Contents

1 Basic

1.1 .vimrc

```
syntax on
color evening
set guifont=Consolas:h11
set nu ts=4 sw=4 sts=4 et ai si cin hls ru t_Co=256
set mouse=a bs=2 ci nocp ar fencs=utf-8
set sm mat=0
filetype plugin indent on
inoremap {<CR>  {<CR>}<Esc>O
nnoremap <C-Up> <Up>ddp<Up>
nnoremap <C-Down> ddp
```

1.2 Script

1.2.1 new.sh

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

1.2.2 cp.sh

```
#!/bin/bash
clear
pushd $1 > /dev/null
echo compiling...
g++ -DDBG $1.cpp -o $1
if [[ "$?" == "0" ]]; then
    echo FINISHED
    ./$1
fi
popd > /dev/null
```

2 DS

2.1 Disjoint Set

```
#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;
    }
}
```

2.2 int128

```
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;
}
```

3 Math

3.1 prime

3.1.1 sieve

```
#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;
}
```

3.1.2 isPrime

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

3.1.3 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=',')
```

3.2 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;
    }
}
```

3.3 數列

3.3.1 Fibonacci

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

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

3.3.2 Tribonacci

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

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

4 Graph

4.1 convex hull

```
struct point
{
  int x, y;

  int dist(point b)
  {
    return (b.x - x) * (b.x - x) + (b.y - y) * (b.y - y);
  }
  int cross(point p2, point p3)
  {
    return (p2.x - x) * (p3.y - y) - (p2.y - y) * (p3.x - x);
```

```
  }
}base;

//K[N] 第N個凸包
//all:全部的點
//checked:真正凸包上的點
struct polygon
{
  vector<point> all;
  vector<point> checked;
}K[MAXN];

//找出最左下角的點
bool min_y(point a, point b)
{
  if (a.y == b.y)
    return a.x < b.x;
  return a.y < b.y;
}

//點逆時針排序
bool c_clockwise(point a, point b)
{
  int c = base.cross(a, b);
  return c > 0
    || (c == 0 && base.dist(a) < base.dist(b));
}

//畫凸包
//base左下角的點
void convex_hull(int num)
{
  //最左下角點開始
  swap(K[num].all[0], *min_element(K[num].all.begin(),
      K[num].all.end(), min_y));
  base = K[num].all[0];
  sort(K[num].all.begin()+1, K[num].all.end(),
      c_clockwise);
  K[num].all.PB(base);

  //枚舉，把外積負的人淘汰
  int m = 0;
  for (int i = 0; i < K[num].all.size(); i++)
  {
    //stack的上面兩個與該點做嘗試
    while (m >= 2
      && K[num].checked[m-2].cross(K[num].checked[m -
          1], K[num].all[i]) <= 0)
    {
      K[num].checked.pop_back();
      m--;
    }
    K[num].checked.PB(K[num].all[i]);
    m++;
  }
}

//確認點是否在凸包內
bool isinside(point pnt, int num)
{
  for (int i = 1; i < K[num].checked.size(); i++)
  {
    if (K[num].checked[i-1].cross(K[num].checked[i],
        pnt) < 0)
      return false;
  }
  return true;
}

//計算面積
double area(int num)
{
  double a = 0;
  for (int i = 1; i < K[num].checked.size(); i++)
    a += (K[num].checked[i-1].x * K[num].checked[i].y)
        - (K[num].checked[i].x * K[num].checked[i-1].y)
        ;

  a += (K[num].checked[K[num].checked.size()-1].x * K[
      num].checked[0].y) - (K[num].checked[0].x * K[num
      ].checked[K[num].checked.size()-1].y);

  return a/2;
}
```

4.2 最短路

4.2.1 dijkstra

```
#define N 500
#define E 10000
#define INF 2e9
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 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);

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

    tie(wei, a) = pq.top(); pq.pop();

    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);
  }
}
```

4.2.2 SPFA

```
#define N 505
#define E 20005
#define INF 2e9
vector<P> e[E]; // (to, weight) 邊
int d[N];       // 距離
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;
  d[src] = 0;
```

```
  parent[src] = src;
  cnt[src] = 0;
  queue<int> q;
  q.push(src);

  while(!q.empty())
  {
    int cur = q.front(), to, wei;
    q.pop();
    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 << ' ';
}
```

4.2.3 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 << ' ';
}
```

4.3 MST

4.3.1 kruskal

```
// Need disjoint set
#define V 50000
#define E 200000

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+5];
int n, m;  // n vertice, m edges
int kruskal()
{
  init(); // disjoint
  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++)
  {
    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;
}
```

4.3.2 prim

```
#define V 50000
#define E 200000
// e[from] => (to, weight)
vector<P> e[E+5];

// 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+5];
int nt[V+5];
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 (weight, idx)
  priority_queue<P, vector<P>, greater<P> > pq;
  int total = 0, v = 0, pre = -1; // 總和, 找到的點
  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] != src)
      cout << i << ' ' << nt[i] << '\n';
}
```

5 DP

5.1 背包

5.1.1 common

```
#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];
```

5.1.2 01

```
// 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];
  }
}
```

5.1.3 無限

```
// 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];
  }
}
```

5.1.4 分組

```
// 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];
  }
}
```

5.1.5 多重

```
// 多重背包
int limit[N];           //物品上限

void Multiple(){

  for(int i = 0 ; i < N ; i++ ){
    int tmp = 1;
    while( tmp <= weight[i] ){
      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];

      weight[i] = weight[i]-tmp;
      tmp = tmp*2;
    }
    if( weight[i] > 0 ){
      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];
    }
  }
}
```

6 Sequence

6.1 RMQ

6.1.1 seg-tree

```
#define N 10000
// 1-index
int t[4*N+5];
int in[N+5];
```

```

#define LEFT(x) ((x)<<1)
#define RIGHT(x) (((x)<<1)+1)
// parent, left, right
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);   // build left subtree
  buildSeg(RIGHT(p), mid+1, inR);// build right subtree
  t[p] = max(t[LEFT(p)], t[RIGHT(p)]);
}
// treeIdx, left, right, targetIdx, tragetVal
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)]);
}
// 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));
}
```

```
#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);

            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;
}
```

7 String

7.1 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);
```

8 Ad-hoc

8.1 n 皇后