Contents

1 Basic

1.1 Disjoint Set

```cpp
#define SIZE 1000005
int p[SIZE];

void init()
{
    for(int i = 0; i < SIZE; i++)
        p[i] = i;
}

int find(const int x)
{
    return x==p[x] ? (p[x] = find(p[x]));
}

void uni(const int a, const int b)
{
    p[find(a)] = p[find(b)];
}

bool equ(const int a, const int b)
{
    return find(a) == find(b);
}
```

1.2 int128

```cpp
static int print_i128(__int128 i128)
{
    char ch128[40], *now = ch128, *head = ch128;
    int len = 0;

    if(i128 < 0)
    {
        putchar('-');
        i128 = -i128;
    }
    // Turn __int28 into char[] from lowest digit
    while(i128 > 9)
    {
        *now++ = i128 % 10 + '0';
        i128 /= 10;
    }
    *now = i128 + '0';

    // Print
    while(now >= head)
    {
        putchar(*now--);
    }

    return 1;
}

static int scan_i128(__int128 *n)
{
    #ifdef DBG
    assert(n != NULL);
    #endif
    char num[40], *now = num;
    bool minus = false;
    *n = 0; // reset n

    int ret = scanf("%s", num);
    if(ret == EOF) // scanf fails
        return EOF;
    // Judge if minus
    if(*now == '-')
    {
        minus = true;
        now++; // skip '-'
    }

    // Add the digit and multiply it by 10 one after another
    while(*now)
    {
        *n += *now - '0';
        now++;
        if(*now) // check if now touches '\0'
            *n *= 10;
    }

    *n = minus ? -(*n) : *n;

    return 1;
}
```

2 Math

2.1 sieve

```cpp
#define TABLE_SIZE 100000
bool prime[TABLE_SIZE];
void buildPrimeTable()
{
    prime[0] = prime[1] = false;
    for(int i = 2; i < TABLE_SIZE; i++)
        prime[i] = true;

    for(int i = 2; i < TABLE_SIZE; i++)
    {
        if(prime[i])
            for(size_t a = i*i; a < TABLE_SIZE; a += i)
                prime[a] = false;
    }
}
```

2.2 isPrime

```cpp
// O(sqrt(n))
bool isPrime(int n)
{
    for(int i = 2; i <= sqrt(n); i++)
        if(n % i == 0)
            return false;
    return true;
}
//////////////////////////////////////////////////////
// fast
bool prime[50000];
vector <size_t> vec;

// 篩法
void sie()
{
    fill(prime, prime+50000, true);

    for (int i = 2; i < 50000; ++i)
        if (prime[i])
        {
            vec.push_back(i);
            for (int j = i + i; j < 50000; j += i)
                prime[j] = false;
        }
}

bool isPrime(size_t x)
{
    if (x < 50000 && !prime[x])
        return false;
    if (x < 50000 && prime[x])
        return true;
    size_t sqr = sqrt(x);

    if (x % 6 == 0 || x % 6 == 2
     || x % 6 == 3 || x % 6 == 4)
    {
        return false;
    }
    for (size_t i : vec)
    {
        if (x % i == 0)
            return false;
        if (sqr <= i)
            break;
    }
    return true;
}
```

2.3 genPrime.py

```python
import math
n = 10000
for i in range(1, n+1):
    pt = True
    for a in range(2, (int)(math.sqrt(n))+1):
        if i % a == 0:
            pt = False
    if pt:
        print(i, end=',')
```

2.4 數列

2.4.1 Fibonacci

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

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

2.4.2 Tribonacci

$$a_n = a_{n-1} + a_{n-2} + a_{n-3},\ a_0 = 1,\ a_1 = 1,\ a_3 = 1$$

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

2.5 extgcd

```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
```

2.6 matrix

```cpp
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;
    }
    friend ostream& operator<<(ostream& out, Mat m)
    {
        for(int i = 0; i < N; i++)
          for(int j = 0; j < N; j++)
            out << m.v[i][j] << (j==N-1 ? "\n" : " ");
        return out;
    }
};

// 用法
Mat<int> a, b;

a.v[0][0] = 1;
a.v[0][1] = 2;
a.v[1][0] = 3;
a.v[1][1] = 4;

b.v[0][0] = 5;
b.v[0][1] = 6;
b.v[1][0] = 7;
b.v[1][1] = 8;
cout << a << b << a*b;
```

