cost << '\n';
46     else cout << -1 << '\n';
47     return 0;
48 }

```

3.6 Prim's Algorithm

```

1 #include <iostream>
2 #include <queue>
3 #include <algorithm>
4 #include <cstring>
5 #define IOS
6     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
7 using namespace std;
8 int n, m, dis[10005], parent[10005];
9 bool visited[10005] = {false};
10 vector<pair<int, int>> adj[100005];

```

```

10 int main(){
11     IOS
12     // freopen("input.in", "r", stdin);
13     cin >> n >> m;
14     memset(dis, 0x3f3f3f3f, sizeof(dis));
15     memset(parent, -1, sizeof(parent));
16     for(int i = 0; i < m; i++){
17         int u, v, w;
18         cin >> u >> v >> w;
19         adj[u].push_back({v, w});
20         adj[v].push_back({u, w});
21     }
22     int start = 0;
23     dis[start] = 0;
24     priority_queue<pair<int, int>, vector<pair<int,
25         int>>, greater<pair<int, int>>> > pq;
26     pq.push({dis[start], start});
27     while(!pq.empty()){
28         pair<int, int> cur = pq.top();
29         pq.pop();
30         if(visited[cur.second]) continue;
31         visited[cur.second] = true;
32         for(auto i : adj[cur.second]){
33             if(visited[i.first]) continue;
34             if(dis[i.first] > i.second){
35                 dis[i.first] = i.second;
36                 parent[i.first] = cur.second;
37                 pq.push({dis[i.first], i.first});
38             }
39         }
40         int cost = 0, err = 0;
41         for(int i = 0; i < n; i++){
42             if(dis[i] < 0x3f3f3f3f) cost += dis[i];
43             else err++;
44         }
45         cout << (err ? -1 : cost) << "\n";
46         // for(int i = 0; i < n; i++) cout << dis[i] << ' ';
47
48         return 0;
49     }

```

3.7 LCA

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 const int N = 2e5 + 5;
8 int n, q;
9 vector<int> vec[N];
10 int p[20][N], in[N], out[N];
11 bool valid(int a, int b){
12     return (in[a] <= in[b] && out[b] <= out[a]);
13 }
14 void dfs(int cur, int par){
15     static int t = 0;
16     p[0][cur] = par;
17     in[cur] = t++;
18     for(auto e : vec[cur]){
19         dfs(e, cur);
20     }
21     out[cur] = t++;
22 }
23 int lca(int a, int b){
24     if(valid(a, b)) return a;
25     for(int i = 19; i >= 0; i--){
26         if(!valid(p[i][a], b)) a = p[i][a];
27     }
28     return p[0][a];
29 }
30 int main(){
31     IOS
32     cin >> n >> q;

```

```

32     for(int i = 2; i <= n; i++){
33         int e;
34         cin >> e;
35         vec[e].push_back(i);
36     }
37     dfs(1, 1);
38     for(int i = 1; i < 20; i++){
39         for(int j = 1; j <= n; j++){
40             p[i][j] = p[i - 1][p[i - 1][j]];
41         }
42     }
43     while(q--){
44         int u, v;
45         cin >> u >> v;
46         cout << lca(u, v) << "\n";
47     }
48     return 0;
49 }
50 }

```

3.8 Topological Sort

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 using namespace std;
5 typedef long long ll;
6 vector<int> vec[200005];
7 int ind[100005];
8 int main(){
9     IOS
10     int n, m;
11     cin >> n >> m;
12     memset(ind, 0, sizeof(ind));
13     for(int i = 0; i < m; i++){
14         int a, b;
15         cin >> a >> b;
16         ind[b]++;
17         vec[a].push_back(b);
18     }
19     queue<int> q;
20     for(int i = 1; i <= n; i++){
21         if(ind[i] == 0) q.push(i);
22     }
23     vector<int> top;
24     while(!q.empty()){
25         int cur = q.front();
26         q.pop();
27         top.push_back(cur);
28         for(auto e : vec[cur]){
29             ind[e]--;
30             if(ind[e] == 0){
31                 q.push(e);
32             }
33         }
34     }
35     if(top.size() == n){
36         for(auto i : top) cout << i << ' ';
37         cout << "\n";
38     }
39     else cout << "IMPOSSIBLE" << "\n";
40     return 0;
41 }

```

4 RMQ

4.1 Segment Tree

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);

