

# Contents

<b>1 Basic</b>	10
1.1 Disjoint Set	11
1.2 int128	12
<b>2 Math</b>	13
2.1 sieve	14
2.2 isPrime	15
2.3 genPrime.py	16
2.4 數列	17
2.4.1 Fibonacci	18
2.4.2 Tribonacci	19
2.5 extgcd	20
2.6 matrix	21
2.7 GaussElimination	22
<b>3 Graph</b>	23
3.1 最短路	24
3.1.1 dijkstra	25
3.1.2 SPFA	26
3.1.3 floyd	27
3.2 MST	28
3.2.1 kruskal	29
3.2.2 prim	30
<b>4 DP</b>	31
4.1 背包	32
<b>5 Sequence</b>	33
5.1 RMQ	34
5.1.1 seg-tree	35
5.1.2 sparse table	36
5.2 lcs	37
<b>6 String</b>	38
6.1 hash	39
6.2 lcs	40
<b>7 Ad-hoc</b>	41
7.1 n 皇后	42

## 1 Basic

### 1.1 Disjoint Set

```

1 #define SIZE 1000005
2 int p[SIZE];
3
4 void init()
5 {
6     for(int i = 0; i < SIZE; i++)
7         p[i] = i;
8 }
9
10 int find(const int x)
11 {
12     return x==p[x] ? (p[x] = find(p[x]));
13 }
14
15 void uni(const int a, const int b)
16 {
17     p[find(a)] = p[find(b)];
18 }
19
20 bool equ(const int a, const int b)
21 {
22     return find(a) == find(b);
23 }

```

### 1.2 int128

```

1 static int print_i128(__int128 i128)
2 {
3     char ch128[40], *now = ch128, *head = ch128;
4     int len = 0;
5
6     if(i128 < 0)

```

```

7 {
8     putchar('-');
9     i128 = -i128;
10 }
11 // Turn __int128 into char[] from lowest digit
12 while(i128 > 9)
13 {
14     *now++ = i128 % 10 + '0';
15     i128 /= 10;
16 }
17 *now = i128 + '0';
18
19 // Print
20 while(now >= head)
21 {
22     putchar(*now--);
23 }
24
25 return 1;
26 }
27
28 static int scan_i128(__int128 *n)
29 {
30     #ifdef DBG
31     assert(n != NULL);
32     #endif
33     char num[40], *now = num;
34     bool minus = false;
35     *n = 0; // reset n
36
37     int ret = scanf("%s", num);
38     if(ret == EOF) // scanf fails
39         return EOF;
40     // Judge if minus
41     if(*now == '-')
42     {
43         minus = true;
44         now++; // skip '-'
45     }
46
47     // Add the digit and multiply it by 10 one after
48     // another
49     while(*now)
50     {
51         *n += *now - '0';
52         now++;
53         if(*now) // check if now touches '\0'
54             *n *= 10;
55     }
56     *n = minus ? -(*n) : *n;
57
58     return 1;
59 }

```

## 2 Math

### 2.1 sieve

```

1 #define TABLE_SIZE 100000
2 bool prime[TABLE_SIZE];
3 void buildPrimeTable()
4 {
5     prime[0] = prime[1] = false;
6     for(int i = 2; i < TABLE_SIZE; i++)
7         prime[i] = true;
8
9     for(int i = 2; i < TABLE_SIZE; i++)
10     {
11         if(prime[i])
12             for(size_t a = i*i; a < TABLE_SIZE; a += i)
13                 prime[a] = false;
14     }
15 }

