

# Contents

1	Basic	
1.1	.vimrc	
1.2	Script	
1.2.1	new.sh	
1.2.2	cp.sh	
2	DS	
2.1	Disjoint Set	
2.2	int128	
3	Math	
3.1	prime	
3.1.1	sieve	
3.1.2	isPrime	
3.1.3	genPrime	
3.2	combination	
3.3	數列	
3.3.1	Fibonacci	
3.3.2	Tribonacci	
4	Graph	
4.1	convex hull	
4.2	最短路	
4.2.1	dijkstra	
4.2.2	SPFA	
4.2.3	floyd	
4.3	MST	
4.3.1	kruskal	
4.3.2	prim	
5	DP	
5.1	背包	
5.1.1	common	
5.1.2	01	
5.1.3	無限	
5.1.4	分組	
5.1.5	多重	
6	Sequence	
6.1	RMQ	
6.1.1	seg-tree	
7	String	
7.1	hash	
8	Ad-hoc	
8.1	n 皇后	

## 1 Basic

### 1.1 .vimrc

```

1|syntax on
2|color evening
3|set guifont=Consolas:h11
4|set nu ts=4 sw=4 sts=4 et ai si cin hls ru t_Co=256
5|set mouse=a bs=2 ci nocr ar fences=utf-8
6|set sm mat=0
7|filetype plugin indent on
8|inoremap {<CR> {<CR>}<Esc>O
9|nnoremap <C-Up> <Up>ddp<Up>
10|nnoremap <C-Down> ddp

```

### 1.2 Script

#### 1.2.1 new.sh

```

1|#!/bin/bash
2|clear
3|mkdir $1
4|pushd $1 > /dev/null
5|cat ../template.cpp \
6| | sed -e "s/{1}/$1/g; s/{2}/$2/g" > $1.cpp
7|touch $1.in $1.out
8|echo $1 CREATED
9|popd > /dev/null

```

#### 1.2.2 cp.sh

```

1|#!/bin/bash
2|clear
3|pushd $1 > /dev/null
4|echo compiling...
5|g++ -DDBG $1.cpp -o $1
6|if [[ "$?" == "0" ]]; then
7|    echo FINISHED
8|    ./$1
9|fi
10|popd > /dev/null

```

## 2 DS

### 2.1 Disjoint Set

```

21|
22|#define N 100000
23|int p[N+5], siz[N+5];
24|void init()
25|{
26|    for(int i = 0; i <= N; i++)
27|    {
28|        p[i] = i;
29|        siz[i] = 1;
30|    }
31|}
32|int find(int x)
33|{
34|    return p[x]==x ? x : (p[x]=find(p[x]));
35|}
36|void uni(int a, int b)
37|{
38|    a = find(a), b = find(b);
39|    if(siz[a] > siz[b])
40|    {
41|        p[b] = p[a];
42|        siz[a] += siz[b];
43|        siz[b] = 0;
44|    }
45|    else
46|    {
47|        p[a] = p[b];
48|        siz[b] += siz[a];
49|        siz[a] = 0;
50|    }
51|}

```

### 2.2 int128

```

1|static int print_int128(__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 __int28 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     char num[40], *now = num;
31     bool minus = false;
32     *n = 0; // reset n
33
34     int ret = scanf("%s", num);
35     if(ret == EOF) // scanf fails
36         return EOF;
37     // Judge if minus
38     if(*now == '-')
39     {
40         minus = true;
41         now++; // skip '-'
42     }
43     // Add the digit and multiply it by 10 one after
44     // another
45     while(*now)
46     {
47         *n += *now - '0';
48         now++;
49         if(*now) // check if now touches '\0'
50             *n *= 10;
51     }
52     *n = minus ? -(*n) : *n;
53
54     return 1;
55 }

```

## 3 Math

### 3.1 prime

#### 3.1.1 sieve

```

1 #define N 100000
2 bool pr[N+5];
3 void buildPr()
4 {
5     pr[0] = pr[1] = false;
6     for(LL i = 2; i <= N; i++)
7         pr[i] = true;
8
9     for(LL i = 2; i <= N; i++)
10         if(pr[i])
11             for(LL a = i*i; a < N; a += i)
12                 pr[a] = false;
13 }

```

#### 3.1.2 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 }

```

#### 3.1.3 genPrime

```

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=',')

```

## 3.2 combination

