Contents

1 Basic

1.1 Disjoint Set

```cpp
#define N 100000
int p[N+5], siz[N+5];
void init()
{
    for(int i = 0; i <= N; i++)
    {
        p[i] = i;
        siz[i] = 1;
    }
}
int find(int x)
{
    return p[x]==x ? x : (p[x]=find(p[x]));
}
void uni(int a, int b)
{
    a = find(a), b = find(b);
    if(siz[a] > siz[b])
    {
        p[b] = p[a];
        siz[a] += siz[b];
        siz[b] = 0;
    }
    else
    {
        p[a] = p[b];
        siz[b] += siz[a];
        siz[a] = 0;
    }
}
```

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)
{
    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 prime

2.1.1 sieve

```cpp
#define N 100000
bool pr[N+5];
void buildPr()
{
    pr[0] = pr[1] = false;
    for(LL i = 2; i <= N; i++)
        pr[i] = true;

    for(LL i = 2; i <= N; i++)
```

```
        if(pr[i])
            for(LL a = i*i; a < N; a += i)
                pr[a] = false;
}
```

2.1.2 isPrime

```
// O(sqrt(n))
bool isPrime(int n)
{
    for(int i = 2; i <= sqrt(n); i++)
        if(n % i == 0)
            return false;
    return true;
}
```

2.1.3 genPrime

```
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=',')
```

2.2 combination

```
LL table[140000]; //start from 1 to 2^17-1
int digit[16]; //if digit add,table need add
// one and two cannott be 0
void build(int one,int two){
    int i,j,k=1;
    memset(digit,-1,sizeof(digit));
    memset(table,0,sizeof(table));

    while( k < 140000 ){
        i = 0;
        digit[i] += 1;
        while( digit[i] == 2 ){
            digit[i] = 0;
            i++;
            digit[i]++;
        }
        j=15;
        while( digit[j] < 0 ){
            j--;
        }
        while( j >= 0 ){
            table[k] = table[k] * 10;
            if( digit[j] == 1 )
                table[k] = table[k] + two;
            else if( digit[j] == 0 )
                table[k] = table[k] + one ;
            j--;
        }
        k=k+1;
    }
}
```

2.3 數列

2.3.1 Fibonacci

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

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

2.3.2 Tribonacci

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

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

3 Graph

3.1 最短路

3.1.1 dijkstra

```
/*
 * Dijkstra shortest path
 * Team: FJU_ElPsyCongroo
 */
#include <bits/stdc++.h>
using namespace std;

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

// ATTENTION: 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

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

```cpp
/*
 * SPFA (negative edge & negative cycle)
 * Team: FJU_ElPsyCongroo
 */
#include <bits/stdc++.h>

// ATTENTION: 0-index

using namespace std;

#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 pf 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)
{
    for(int i = 0; i < n; i++) d[i]=  INF; // init all d[]
    // 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

```cpp
/*
 * Floyd warshall algorithm implementation
 * Team: FJU_ElPsyCongroo
 */
#include <bits/stdc++.h>
using namespace std;

#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

```cpp
/*
 * Kruskal's algorithm implementation
 * Team: FJU_ElPsyCongroo
 */
#include <bits/stdc++.h>
using namespace std;

#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];

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

int kruskal()
{
    init();

    sort(e, e+m);

    int total = 0, i, j; // i -> cur vectex, j -> cur edge
    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

```
/*
 * Prim's algorithm implementation
 * Team: FJU_ElPsyCongroo
 */
#include <bits/stdc++.h>
using namespace std;

#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];

int n, m; // n個點, m個邊

int prim(int src)
{
    for(int i = 0; i < V; i++) d[i] = INF;
    // 放點的暫時權重
    priority_queue<P, vector<P>, greater<P> > pq; // (weight, idx)
    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;

        d[cur.S] = 0; // 更新該點的d[]，代表選了該點
        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)
            {
                if(wei < d[to]) // 看有沒有更小的權重
                {
                    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

```
/*
 * Backpack implmentation
 * Author: SunTalk
 * Team: FJU_ElPsyCongroo
 * ver 0.0.1
 */
#include <bits/stdc++.h>