2.7 GaussElimination

```cpp
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;
      }
    }
  }
}
```

3 Graph

3.1 最短路

3.1.1 dijkstra

```
#define N 500
#define E 10000
typedef pair<int, int> P;
#define INF 2e9

//  【注意】  0-index

vector<P> e[E]; // (to, weight) edge
int d[N];       // 記錄起點到各個點的最短路徑長度
int parent[N];  // 記錄各個點在最短路徑樹上的父親是誰
int n, m;       // n 個點 m個邊
// 沒有要找路徑的話可以拔掉parent[]
void dijkstra(int src)
{
    for (int i = 0; i < n; i++) d[i] = INF;

    // (dis, idx)
    priority_queue<P, vector<P>, greater<P> > pq;
    pq.emplace(0, src);

    parent[src] = src;

    while(!pq.empty())
    {
        int a = -1, b = -1, wei;

        // 從pq裡拿出最小的點
        tie(wei, a) = pq.top();
        pq.pop();

        // 如果拜訪過了(當過pq.top)
        if(d[a] != INF) continue;

        d[a] = wei;
        for(auto i : e[a])
        {
            if(d[i.first] == INF)
            {
                pq.emplace(wei + i.second, i.first);
                parent[i.first] = a; // set parent node
            }
        }
    }
}

void findPath(int src, int start)
{
    if(start != src)
        findPath(src, parent[start]);
    cout << start << ' ';
}
```

3.1.2 SPFA

```
// ATTENTION: 0-index

#define N 505
#define E 20005
#define INF 2e9
typedef pair<int, int> P;

vector<P> e[E];// (to, weight) 邊

int d[N];     // distance of each node from the src
    node
int parent[N];// parent idx of a node
bool inq[N];  // inqueue
int cnt[N];   // path updating counter
bool del[N];  // 被刪掉的node，用於判斷

int n, m;     // vertice, edges

void dfs(int src)
{
    int to, wei;

    del[src] = true;
    for(auto i : e[src])
    {
        tie(to, wei) = i;
        // 如果沒刪除的話
        if(!del[to])
            dfs(to);
    }
}

void spfa(int src)
{
    // init all d[]
    for(int i = 0; i < n; i++) d[i]=  INF;
    // init src node
    d[src] = 0;
    parent[src] = src;
    cnt[src] = 0;

    queue<int> q; // node queue

    q.push(src);

    while(!q.empty())
    {
        int cur = q.front(), to, wei;
        q.pop();            // take out
        inq[cur] = false;

        for(auto i : e[cur])
        {
            tie(to, wei) = i;
            if(!del[to] && d[cur] + wei < d[to])
            {
                d[to] = d[cur] + wei;
                parent[to] = cur;

                if(!inq[to])
                {
                    cnt[to]++;

                    if(cnt[to] >= n)
                    {
                        dfs(to);
                        continue;
                    }

                    q.push(to);
                    inq[to] = true;
                }
            }
        }
    }
    // 無限小
    for(int i = 0; i < n; i++)
```

```
        if(del[i])
            d[i] = -INF;
}

void findPath(int src, int end)
{
    if(src != end)
        findPath(src, parent[end]);
    cout << end << ' ';
}
```

3.1.3 floyd

```
#define INF 0x3f3f3f3f
#define N 100

int d[N][N]; // 0-index
int p[N][N]; // path
int n;       // n vertice

void floyd_warshall()
{
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            p[i][j] = (d[i][j] == INF || d[i][j] == 0
                        ? -1 : i);

    for(int k = 0; k < n; k++)
        for(int i = 0; i < n; i++)
            for(int j = 0; j < n; j++)
                if(d[i][j] > d[i][k] + d[k][j])
                {
                    d[i][j] = d[i][k] + d[k][j];
                    p[i][j] = p[k][j];
                }
}

void findPath(int src, int end)
{
    if(p[src][end] != -1)
        findPath(src, p[src][end]);

    cout << end << ' ';
}
```