```

## 2.2 isPrime

```

1 // O(sqrt(n))
2 bool isPrime(int n)
3 {
4     for(int i = 2; i <= sqrt(n); i++)
5         if(n % i == 0)
6             return false;
7     return true;
8 }
9 ///////////////////////////////////////////////////
10 // fast
11 bool prime[50000];
12 vector<size_t> vec;
13
14 // 篩法
15 void sie()
16 {
17     fill(prime, prime+50000, true);
18
19     for (int i = 2; i < 50000; ++i)
20         if (prime[i])
21         {
22             vec.push_back(i);
23             for (int j = i + i; j < 50000; j += i)
24                 prime[j] = false;
25         }
26 }
27
28 bool isPrime(size_t x)
29 {
30     if (x < 50000 && !prime[x])
31         return false;
32     if (x < 50000 && prime[x])
33         return true;
34     size_t sqr = sqrt(x);
35
36     if (x % 6 == 0 || x % 6 == 2
37         || x % 6 == 3 || x % 6 == 4)
38     {
39         return false;
40     }
41     for (size_t i : vec)
42     {
43         if (x % i == 0)
44             return false;
45         if (sqr <= i)
46             break;
47     }
48     return true;
49 }

```

## 2.3 genPrime.py

```

1 import math
2 n = 10000
3 for i in range(1, n+1):
4     pt = True
5     for a in range(2, (int)(math.sqrt(n))+1):
6         if i % a == 0:
7             pt = False
8     if pt:
9         print(i, end=', ')

```

## 2.4 數列

### 2.4.1 Fibonacci

$$a_n = a_{n-1} + a_{n-2}, \quad a_0 = 1, \quad a_1 = 1$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233...

### 2.4.2 Tribonacci

$$a_n = a_{n-1} + a_{n-2} + a_{n-3}, \quad a_0 = 1, \quad a_1 = 1, \quad a_2 = 1$$

1, 1, 1, 3, 5, 9, 17, 31, 57, 105, 193, 355, 653, 1201, 2209, 4063, 7473, 13745, 25281

## 2.5 extgcd

```

1 int extgcd(int a, int b, int &x, int &y)
2 {
3     int d = a;
4     if(b) { d = extgcd(b, a%b, y, x); y -= (a/b)*x; }
5     else x = 1, y = 0;
6     return d;
7 }
8 //ax+by=1 ax同餘 1 mod b

```

## 2.6 matrix

```

1 template<typename T, int N=2>
2 struct Mat { //Matrix
3     unsigned long long v[N][N];
4     Mat operator*(Mat b) const {
5         Mat val;
6         for (int i = 0; i < N; i++) {
7             for (int j = 0; j < N; j++) {
8                 val.v[i][j] = 0;
9                 for (int k = 0; k < N; k++) {
10                     val.v[i][j] += v[i][k] * b.v[k][j];
11                 }
12             }
13         }
14         return val;
15     }
16     friend ostream& operator<<(ostream& out, Mat m)
17     {
18         for(int i = 0; i < N; i++)
19             for(int j = 0; j < N; j++)
20                 out << m.v[i][j] << (j==N-1 ? "\n" : " ");
21         return out;
22     }
23 };
24
25 // 用法
26 Mat<int> a, b;
27
28 a.v[0][0] = 1;
29 a.v[0][1] = 2;
30 a.v[1][0] = 3;
31 a.v[1][1] = 4;
32
33 b.v[0][0] = 5;
34 b.v[0][1] = 6;
35 b.v[1][0] = 7;
36 b.v[1][1] = 8;
37 cout << a << b << a*b;

```

## 2.7 GaussElimination

```

1 const int MAXN = 300;
2 const double EPS = 1e-8;
3 int n;
4 double A[MAXN][MAXN];
5 void Gauss() {
6     for(int i = 0; i < n; i++) {
7         bool ok = 0;
8         for(int j = i; j < n; j++) {
9             if(fabs(A[j][i]) > EPS) {
10                 swap(A[j], A[i]);
11                 ok = 1;
12                 break;
13             }
14         }
15     }
16 }

```

```

15     if(!ok) continue;
16     double fs = A[i][i];
17     for(int j = i+1; j < n; j++) {
18         double r = A[j][i] / fs;
19         for(int k = i; k < n; k++) {
20             A[j][k] -= A[i][k] * r;
21         }
22     }
23 }
24 }