#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++ ){
```

```
22
23          for(int j = 0 ; j < W ; j++ )
24              if( j >= weight[i] )
25                  bag[j][1] = max( bag[j][0] ,bag[j-
                        weight[i]][0] + value[i] );
26
27          for(int j = 0 ; j < W ; j++ )
28              bag[j][0] = bag[j][1];
29      }
30  }
31
32  //
        ------------------------------------------------
33  // 無限背包
34  void Unlimited(){
35
36      memset(bag,0,sizeof(bag));
37
38      for(int i = 0 ; i < N ; i++ ){
39
40          for(int j = 0 ; j < W ; j++ )
41              if( j >= weight[i] )
42                  bag[j][1] = max( bag[j][0] ,bag[j-
                        weight[i]][1] + value[i] );
43
44          for(int j = 0 ; j < W ; j++ )
45              bag[j][0] = bag[j][1];
46      }
47  }
48
49  //
        ------------------------------------------------
50  // 分組背包
51  int group;              //有幾組
52  int how_many;           //一組幾個
53  int WEIGHT,VALUE;
54
55  void Grouping(){
56
57      memset(bag,0,sizeof(bag));
58
59      for(int i = 0 ; i < group ; i++ ){
60
61          for(int j = 0 ; j < how_many ; j++ ){
62
63              scanf("%d %d",&WEIGHT,&VALUE);
64
65              for(int k = 0 ; k < W ; k++ ){
66
67                  if( j >= WEIGHT ){
68                      bag[j][1] = max( bag[j][1] , bag[j
                            ][0] );
69                      bag[j][1] = max( bag[j][1] ,bag[j-
                            WEIGHT][0] + VALUE );
70                  }
71              }
72          }
73
74          for(int j = 0 ; j < W ; j++ )
75              bag[j][0] = bag[j][1];
76      }
77  }
78
79  //
        ------------------------------------------------
80  // 多重背包
81  int limit[N];           //物品上限
82
83  void Multiple(){
84
85      for(int i = 0 ; i < N ; i++ ){
86
87          int tmp = 1;
88
89          while( tmp <= weight[i] ){
90
91              for(int j = 0 ; j < W ; j++)
92                  if( j >= weight[i]*tmp )
93                      bag[j][1] = max( bag[j-weight[i]*
                            tmp][0] + value[i]*tmp , bag[j
                            ][0] );
94
95              for(int j = 0 ; j < W ; j++ )
96                  bag[j][0] = bag[j][1];
97
98              weight[i] = weight[i]-tmp;
99              tmp = tmp*2;
100
101         }
102
103         if( weight[i] > 0 ){
104
105             for(int j = 0 ; j < W ; j++)
106                 if( j >= weight[i]*tmp )
107                     bag[j][1] = max( bag[j-weight[i]*
                            tmp][0] + value[i]*tmp , bag[j
                            ][0] );
108
109             for(int j = 0 ; j < W ; j++ )
110                 bag[j][0] = bag[j][1];
111
112         }
113
114     }
115 }
```

5 Sequence

5.1 RMQ

5.1.1 seg-tree

```
1  #define N 10000
2  // 1-index
3  int t[4*N+5];
4  int in[N+5];
5
6  #define LEFT(x) ((x)<<1)
7  #define RIGHT(x) (((x)<<1)+1)
8  // parent, left, right
9  void buildSeg(int p, int inL, int inR)
10 {
11     if(inL == inR) {
12         t[p] = in[inL];
13         return;
14     }
15     int mid = (inL+inR)/2;
16     buildSeg(LEFT(p), inL, mid);    // build left
           subtree
17     buildSeg(RIGHT(p), mid+1, inR); // build right
           subtree
18     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
19 }
20 // treeIdx, left, right, targetIdx, tragetVal
21 void modify(int p, int L, int R, int i, int x)
22 {
23     // stop point
24     if(i == L && L == R) {
25         t[p] = x;
26         return;
27     }
28     int mid = (L+R) / 2;
29     if(i <= mid)
30         modify(LEFT(p), L, mid, i, x);
31     else
32         modify(RIGHT(p), mid+1, R, i, x);
33     // update this node
34     t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
```

```
}
// treeIdx, left, right, queryleft, queryright
int query(int p, int L, int R, int quL, int quR)
{
    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))
                       ;
}
```

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

        }
    }
}

int main(int argc, char const *argv[]){
    int num;
    cin >> num;
    N_Queen(0,num);
    return 0;
}
```

6 String

6.1 hash

```
size_t BKDRHash(const char str[])
{
    size_t seed = 131; // 31 131 1313 13131 131313 etc
        ..
    size_t hash = 0;
    while (*str) {
        hash = hash * seed + (*str++);
    }
    return hash & 0x7FFFFFFF;
}

// c++ build-in hash
string in;
hash<string> hg;
num = hg(in);
```

7 Ad-hoc

7.1 n 皇后

```
#include <bits/stdc++.h>

using namespace std;

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);
```