3.2 MST

3.2.1 kruskal

```
#define V 50000
#define E 200000

// disjoint set
int ds[V];
void init() { for(int i = 0; i < V; i++) ds[i] = i; }
int find(int i)
{ return ds[i] == i ? i : (ds[i] = find(ds[i])); }
void uni(int a, int b) { ds[find(a)] = find(b); }

struct edge
{
    int fr, to, wei;
    void setEdge(int f, int t, int w) { fr = f; to = t;
         wei = w;}
    friend bool operator<(edge &lhs, edge &rhs)
    { return lhs.wei < rhs.wei; }
}e[E];

// n vertice, m edges
int n, m;

int kruskal()
{
    init();

    sort(e, e+m);
    // i -> cur vectex, j -> cur edge
    int total = 0, i, j;
    for(i = 0, j = 0; i < n-1 && j < m; i++, j++)
    {
        // if it's in the same group, skipping it.
        while(find(e[j].fr) == find(e[j].to))
            j++;

        uni(e[j].fr, e[j].to);

        total += e[j].wei;
    }

    return i == n-1 ? total : -1;
}
```

3.2.2 prim

```
#define INF 0x3f3f3f3f
#define F first
#define S second

#define V 50000
#define E 200000

typedef pair<int, int> P;
vector<P> e[E]; // e[from] => (to, weight)

// 無向圖
void addEdge(int from, int to, int weight)
{
    e[from].emplace_back(to, weight);
    e[to].emplace_back(from, weight);
}

int d[V];
int nt[V];

// n個點, m個邊
int n, m;

int prim(int src)
{
    for(int i = 0; i < V; i++) d[i] = INF;
    // 放點的暫時權重 (weight, idx)
    priority_queue<P, vector<P>, greater<P> > pq;
    int total = 0, v = 0, pre = -1; // 總和, 找到的點
    // 加入起點
    pq.emplace(0, src);

    int to, wei;
    while(!pq.empty())
    {
        auto cur = pq.top();
        pq.pop();
        // 如果點cur已經當過top(不是第一次)
        if(d[cur.S] != INF)
            continue;
        // 更新該點的d[], 代表選了該點
        d[cur.S] = 0;
        nt[pre] = cur.S;
        pre = cur.S;
        total += cur.F;
        v++;
        // 遍歷所有跟點cur相連的邊
        for(auto i : e[cur.S])
        {
            tie(to, wei) = i;
            // 如果點to沒有選到過
            if(d[to] == INF)
            {   // 看有沒有更小的權重
                if(wei < d[to])
                {
```

```
                    pq.emplace(wei, to);
                }
                else
                    pq.emplace(d[to], to);
            }
        }
    }
    // 如果生成樹有n個點(全部找到)則輸出，否則輸出-1
    return v == n ? total : -1;
}

void findPath(int src)
{
    for(int i = 0; i < n; i++)
        if(nt[i]+1 != src+1)
            cout << i+1 << ' ' << nt[i]+1 << '\n';
}
```

4 DP

4.1 背包

```
#define W 1000          // 背包最重 W
#define N 100           // 最多 N 種物品

int weight[N];          //物品重量
int value[N];           //物品價值
int bag[W][2];

// 0/1 背包
void ZeroOne(){
  memset(bag,0,sizeof(bag));
  for(int i = 0; i < N; i++ ){
    for(int j = 0; j < W; j++ )
      if( j >= weight[i] )
        bag[j][1] = max( bag[j][0]
          ,bag[j-weight[i]][0] + value[i] );
    for(int j = 0; j < W; j++ )
      bag[j][0] = bag[j][1];
  }
}

// 無限背包
void Unlimited(){
  memset(bag,0,sizeof(bag));
  for(int i = 0; i < N; i++ ){
    for(int j = 0; j < W; j++ )
      if( j >= weight[i] )
        bag[j][1] = max( bag[j][0]
          ,bag[j-weight[i]][1] + value[i] );
    for(int j = 0; j < W; j++ )
      bag[j][0] = bag[j][1];
  }
}

// 分組背包
int group;    //有幾組
int how_many; //一組幾個
int WEIGHT,VALUE;

void Grouping(){
  memset(bag,0,sizeof(bag));
  for(int i = 0; i < group; i++ ){
    for(int j = 0; j < how_many; j++ ){
      scanf("%d %d", &WEIGHT, &VALUE);
      for(int k = 0; k < W; k++ ){
        if( j >= WEIGHT ){
          bag[j][1] = max( bag[j][1]
            ,bag[j][0] );
          bag[j][1] = max( bag[j][1]
            ,bag[j-WEIGHT][0] + VALUE );
        }
      }
    }
    for(int j = 0; j < W; j++ )
      bag[j][0] = bag[j][1];
  }
}

// 多重背包
int limit[N]; //物品上限

void Multiple(){
  for(int i = 0 ; i < N ; i++ ){
    int tmp = 1;
    bool check = true;

    while(true){
      if( tmp == 0 )
        break;

      for(int j = 0 ; j < W ; j++)
        if( j >= weight[i]*tmp )
          bag[j][1] = max( bag[j-weight[i]*tmp][0]
                         + value[i]*tmp
                         , bag[j][0] );
      for(int j = 0 ; j < W ; j++ )
        bag[j][0] = bag[j][1];

      if(!check)
        break;
      else if( limit[i] - tmp*2 + 1 <= tmp*2 ){
        tmp = limit[i] - tmp*2 + 1;
        check = false;
      }
      else
        tmp = tmp*2;

    }
  }
}
```