```

### 3 Graph

#### 3.1 最短路

##### 3.1.1 dijkstra

```

1 #define N 500
2 #define E 10000
3 typedef pair<int, int> P;
4 #define INF 2e9
5
6 // 【注意】 0-index
7
8 vector<P> e[E]; // (to, weight) edge
9 int d[N]; // 記錄起點到各個點的最短路徑長度
10 int parent[N]; // 記錄各個點在最短路徑樹上的父親是誰
11 int n, m; // n 個點 m個邊
12 // 沒有要找路徑的話可以拔掉parent[]
13 void dijkstra(int src)
14 {
15     for (int i = 0; i < n; i++) d[i] = INF;
16
17     // (dis, idx)
18     priority_queue<P, vector<P>, greater<P> > pq;
19     pq.emplace(0, src);
20
21     parent[src] = src;
22
23     while(!pq.empty())
24     {
25         int a = -1, b = -1, wei;
26
27         // 從pq裡拿出最小的點
28         tie(wei, a) = pq.top();
29         pq.pop();
30
31         // 如果拜訪過了(當過pq.top)
32         if(d[a] != INF) continue;
33
34         d[a] = wei;
35         for(auto i : e[a])
36         {
37             if(d[i.first] == INF)
38             {
39                 pq.emplace(wei + i.second, i.first);
40                 parent[i.first] = a; // set parent node
41             }
42         }
43     }
44 }
45
46 void findPath(int src, int start)
47 {
48     if(start != src)
49         findPath(src, parent[start]);
50     cout << start << ' ';
51 }

```

##### 3.1.2 SPFA

```

1 // ATTENTION: 0-index
2
3 #define N 505
4 #define E 20005
5 #define INF 2e9
6 typedef pair<int, int> P;
7
8 vector<P> e[E]; // (to, weight) 邊
9
10 int d[N]; // distance of each node from the src
11 // node
12 int parent[N]; // parent idx pf a node
13 bool inq[N]; // inqueue
14 int cnt[N]; // path updating counter
15 bool del[N]; // 被刪掉的node, 用於判斷
16
17 int n, m; // vertice, edges
18
19 void dfs(int src)
20 {
21     int to, wei;
22
23     del[src] = true;
24     for(auto i : e[src])
25     {
26         tie(to, wei) = i;
27         // 如果沒刪除的話
28         if(!del[to])
29             dfs(to);
30     }
31 }
32
33 void spfa(int src)
34 {
35     // init all d[]
36     for(int i = 0; i < n; i++) d[i] = INF;
37     // init src node
38     d[src] = 0;
39     parent[src] = src;
40     cnt[src] = 0;
41
42     queue<int> q; // node queue
43
44     q.push(src);
45
46     while(!q.empty())
47     {
48         int cur = q.front(), to, wei;
49         q.pop(); // take out
50         inq[cur] = false;
51
52         for(auto i : e[cur])
53         {
54             tie(to, wei) = i;
55             if(!del[to] && d[cur] + wei < d[to])
56             {
57                 d[to] = d[cur] + wei;
58                 parent[to] = cur;
59
60                 if(!inq[to])
61                 {
62                     cnt[to]++;
63
64                     if(cnt[to] >= n)
65                     {
66                         dfs(to);
67                         continue;
68                     }
69
70                     q.push(to);
71                     inq[to] = true;
72                 }
73             }
74         }
75     }
76
77     // 無限小
78     for(int i = 0; i < n; i++)

```

```

77     if(del[i])
78         d[i] = -INF;
79 }
80
81 void findPath(int src, int end)
82 {
83     if(src != end)
84         findPath(src, parent[end]);
85     cout << end << ' ';
86 }