```

1 LL table[140000]; //start from 1 to 2^17-1
2 int digit[16]; //if digit add,table need add
3 // one and two cannott be 0
4 void build(int one,int two){
5     int i,j,k=1;
6     memset(digit,-1,sizeof(digit));
7     memset(table,0,sizeof(table));
8
9     while( k < 140000 ){
10         i = 0;
11         digit[i] += 1;
12         while( digit[i] == 2 ){
13             digit[i] = 0;
14             i++;
15             digit[i]++;
16         }
17         j=15;
18         while( digit[j] < 0 ){
19             j--;
20         }
21         while( j >= 0 ){
22             table[k] = table[k] * 10;
23             if( digit[j] == 1 )
24                 table[k] = table[k] + two;
25             else if( digit[j] == 0 )
26                 table[k] = table[k] + one ;
27             j--;
28         }
29         k=k+1;
30     }
31 }

```

## 3.3 數列

### 3.3.1 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...

### 3.3.2 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

## 4 Graph

### 4.1 convex hull

```

1 struct point
2 {
3     int x, y;
4
5     int dist(point b)
6     {
7         return (b.x - x) * (b.x - x) + (b.y - y) * (b.y - y);
8     }
9     int cross(point p2, point p3)
10    {
11        return (p2.x - x) * (p3.y - y) - (p2.y - y) * (p3.x - x);

```

```

12 }
13 }base;
14
15 //K[N] 第N個凸包
16 //all:全部的點
17 //checked:真正凸包上的點
18 struct polygon
19 {
20     vector<point> all;
21     vector<point> checked;
22 }K[MAXN];
23
24 //找出最左下角的點
25 bool min_y(point a, point b)
26 {
27     if (a.y == b.y)
28         return a.x < b.x;
29     return a.y < b.y;
30 }
31
32 //點逆時針排序
33 bool c_clockwise(point a, point b)
34 {
35     int c = base.cross(a, b);
36     return c > 0
37         || (c == 0 && base.dist(a) < base.dist(b));
38 }
39
40 //畫凸包
41 //base左下角的點
42 void convex_hull(int num)
43 {
44     //最左下角點開始
45     swap(K[num].all[0], *min_element(K[num].all.begin(),
46         K[num].all.end(), min_y));
47     base = K[num].all[0];
48     sort(K[num].all.begin()+1, K[num].all.end(),
49         c_clockwise);
50     K[num].all.PB(base);
51
52     //枚舉，把外積負的人淘汰
53     int m = 0;
54     for (int i = 0; i < K[num].all.size(); i++)
55     {
56         //stack的上面兩個與該點做嘗試
57         while (m >= 2
58             && K[num].checked[m-2].cross(K[num].checked[m-1], K[num].all[i]) <= 0)
59         {
60             K[num].checked.pop_back();
61             m--;
62         }
63         K[num].checked.PB(K[num].all[i]);
64         m++;
65     }
66
67     //確認點是否在凸包內
68     bool isinside(point pnt, int num)
69     {
70         for (int i = 1; i < K[num].checked.size(); i++)
71         {
72             if (K[num].checked[i-1].cross(K[num].checked[i],
73                 pnt) < 0)
74                 return false;
75         }
76         return true;
77     }
78
79     //計算面積
80     double area(int num)
81     {
82         double a = 0;
83         for (int i = 1; i < K[num].checked.size(); i++)
84             a += (K[num].checked[i-1].x * K[num].checked[i].y
85                 - (K[num].checked[i].x * K[num].checked[i-1].y)

```

```

83 ;
84 a += (K[num].checked[K[num].checked.size()-1].x * K[
85     num].checked[0].y) - (K[num].checked[0].x * K[num]
86     ].checked[K[num].checked.size()-1].y);
87
88 return a/2;
89 }

```

## 4.2 最短路

### 4.2.1 dijkstra

```

1 #define N 500
2 #define E 10000
3 #define INF 2e9
4 vector<P> e[E]; // (to, weight) edge
5 int d[N]; // 記錄起點到各個點的最短路徑長度
6 int n, m; // n 個點 m個邊
7
8 void dijkstra(int src)
9 {
10     for (int i = 0; i < n; i++) d[i] = INF;
11     // (dis, idx)
12     priority_queue<P, vector<P>, greater<P> > pq;
13     pq.emplace(0, src);
14
15     while(!pq.empty())
16     {
17         int a = -1, b = -1, wei;
18
19         tie(wei, a) = pq.top(); pq.pop();
20
21         if(d[a] != INF) continue;
22
23         d[a] = wei;
24         for(auto i : e[a])
25             if(d[i.first] == INF)
26                 pq.emplace(wei + i.second, i.first);
27     }
28 }

```

### 4.2.2 SPFA

```

1 #define N 505
2 #define E 20005
3 #define INF 2e9
4 vector<P> e[E]; // (to, weight) 邊
5 int d[N]; // 距離
6 int parent[N]; // parent idx pf a node
7 bool inq[N]; // inqueue
8 int cnt[N]; // path updating counter
9 bool del[N]; // 被刪掉的node，用於判斷
10 int n, m; // vertice, edges
11
12 void dfs(int src)
13 {
14     int to, wei;
15
16     del[src] = true;
17     for(auto i : e[src])
18     {
19         tie(to, wei) = i;
20         // 如果沒刪除的話
21         if(!del[to])
22             dfs(to);
23     }
24 }
25
26 void spfa(int src)
27 {
28     for(int i = 0; i < n; i++) d[i] = INF;
29     d[src] = 0;

```