5 Sequence

5.1 RMQ

5.1.1 seg-tree

```
#define MAX_S 10000

 //  【注意】 1-index

int t[4*MAX_S+5];
int in[MAX_S+5];

#define LEFT(x) ((x)<<1)
#define RIGHT(x) (((x)<<1)+1)

// Build the segment tree
// parent, inputL, inputR
void buildSeg(int p, int inL, int inR)
{
    if(inL == inR)
    {
        t[p] = in[inL];
        return;
    }

    int mid = (inL+inR)/2;
    buildSeg(LEFT(p), inL, mid);
    buildSeg(RIGHT(p), mid+1, inR);
    t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
}


// Modify single point, and maintain the segment tree
// parent, left, right, idx, val
```

```
void modify(int p, int L, int R, int i, int x)
{
    // stop point
    if(i == L && L == R)
    {
        t[p] = x;
        return;
    }

    int mid = (L+R) / 2;
    if(i <= mid)
        modify(LEFT(p), L, mid, i, x);
    else
        modify(RIGHT(p), mid+1, R, i, x);
    // update this node
    t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
}

// Query in [quL, quR]
// parnet, left, right, queryL, queryR
int query(int p, int L, int R, int quL, int quR)
{
    // stop point
    if(quL <= L && R <= quR)
    {
        return t[p];
    }

    int mid = (L+R) / 2;
    if(quR <= mid) // left
        return query(LEFT(p), L, mid, quL, quR);
    else if(mid < quL) // right
        return query(RIGHT(p), mid+1, R, quL, quR);
    else // middle
        return max(query(LEFT(p), L, mid, quL, quR),
            query(RIGHT(p), mid+1, R, quL, quR));
}
```

5.1.2 sparse table

```
// Sparse Table (1-index)
int N = 14, logN = __lg(N), spI = logN+1;
int sp[spI][N] = {0};

void buildST()
{
    // first row (only one in a group)
    for(int i = 0; i < N; i++)
        sp[0][i] = value[i];
    // number of elements in a group = 2^i
    for(int i = 1; i < spI; i++)
    {   // j < N - (2^i - 1)
        for(int j = 0; j < N - ((1 << i) - 1); j++)
        {
            // Current row overlapped two upper
            // groups in (i-1) row
            sp[i][j] = max(sp[i-1][j], sp[i-1][j+(1 << (i-1))]);
        }
    }
}

// Query
int query(int l, int r)
{
    l--, r--;

    int distance = r - l + 1;
    int targetIdx = l != r ? __lg(distance)-1 : 0;

    return max(sp[targetIdx][l], sp[targetIdx][r - (1<<targetIdx - 1)]);
}
```