```

### 3.1.3 floyd

```

1 #define INF 0x3f3f3f3f
2 #define N 100
3
4 int d[N][N]; // 0-index
5 int p[N][N]; // path
6 int n;       // n vertice
7
8 void floyd_warshall()
9 {
10     for(int i = 0; i < n; i++)
11         for(int j = 0; j < n; j++)
12             p[i][j] = (d[i][j] == INF || d[i][j] == 0
13                 ? -1 : i);
14
15     for(int k = 0; k < n; k++)
16         for(int i = 0; i < n; i++)
17             for(int j = 0; j < n; j++)
18                 if(d[i][j] > d[i][k] + d[k][j])
19                     {
20                         d[i][j] = d[i][k] + d[k][j];
21                         p[i][j] = p[k][j];
22                     }
23 }
24
25 void findPath(int src, int end)
26 {
27     if(p[src][end] != -1)
28         findPath(src, p[src][end]);
29
30     cout << end << ' ';
31 }

```

## 3.2 MST

### 3.2.1 kruskal

```

1 #define V 50000
2 #define E 200000
3
4 // disjoint set
5 int ds[V];
6 void init() { for(int i = 0; i < V; i++) ds[i] = i; }
7 int find(int i)
8 { return ds[i] == i ? i : (ds[i] = find(ds[i])); }
9 void uni(int a, int b) { ds[find(a)] = find(b); }
10
11 struct edge
12 {
13     int fr, to, wei;
14     void setEdge(int f, int t, int w) { fr = f; to = t;
15         wei = w; }
16     friend bool operator<(edge &lhs, edge &rhs)
17     { return lhs.wei < rhs.wei; }
18 }e[E];
19 // n vertice, m edges
20 int n, m;
21
22 int kruskal()
23 {
24     init();

```

```

25     sort(e, e+m);
26     // i -> cur vectex, j -> cur edge
27     int total = 0, i, j;
28     for(i = 0, j = 0; i < n-1 && j < m; i++, j++)
29     {
30         // if it's in the same group, skipping it.
31         while(find(e[j].fr) == find(e[j].to))
32             j++;
33
34         uni(e[j].fr, e[j].to);
35
36         total += e[j].wei;
37     }
38
39     return i == n-1 ? total : -1;
40 }
41 }

```

### 3.2.2 prim

```

1 #define INF 0x3f3f3f3f
2 #define F first
3 #define S second
4
5 #define V 50000
6 #define E 200000
7
8 typedef pair<int, int> P;
9 vector<P> e[E]; // e[from] => (to, weight)
10
11 // 無向圖
12 void addEdge(int from, int to, int weight)
13 {
14     e[from].emplace_back(to, weight);
15     e[to].emplace_back(from, weight);
16 }
17
18 int d[V];
19 int nt[V];
20
21 // n個點, m個邊
22 int n, m;
23
24 int prim(int src)
25 {
26     for(int i = 0; i < V; i++) d[i] = INF;
27     // 放點的暫時權重 (weight, idx)
28     priority_queue<P, vector<P>, greater<P>> > pq;
29     int total = 0, v = 0, pre = -1; // 總和, 找到的點
30     // 加入起點
31     pq.emplace(0, src);
32
33     int to, wei;
34     while(!pq.empty())
35     {
36         auto cur = pq.top();
37         pq.pop();
38         // 如果點cur已經當過top(不是第一次)
39         if(d[cur.S] != INF)
40             continue;
41         // 更新該點的d[], 代表選了該點
42         d[cur.S] = 0;
43         nt[pre] = cur.S;
44         pre = cur.S;
45         total += cur.F;
46         v++;
47         // 遍歷所有跟點cur相連的邊
48         for(auto i : e[cur.S])
49         {
50             tie(to, wei) = i;
51             // 如果點to沒有選到過
52             if(d[to] == INF)
53             {
54                 // 看有沒有更小的權重
55                 if(wei < d[to])