```

30 parent[src] = src;
31 cnt[src] = 0;
32 queue<int> q;
33 q.push(src);
34
35 while(!q.empty())
36 {
37     int cur = q.front(), to, wei;
38     q.pop();
39     inq[cur] = false;
40
41     for(auto i : e[cur])
42     {
43         tie(to, wei) = i;
44         if(!del[to] && d[cur] + wei < d[to])
45         {
46             d[to] = d[cur] + wei;
47             parent[to] = cur;
48
49             if(!inq[to])
50             {
51                 cnt[to]++;
52
53                 if(cnt[to] >= n)
54                 {
55                     dfs(to);
56                     continue;
57                 }
58
59                 q.push(to);
60                 inq[to] = true;
61             }
62         }
63     }
64 }
65 // 無限小
66 for(int i = 0; i < n; i++)
67     if(del[i])
68         d[i] = -INF;
69 }
70
71 void findPath(int src, int end)
72 {
73     if(src != end)
74         findPath(src, parent[end]);
75     cout << end << ' ';
76 }

```

#### 4.2.3 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
13                 || d[i][j] == 0 ? -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     cout << end << ' ';
30 }

```

## 4.3 MST

### 4.3.1 kruskal

```

1 // Need disjoint set
2 #define V 50000
3 #define E 200000
4
5 struct edge
6 {
7     int fr, to, wei;
8     void setEdge(int f, int t, int w)
9     { fr = f; to = t; wei = w; }
10     friend bool operator<(edge &lhs, edge &rhs)
11     { return lhs.wei < rhs.wei; }
12 }e[E+5];
13 int n, m; // n vertice, m edges
14 int kruskal()
15 {
16     init(); // disjoint
17     sort(e, e+m);
18     // i -> cur vertex, j -> cur edge
19     int total = 0, i, j;
20     for(i = 0, j = 0; i < n-1 && j < m; i++, j++)
21     {
22         while(find(e[j].fr) == find(e[j].to))
23             j++;
24         uni(e[j].fr, e[j].to);
25         total += e[j].wei;
26     }
27     return i == n-1 ? total : -1;
28 }

```

### 4.3.2 prim

```

1 #define V 50000
2 #define E 200000
3 // e[from] => (to, weight)
4 vector<P> e[E+5];
5
6 // 無向圖
7 void addEdge(int from, int to, int weight)
8 {
9     e[from].emplace_back(to, weight);
10    e[to].emplace_back(from, weight);
11 }
12 int d[V+5];
13 int nt[V+5];
14 int n, m; // n個點, m個邊
15
16 int prim(int src)
17 {
18     for(int i = 0; i <= V; i++) d[i] = INF;
19     // 放點的暫時權重 (weight, idx)
20     priority_queue<P, vector<P>, greater<P>> pq;
21     int total = 0, v = 0, pre = -1; // 總和, 找到的點
22     pq.emplace(0, src);
23
24     int to, wei;
25     while(!pq.empty())
26     {
27         auto cur = pq.top(); pq.pop();
28         // 如果點cur已經當過top(不是第一次)
29         if(d[cur.S] != INF)
30             continue;
31         d[cur.S] = 0; // 更新該點的d[], 代表選了該點
32         nt[pre] = cur.S;
33         pre = cur.S;

```

```

34 total += cur.F;
35 v++;
36 // 遍歷所有跟點cur相連的邊
37 for(auto i : e[cur.S])
38 {
39     tie(to, wei) = i;
40     // 如果點to沒有選到過
41     if(d[to] == INF)
42     {
43         if(wei < d[to]) // 看有沒有更小的權重
44             pq.emplace(wei, to);
45         else
46             pq.emplace(d[to], to);
47     }
48 }
49 // 如果生成樹有n個點(全部找到)則輸出，否則輸出-1
50 return v == n ? total : -1;
51 }
52 }
53
54 void findPath(int src)
55 {
56     for(int i = 0; i < n; i++)
57         if(nt[i] != src)
58             cout << i << ' ' << nt[i] << '\n';
59 }