```

```

3 #define L(x) (x << 1)
4 #define R(x) ((x << 1) | 1)
5 using namespace std;
6 typedef long long ll;
7 ll seg[500005 << 2], lazy[500005 << 2];
8 int n, q;
9 void init(){
10     memset(seg, 0, sizeof(seg));
11     memset(lazy, 0, sizeof(lazy));
12 }
13 void build(int x, int l, int r){
14     if(l == r){
15         cin >> seg[x];
16         return;
17     }
18     int mid = (l + r) >> 1;
19     build(L(x), l, mid);
20     build(R(x), mid + 1, r);
21     seg[x] = seg[L(x)] + seg[R(x)];
22 }
23 void push(int pos, int size){
24     lazy[L(pos)] += lazy[pos];
25     lazy[R(pos)] += lazy[pos];
26     seg[pos] = seg[pos] + lazy[pos] * size;
27     lazy[pos] = 0;
28 }
29 void modify(int x, int l, int r, int ql, int qr, int
    val){
30     if(lazy[x]) push(x, (r - l) + 1);
31     // seg[x] = seg[L(x)] + (mid - l) * lazy[L(x)] +
32     // seg[R(x)] + (r - mid) * lazy[R(x)];
33     seg[x] += val * (qr - ql + 1);
34     if(ql <= l && qr >= r){
35         lazy[x] += val;
36         return;
37     }
38     int mid = (l + r) >> 1;
39     if(qr <= mid) modify(L(x), l, mid, ql, qr, val);
40     else if(ql > mid) modify(R(x), mid + 1, r, ql,
41         qr, val);
42     else{
43         modify(L(x), l, mid, ql, mid, val);
44         modify(R(x), mid + 1, r, mid + 1, qr, val);
45     }
46 }
47 ll query(int x, int l, int r, int ql, int qr){
48     if(ql <= l && qr >= r) return seg[x] + lazy[x] *
49     (r - l);
50     if(lazy[x]) push(x, (r - l) + 1);
51     int mid = (l + r) >> 1;
52     if(qr <= mid) return query(L(x), l, mid, ql, qr);
53     else if(ql > mid) return query(R(x), mid + 1, r,
54         ql, qr);
55     else return query(L(x), l, mid, ql, mid) +
56     query(R(x), mid + 1, r, mid + 1, qr);
57 }
58 int main(){
59     IOS
60     init();
61     cin >> n;
62     build(1, 1, n);
63     cin >> q;
64     while(q--){
65         int v, x, y, k;
66         cin >> v;
67         if(v == 1){
68             cin >> x >> y >> k;
69             modify(1, 1, n, x, y, k);
70         }
71         else{
72             cin >> x >> y;
73             ll ans = query(1, 1, n, x, y);
74             cout << ans << '\n';
75         }
76     }
77     return 0;

```

74 | }

4.2 BIT

```

1 // BIT
2 #include <bits/stdc++.h>
3 #include <ext/pb_ds/assoc_container.hpp>
4 #include <ext/pb_ds/tree_policy.hpp>
5 // #include <ext/pb_ds/detail/standard_policies.hpp>
6 #define IOS
7     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
8 #define INF 0x3f3f3f3f
9 #define lowbit(x) x & (-x)
10 using namespace std;
11 using namespace __gnu_pbds;
12 typedef long long ll;
13 const int N = 2e5 + 5;
14 ll bit[N], n, q;
15 ll query(int idx){
16     ll sum = 0;
17     for(int i = idx; i > 0; i -= lowbit(i))
18         sum += bit[i];
19     return sum;
20 }
21 void update(ll val, int idx){
22     for(int i = idx; i <= n; i += lowbit(i))
23         bit[i] += val;
24 }
25 int main(){
26     IOS
27     cin >> n >> q;
28     for(int i = 1; i <= n; i++){ // 1-based
29         ll in;
30         cin >> in;
31         update(in, i);
32     }
33     while(q--){
34         ll o, a, b;
35         cin >> o >> a >> b;
36         if(o == 1){
37             ll u = query(a) - query(a - 1);
38             update(b - u, a);
39         }
40         else{
41             cout << query(b) - query(a - 1) << '\n';
42         }
43     }
44     return 0;
45 }

```

4.3 Sparse Table

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 const int N = 5e5 + 5;
8 int n, m, arr[N], dp[35][N];
9 void sparse_table(int n){
10     for(int i = 1; i <= n; i++){
11         for(int j = 0; (j + (1LL << (i - 1))) <= n; j++){
12             dp[i][j] = max(dp[i - 1][j], dp[i - 1][j
13                 + (1LL << (i - 1))]);
14         }
15     }
16 }
17 int query(int l, int r){
18     int idx = __lg(r - l + 1);
19     return max(dp[idx][l], dp[idx][r - (1LL << idx) +
20         1]);

```