```

```

56         pq.emplace(wei, to);
57     }
58     else
59         pq.emplace(d[to], to);
60 }
61 }
62 }
63 // 如果生成樹有n個點(全部找到)則輸出，否則輸出-1
64 return v == n ? total : -1;
65 }
66
67 void findPath(int src)
68 {
69     for(int i = 0; i < n; i++)
70         if(visited[i] != src+1)
71             cout << i+1 << ' ' << visited[i]+1 << '\n';
72 }

```

## 4 DP

### 4.1 背包

```

1  #define W 1000          // 背包最重 W
2  #define N 100           // 最多 N 種物品
3
4  int weight[N];          // 物品重量
5  int value[N];           // 物品價值
6  int bag[W][2];
7
8  // 0/1 背包
9  void ZeroOne(){
10     memset(bag, 0, sizeof(bag));
11     for(int i = 0; i < N; i++){
12         for(int j = 0; j < W; j++){
13             if( j >= weight[i] )
14                 bag[j][1] = max( bag[j][0],
15                                 bag[j-weight[i]][0] + value[i] );
16         }
17         for(int j = 0; j < W; j++){
18             bag[j][0] = bag[j][1];
19         }
20     }
21
22     // 無限背包
23     void Unlimited(){
24         memset(bag, 0, sizeof(bag));
25         for(int i = 0; i < N; i++){
26             for(int j = 0; j < W; j++){
27                 if( j >= weight[i] )
28                     bag[j][1] = max( bag[j][0],
29                                     bag[j-weight[i]][1] + value[i] );
30             }
31             for(int j = 0; j < W; j++){
32                 bag[j][0] = bag[j][1];
33             }
34         }
35
36         // 分組背包
37         int group;        // 有幾組
38         int how_many;     // 一組幾個
39         int WEIGHT, VALUE;
40
41         void Grouping(){
42             memset(bag, 0, sizeof(bag));
43             for(int i = 0; i < group; i++){
44                 for(int j = 0; j < how_many; j++){
45                     scanf("%d %d", &WEIGHT, &VALUE);
46                     for(int k = 0; k < W; k++){
47                         if( j >= WEIGHT ){
48                             bag[j][1] = max( bag[j][1],
49                                             bag[j-WEIGHT][0] + VALUE );
50                         }
51                     }
52                 }
53             }
54         }
55     }

```

```

52     }
53     for(int j = 0; j < W; j++){
54         bag[j][0] = bag[j][1];
55     }
56 }
57
58 // 多重背包
59 int limit[N]; // 物品上限
60
61 void Multiple(){
62     for(int i = 0; i < N; i++){
63         int tmp = 1;
64         bool check = true;
65
66         while(true){
67             if( tmp == 0 )
68                 break;
69
70             for(int j = 0; j < W; j++){
71                 if( j >= weight[i]*tmp )
72                     bag[j][1] = max( bag[j-weight[i]*tmp][0],
73                                     bag[j-weight[i]*tmp][1] + value[i]*tmp );
74             }
75             for(int j = 0; j < W; j++){
76                 bag[j][0] = bag[j][1];
77             }
78
79             if(!check)
80                 break;
81             else if( limit[i] - tmp*2 + 1 <= tmp*2 ){
82                 tmp = limit[i] - tmp*2 + 1;
83                 check = false;
84             }
85             else
86                 tmp = tmp*2;
87         }
88     }
89 }

```

## 5 Sequence

### 5.1 RMQ

#### 5.1.1 seg-tree

```

1  #define MAX_S 10000
2
3  // 【注意】 1-index
4
5  int t[4*MAX_S+5];
6  int in[MAX_S+5];
7
8  #define LEFT(x) ((x)<<1)
9  #define RIGHT(x) (((x)<<1)+1)
10
11 // Build the segment tree
12 // parent, inputL, inputR
13 void buildSeg(int p, int inL, int inR)
14 {
15     if(inL == inR)
16     {
17         t[p] = in[inL];
18         return;
19     }
20
21     int mid = (inL+inR)/2;
22     buildSeg(LEFT(p), inL, mid);
23     buildSeg(RIGHT(p), mid+1, inR);
24     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
25 }
26
27 // Modify single point, and maintain the segment tree
28 // parent, left, right, idx, val
29