5.2 lcs

```
#define LEN 100

char s1[LEN];
char s2[LEN];
int length[LEN + 1][LEN + 1];
// 記錄每一格的的結果是從哪一格而來
int preve[LEN + 1][LEN + 1];
int lcs[LEN];

void print_LCS_s1(int i, int j)
{
    if (i == 0 || j == 0) return;

    if (preve[i][j] == 0)
    {
        print_LCS_s1(i-1, j-1);
        cout << s1[i];
    }
    else if (preve[i][j] == 1)
        print_LCS_s1(i, j-1);
    else if (preve[i][j] == 2)
        print_LCS_s1(i-1, j);
}

void print_LCS_s2(int i, int j)
{
    if (i == 0 || j == 0) return;

    if (preve[i][j] == 0)
    {
        print_LCS_s2(i-1, j-1);
        cout << s2[i];
    }
    else if (preve[i][j] == 1)
        print_LCS_s2(i, j-1);
    else if (preve[i][j] == 2)
        print_LCS_s2(i-1, j);
}

void LCS(int n1,int n2,char s1[],char s2[])
{

    for (int i = 0; i <= n1; i++)
        length[i][0] = 0;
    for (int j = 0; j <= n2; j++)
        length[0][j] = 0;

    if (n1 > n2)
    {
        for (int i = 1; i <= n1; i++)
        for (int j = 1; j <= n2; j++) // 兩層for
            if (s1[i] == s2[j])
            {
                length[i][j] = length[i-1][j-1] + 1;
                preve[i][j] = 0; // 左上方
            }
            else
            {
                if (length[i-1][j] < length[i][j-1])
                {
                    length[i][j] = length[i][j-1];
                    preve[i][j] = 1; // 左方
                }
                else
                {
                    length[i][j] = length[i-1][j];
                    preve[i][j] = 2; // 上方
                }
            }
        //
        cout << "LCS_len:" << length[n1][n2]  << '\n';
        cout << "LCS:";
        print_LCS_s1(n1, n2);
        //從right down開始
```

```
    }
    else
    {
        for (int i = 1; i <= n2; i++)
        for (int j = 1; j <= n1; j++) // 兩層for
            if ( s2[i] == s1[j] )
            {
                length[i][j] = length[i-1][j-1] + 1;
                preve[i][j] = 0; // 左上方
            }
            else
            {
                if (length[i-1][j] < length[i][j-1])
                {
                    length[i][j] = length[i][j-1];
                    preve[i][j] = 1; // 左方
                }
                else
                {
                    length[i][j] = length[i-1][j];
                    preve[i][j] = 2; // 上方
                }
            }
        //
        cout << "LCS_len:" << length[n2][n1]  << '\n';
        cout << "LCS:";
        print_LCS_s2(n2, n1);
    }
}
// 用法
LCS(s1_length,s2_length,s1,s2);
```

6 String

6.1 hash

```
size_t bkdr(const char str[]){
    // 31 131 1313 13131 131313 etc..
    size_t seed = 131;
    size_t hash = 0;

    while (*str){
        hash = hash * seed + (*str++);
    }

    return (hash & 0x7FFFFFFF);
}

// C++ build-in hash
hash<string> hash_gen;
size_t num = hash_gen(str);
```

6.2 lcs

```
// LCString
string LCStr(string &s1, string &s2)
{
  string rec;
  int arr[100][100];

  memset(arr, 0, sizeof(arr));

  int maxx = 0;
  int position = 0;

  for (int i = 1; i <= s1.length(); ++i)
  {
    for (int j = 1; j <= s2.length(); ++j)
    {
      if (s1[i-1] == s2[j-1])
      {
        arr[i][j] = arr[i-1][j-1] + 1;

        if (maxx < arr[i][j])
        {
          position = i;
          maxx = arr[i][j];
        }
      }
      else
      {
        arr[i][j] = 0;
      }
    }
  }
  for (int k = position - maxx; k < position; ++k)
  {
    rec += s1[k];
  }
  return rec;
}
```

7 Ad-hoc

7.1 n 皇后

```
int Queen[37000][14];
int Tmp[14];
int total=0;
int Row[14]={0},Left[27]={0},Right[27]={0};

void N_Queen(int k,int Number){
    int i,j;
    if(k==Number){
        for(j=0;j<Number;j=j+1){
            Queen[total][j]=Tmp[j];
        }
        total=total+1;
        return;
    }
    for(i=0;i<Number;i=i+1){
        int right= k+i;
        int left= k-i+Number-1;
        if( !Row[i] && !Left[left] && !Right[right] ){
            Row[i]=1;
            Left[left]=1;
            Right[right]=1;

            Tmp[k]=i;

            N_Queen(k+1,Number);

            Row[i]=0;
            Left[left]=0;
            Right[right]=0;

        }
    }
}

// 用法
N_Queen(0, num);
```