```

18 }
19 int main(){
20     IOS
21     cin >> n;
22     for(int i = 0; i < n; i++) cin >> arr[i];
23     cin >> m;
24     for(int i = 0; i < n; i++) dp[0][i] = arr[i];
25     sparse_table(n);
26     while(m--){
27         int l, r;
28         cin >> l >> r;
29         if(l > r) swap(l, r);
30         l--, r--;
31         cout << query(l, r) << '\n';
32     }
33
34     return 0;
35 }

```

5 Uncategorized

5.1 快速幂

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 using namespace std;
5 typedef long long ll;
6 ll mod = 1000000007;
7 ll fast_pow(int base, int exp){
8     ll res = 1;
9     while(exp > 0){
10         if(exp & 1) res = res * base % mod;
11         base = base * base % mod;
12         exp >>= 1;
13     }
14     return res;
15 }
16 int main(){
17     IOS
18     int base = 3, exp = 15;
19     cout << fast_pow(base, exp) << '\n';
20
21     return 0;
22 }

```

5.2 矩陣快速幂

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 ll mod = 1000000007;
8 struct Matrix{
9     ll mat[2][2] = {{0}};
10     Matrix operator * (Matrix &inp){
11         Matrix tmp;
12         for(int i = 0; i < 2; i++){
13             for(int j = 0; j < 2; j++){
14                 for(int k = 0; k < 2; k++){
15                     tmp.mat[i][j] = ((tmp.mat[i][j] +
16                                     (mat[i][k] % mod) *
17                                     (inp.mat[k][j] % mod)) % mod)
18                                     % mod;
19                 }
20             }
21         }
22         return tmp;
23     }
24 }
25
26 Matrix base;
27 Matrix fast_pow(int exp){
28     if(exp == 1) return base;
29     if(exp % 2 == 0){
30         Matrix res = fast_pow(exp >> 1);
31         return res * res;
32     }
33     Matrix res = fast_pow(exp >> 1);
34     return base * res * res;
35 }
36
37 int main(){
38     IOS
39     base.mat[0][0] = 1;
40     base.mat[0][1] = 4;
41     base.mat[1][0] = 2;
42     base.mat[1][1] = 3;
43     Matrix output = fast_pow(10);
44     for(int i = 0; i < 2; i++){
45         for(int j = 0; j < 2; j++){
46             cout << output.mat[i][j] << ' ';
47         }
48         cout << '\n';
49     }
50
51     return 0;
52 }

```

```

21 Matrix base;
22 Matrix fast_pow(int exp){
23     if(exp == 1) return base;
24     if(exp % 2 == 0){
25         Matrix res = fast_pow(exp >> 1);
26         return res * res;
27     }
28     Matrix res = fast_pow(exp >> 1);
29     return base * res * res;
30 }
31 int main(){
32     IOS
33     base.mat[0][0] = 1;
34     base.mat[0][1] = 4;
35     base.mat[1][0] = 2;
36     base.mat[1][1] = 3;
37     Matrix output = fast_pow(10);
38     for(int i = 0; i < 2; i++){
39         for(int j = 0; j < 2; j++){
40             cout << output.mat[i][j] << ' ';
41         }
42         cout << '\n';
43     }
44
45     return 0;
46 }

```

5.3 快速計算費氏數列

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 ll mod = 1000000007;
8 struct Matrix{
9     ll mat[2][2] = {{0}};
10     Matrix operator * (Matrix &inp){
11         Matrix tmp;
12         for(int i = 0; i < 2; i++){
13             for(int j = 0; j < 2; j++){
14                 for(int k = 0; k < 2; k++){
15                     tmp.mat[i][j] = ((tmp.mat[i][j] +
16                                     (mat[i][k] % mod) *
17                                     (inp.mat[k][j] % mod)) % mod)
18                                     % mod;
19                 }
20             }
21         }
22         return tmp;
23     }
24 }
25
26 Matrix base;
27 Matrix fast_pow(int n){
28     if(n == 1) return base;
29     if(n % 2 == 0){
30         Matrix res = fast_pow(n >> 1);
31         return res * res;
32     }
33     Matrix res = fast_pow(n >> 1);
34     return base * res * res;
35 }
36
37 int main(){
38     IOS
39     base.mat[0][0] = 1;
40     base.mat[0][1] = 1;
41     base.mat[1][0] = 1;
42     base.mat[1][1] = 0;
43     Matrix output = fast_pow(20);
44     cout << output.mat[0][0] << '\n';
45
46     return 0;
47 }

```