```

```

30 void modify(int p, int L, int R, int i, int x)
31 {
32     // stop point
33     if(i == L && L == R)
34     {
35         t[p] = x;
36         return;
37     }
38
39     int mid = (L+R) / 2;
40     if(i <= mid)
41         modify(LEFT(p), L, mid, i, x);
42     else
43         modify(RIGHT(p), mid+1, R, i, x);
44     // update this node
45     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
46 }
47
48 // Query in [quL, quR]
49 // parnet, left, right, queryL, queryR
50 int query(int p, int L, int R, int quL, int quR)
51 {
52     // stop point
53     if(quL <= L && R <= quR)
54     {
55         return t[p];
56     }
57
58     int mid = (L+R) / 2;
59     if(quR <= mid) // left
60         return query(LEFT(p), L, mid, quL, quR);
61     else if(mid < quL) // right
62         return query(RIGHT(p), mid+1, R, quL, quR);
63     else // middle
64         return max(query(LEFT(p), L, mid, quL, quR),
65                     query(RIGHT(p), mid+1, R, quL, quR));
66 }

```

### 5.1.2 sparse table

```

1 // Sparse Table (1-index)
2 int N = 14, logN = __lg(N), spI = logN+1;
3 int sp[spI][N] = {0};
4
5 void buildST()
6 {
7     // first row (only one in a group)
8     for(int i = 0; i < N; i++)
9         sp[0][i] = value[i];
10    // number of elements in a group = 2^i
11    for(int i = 1; i < spI; i++)
12    {
13        // j < N - (2^i - 1)
14        for(int j = 0; j < N - ((1 << i) - 1); j++)
15        {
16            // Current row overlapped two upper
17            // groups in (i-1) row
18            sp[i][j] = max(sp[i-1][j], sp[i-1][j+(1 << (i-1))]);
19        }
20    }
21
22    // Query
23    int query(int l, int r)
24    {
25        l--, r--;
26
27        int distance = r - l + 1;
28        int targetIdx = 1 != r ? __lg(distance)-1 : 0;
29
30        return max(sp[targetIdx][l], sp[targetIdx][r - (1<<targetIdx - 1)]);
31    }

```

## 5.2 lcs

```

1 #define LEN 100
2
3 char s1[LEN];
4 char s2[LEN];
5 int length[LEN + 1][LEN + 1];
6 // 記錄每一格的結果是從哪一格而來
7 int preve[LEN + 1][LEN + 1];
8 int lcs[LEN];
9
10 void print_LCS_s1(int i, int j)
11 {
12     if (i == 0 || j == 0) return;
13
14     if (preve[i][j] == 0)
15     {
16         print_LCS_s1(i-1, j-1);
17         cout << s1[i];
18     }
19     else if (preve[i][j] == 1)
20         print_LCS_s1(i, j-1);
21     else if (preve[i][j] == 2)
22         print_LCS_s1(i-1, j);
23 }
24
25 void print_LCS_s2(int i, int j)
26 {
27     if (i == 0 || j == 0) return;
28
29     if (preve[i][j] == 0)
30     {
31         print_LCS_s2(i-1, j-1);
32         cout << s2[i];
33     }
34     else if (preve[i][j] == 1)
35         print_LCS_s2(i, j-1);
36     else if (preve[i][j] == 2)
37         print_LCS_s2(i-1, j);
38 }
39
40 void LCS(int n1, int n2, char s1[], char s2[])
41 {
42
43     for (int i = 0; i <= n1; i++)
44         length[i][0] = 0;
45     for (int j = 0; j <= n2; j++)
46         length[0][j] = 0;
47
48     if (n1 > n2)
49     {
50         for (int i = 1; i <= n1; i++)
51             for (int j = 1; j <= n2; j++) // 兩層for
52                 if (s1[i] == s2[j])
53                 {
54                     length[i][j] = length[i-1][j-1] + 1;
55                     preve[i][j] = 0; // 左上方
56                 }
57             else
58             {
59                 if (length[i-1][j] < length[i][j-1])
60                 {
61                     length[i][j] = length[i][j-1];
62                     preve[i][j] = 1; // 左方
63                 }
64                 else
65                 {
66                     length[i][j] = length[i-1][j];
67                     preve[i][j] = 2; // 上方
68                 }
69             }
70         //
71         cout << "LCS_Len:" << length[n1][n2] << '\n';
72         cout << "LCS:";
73         print_LCS_s1(n1, n2);
74         //從right down開始