```

## 5 DP

### 5.1 背包

#### 5.1.1 common

```

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];

```

#### 5.1.2 01

```

1 // 0/1 背包
2 void ZeroOne(){
3     memset(bag,0,sizeof(bag));
4     for(int i = 0 ; i < N ; i++){
5         for(int j = 0 ; j < W ; j++){
6             if( j >= weight[i] )
7                 bag[j][1] = max( bag[j][0] , bag[j-weight[i]][0] +
8                                     value[i] );
9
10            for(int j = 0 ; j < W ; j++){
11                bag[j][0] = bag[j][1];
12            }
13        }
14    }
15 }

```

#### 5.1.3 無限

```

1 // 無限背包
2 void Unlimited(){
3     memset(bag,0,sizeof(bag));
4     for(int i = 0 ; i < N ; i++){
5         for(int j = 0 ; j < W ; j++){
6             if( j >= weight[i] )
7                 bag[j][1] = max( bag[j][0] , bag[j-weight[i]][1] +
8                                     value[i] );
9
10            for(int j = 0 ; j < W ; j++){
11                bag[j][0] = bag[j][1];
12            }
13        }
14    }
15 }

```

### 5.1.4 分組

```

1 // 分組背包
2 int group; // 有幾組
3 int how_many; // 一組幾個
4 int WEIGHT,VALUE;
5
6 void Grouping(){
7     memset(bag,0,sizeof(bag));
8     for(int i = 0 ; i < group ; i++){
9         for(int j = 0 ; j < how_many ; j++){
10             scanf("%d %d",&WEIGHT,&VALUE);
11
12             for(int k = 0 ; k < W ; k++){
13                 if( j >= WEIGHT ){
14                     bag[j][1] = max( bag[j][1] , bag[j][0] );
15                     bag[j][1] = max( bag[j][1] , bag[j-WEIGHT][0] +
16                                     VALUE );
17                 }
18             }
19         }
20     }
21     for(int j = 0 ; j < W ; j++){
22         bag[j][0] = bag[j][1];
23     }
24 }

```

### 5.1.5 多重

```

1 // 多重背包
2 int limit[N]; // 物品上限
3
4 void Multiple(){
5
6     for(int i = 0 ; i < N ; i++){
7         int tmp = 1;
8         while( tmp <= weight[i] ){
9             for(int j = 0 ; j < W ; j++){
10                 if( j >= weight[i]*tmp )
11                     bag[j][1] = max( bag[j-weight[i]*tmp][0] +
12                                     value[i]*tmp , bag[j][0] );
13
14                 for(int j = 0 ; j < W ; j++){
15                     bag[j][0] = bag[j][1];
16                 }
17
18                 weight[i] = weight[i]-tmp;
19                 tmp = tmp*2;
20             }
21             if( weight[i] > 0 ){
22                 for(int j = 0 ; j < W ; j++){
23                     if( j >= weight[i]*tmp )
24                         bag[j][1] = max( bag[j-weight[i]*tmp][0] +
25                                             value[i]*tmp , bag[j][0] );
26                     for(int j = 0 ; j < W ; j++){
27                         bag[j][0] = bag[j][1];
28                     }
29                 }
30             }
31         }
32     }
33 }

```

## 6 Sequence

### 6.1 RMQ

#### 6.1.1 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, targetVal
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)]);
35 }
36 // treeIdx, left, right, queryleft, queryright
37 int query(int p, int L, int R, int quL, int quR)
38 {
39     if(quL <= L && R <= quR) {
40         return t[p];
41     }
42     int mid = (L+R) / 2;
43     if(quR <= mid) // left
44         return query(LEFT(p), L, mid, quL, quR);
45     else if(mid < quL) // right
46         return query(RIGHT(p), mid+1, R, quL, quR);
47     else // middle
48         return max(query(LEFT(p), L, mid, quL, quR),
49                     query(RIGHT(p), mid+1, R, quL, quR));
50 }

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int Queen[37000][14];
6 int Tmp[14];
7 int total=0;
8 int Row[14]={0}, Left[27]={0}, Right[27]={0};
9
10 void N_Queen(int k, int Number){
11     int i, j;
12     if(k==Number){
13         for(j=0; j<Number; j=j+1){
14             Queen[total][j]=Tmp[j];
15         }
16         total=total+1;
17         return;
18     }
19     for(i=0; i<Number; i=i+1){
20         int right= k+i;
21         int left= k-i+Number-1;
22         if( !Row[i] && !Left[left] && !Right[right] ){
23             Row[i]=1;
24             Left[left]=1;
25             Right[right]=1;
26
27             Tmp[k]=i;
28
29             N_Queen(k+1, Number);
30
31             Row[i]=0;
32             Left[left]=0;
33             Right[right]=0;
34
35         }
36     }
37 }
38
39 int main(int argc, char const *argv[]){
40     int num;
41     cin >> num;
42     N_Queen(0, num);
43     return 0;
44 }

```

## 7 String

### 7.1 hash

```

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

```

## 8 Ad-hoc

### 8.1 n 皇后