```

```

75     }
76     else
77     {
78         for (int i = 1; i <= n2; i++)
79         for (int j = 1; j <= n1; j++) // 兩層for
80             if ( s2[i] == s1[j] )
81             {
82                 length[i][j] = length[i-1][j-1] + 1;
83                 preve[i][j] = 0; // 左上方
84             }
85             else
86             {
87                 if (length[i-1][j] < length[i][j-1])
88                 {
89                     length[i][j] = length[i][j-1];
90                     preve[i][j] = 1; // 左方
91                 }
92                 else
93                 {
94                     length[i][j] = length[i-1][j];
95                     preve[i][j] = 2; // 上方
96                 }
97             }
98         //
99         cout << "LCS_Len:" << length[n2][n1] << '\n';
100        cout << "LCS:";
101        print_LCS_s2(n2, n1);
102    }
103 }
104 // 用法
105 LCS(s1_length, s2_length, s1, s2);

```

## 6 String

### 6.1 hash

```

1 size_t bkdr(const char str[]){
2     // 31 131 1313 13131 131313 etc..
3     size_t seed = 131;
4     size_t hash = 0;
5
6     while (*str){
7         hash = hash * seed + (*str++);
8     }
9
10    return (hash & 0x7FFFFFFF);
11 }
12
13 // C++ build-in hash
14 hash<string> hash_gen;
15 size_t num = hash_gen(str);

```

### 6.2 lcs

```

1 // LCString
2 string LCStr(string &s1, string &s2)
3 {
4     string rec;
5     int arr[100][100];
6
7     memset(arr, 0, sizeof(arr));
8
9     int maxx = 0;
10    int position = 0;
11
12    for (int i = 1; i <= s1.length(); ++i)
13    {
14        for (int j = 1; j <= s2.length(); ++j)
15        {
16            if (s1[i-1] == s2[j-1])
17            {
18                arr[i][j] = arr[i-1][j-1] + 1;

```

```

19
20         if (maxx < arr[i][j])
21         {
22             position = i;
23             maxx = arr[i][j];
24         }
25     }
26     else
27     {
28         arr[i][j] = 0;
29     }
30 }
31
32 for (int k = position - maxx; k < position; ++k)
33 {
34     rec += s1[k];
35 }
36 return rec;
37 }

```

## 7 Ad-hoc

### 7.1 n 皇后

```

1 int Queen[37000][14];
2 int Tmp[14];
3 int total=0;
4 int Row[14]={0}, Left[27]={0}, Right[27]={0};
5
6 void N_Queen(int k, int Number){
7     int i, j;
8     if(k==Number){
9         for(j=0; j<Number; j=j+1){
10             Queen[total][j]=Tmp[j];
11         }
12         total=total+1;
13         return;
14     }
15     for(i=0; i<Number; i=i+1){
16         int right= k+i;
17         int left= k-i+Number-1;
18         if( !Row[i] && !Left[left] && !Right[right] ){
19             Row[i]=1;
20             Left[left]=1;
21             Right[right]=1;
22
23             Tmp[k]=i;
24
25             N_Queen(k+1, Number);
26
27             Row[i]=0;
28             Left[left]=0;
29             Right[right]=0;
30
31         }
32     }
33 }
34
35 // 用法
36 N_Queen(0, num);

```