Contents

1 Basic

1.1 BigNumber

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstdio>
#include <cstdlib>
#include <cstring>
using namespace std;
void scan(char s[100], int a[100])
{
    int i = 100 - 1;            // 大數的數字位置
    int j = 0, n = strlen(s);   // 字串的字元位置
    while (i >= n) a[i--] = 0;  // 開頭一律填零
    while (i >= 0) a[i--] = s[j++] - '0';    // 字串頭尾顛倒，存入陣列
}
void print(int a[100])
{
    int i = 100 - 1;
    while (i >= 0 && a[i] == 0) i--;

    if (i < 0)
        cout << '0';
    else
        while (i >= 0) cout << a[i--];
}
bool largerthan(int a[100], int b[100])
{
    // 從高位數開始比，對應的位數相比較。
    for (int i=100-1; i>=0; i--)
        if (a[i] != b[i])   // 一旦ab不一樣大，馬上回傳結果。
            return a[i] > b[i];
    return false;   // 完全相等
}
void add(int a[100], int b[100], int c[100])
{
    for (int i=0; i<100; i++)   // 對應的位數相加
        c[i] = a[i] + b[i];

    for (int i=0; i<100-1; i++) // 一口氣進位
    {
        c[i+1] += c[i] / 10;    // 進位
        c[i] %= 10;             // 進位後餘下的數
    }
}
void sub(int a[100], int b[100], int c[100])
{
    for (int i=0; i<100; i++)
        c[i] = a[i] - b[i];

    for (int i=0; i<100-1; i++) // 一口氣借位和補位
        if (c[i] < 0)
        {
            c[i+1]--;           // 借位
            c[i] += 10;         // 補位
        }
}
void mul(int a[100], int b[100], int c[100])
{
    for (int i=0; i<100; i++)
        c[i] = 0;

    for (int i=0; i<100; i++)
        for (int j=0; j<100; j++)
            if (i+j < 100)
                c[i+j] += a[i] * b[j];

    for (int i=0; i<100-1; i++) // 一口氣進位
    {
        c[i+1] += c[i] / 10;
        c[i] %= 10;
    }
}
void mul(int a[100], int b, int c[100])
{
    for (int i=0; i<100; i++)
        c[i] = a[i] * b;

    for (int i=0; i<100-1; i++) // 一口氣進位
    {
        c[i+1] += c[i] / 10;
        c[i] %= 10;
    }
}
void div(int a[100], int b[100], int c[100])
{
    int t[100];

    for (int i=100-1; i>=0; i--)
        for (int k=9; k>0; k--) // 嘗試商數
        {
            mul(b+i, k, t);
            if (largerthan(a+i, t))
            {
                sub(a+i, t, c+i);
                break;
            }
        }
}
void div(int a[100], int b, int c[100])
{
    int r = 0;
    for (int i=100-1; i>=0; i--)
    {
        r = r * 10 + a[i];
        c[i] = r / b;
        r %= b;
    }
}
int main(){
    char Str[100];
    int BigNum[100];
    cin >> Str;
    scan(Str,BigNum);
    print(BigNum);
    return 0;
}
```

1.2 BigNumber+ 小數轉分數

```cpp
//2017 NCPU Problem.1
#include<bits/stdc++.h>
using namespace std;
long long gcd(long long a, long long b)
```

```cpp
{
    while(b)
      b ^= a ^= b ^= a %= b;

    return a;
}
int main(int argc, char const *argv[])
{
    double L;
    while(~scanf("%lf",&L) && L){
        long long rp = L,rq = 1,tmp;

        while(L != (long long)L){
            L *= 10;
            rp = (long long)L;
            rq *= 10;
        }
        tmp = gcd(rp,rq);
        rp /= tmp;
        rq /= tmp;
        printf("%lld/%lld\n",rp,rq);
    }

    return 0;
}
// unsigned   int   0～4294967295
// int   2147483648～2147483647
// unsigned long 0～4294967295
// long   2147483648～2147483647
// long long的最大值：9223372036854775807
// long long的最小值：-9223372036854775808
// unsigned long long的最大值：18446744073709551615
// float的最大值和最小值分别为3.40282e+038（10的38次方），1.17549e-038（10的38次方）
// double的最大值和最小值分别为1.79769e+308（10的308次方），2.22507e-308（10的308次方）

//INPUT 3 3.14 1.234567890123456 0
//OUTPUT 3/1 157/50 19290123283179/15625000000000
```

1.3 Gcd

```cpp
int gcd(int a, int b)
{
    while(b)
      b ^= a ^= b ^= a %= b;

    return a;
}
// gcd(a,b) □ lcm(a,b) = ab
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
```

1.4 Prime

```cpp
#define N 100000
bool pr[N+5];
long long int i , a;
void buildPr()
{

    pr[0] = pr[1] = false;
    for( i = 2; i <= N; i++)
        pr[i] = true;

    for( i = 2; i <= N; i++)
        if(pr[i])
            for(  a = i*i; a < N; a += i)
                pr[a] = false;
}
```

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

2 DP

2.1 LCS

```cpp
//DP
//LCS 最長子字串

#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#define strMAX

using namespace std;

char str1[strMAX];
char str2[strMAX];

scanf("%s", str1);
scanf("%s", str2);

int len1 = strlen(str1);
int len2 = strlen(str2);

int dp[len1+1][len2+2];
//dp包含空字串

memset(dp, 0, sizeof(dp));

for(int i = 1 ; i <= len1 ; i++)
{
    for(int j = 1 ; j < len2 ; j++)
    {
        if(str1[i-1] == str2[j-1])
            dp[i][j] = dp[i-1][j-1] + 1;
        else
            dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
    }
}

int ans = dp[str1][str2];
```

2.2 LIS(輸出幾個)

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main(){
    int N;
    while(cin>>N){
        vector<int>v;
        int num;
        for(int i=0;i<n;i++){
            cin >> num;
            if(!v.size() || num>v.back())
                v.push_back(num);
            else
                *lower_bound(v.begin(), v.end(),num) =
                    num;
        }
        cout << v.size() << endl;
    }
}
```

2.3 LIS(輸出序列)

```
//Longest Increasing Subsequence (LIS)

#include <iostream>
#include <algorithm>
using namespace std;

int Count; //幾個數字
int seq[5];
int length[5];
int pv[5]; // prev[x] 記錄 s[x] 是接在哪個數字後面

void trace(int i)
{
    if (pv[i] != -1) {
        trace(pv[i]);
    }
    cout << seq[i] << ' ';
}

void LIS()
{
    for (int i=0; i< Count; i++) length[i] = 1;

    for (int i=0; i< Count; i++) pv[i] = -1;     // -1
        代表 s[i] 是開頭數字，沒有接在其他數字後面。

    for (int i=0; i< Count; i++)
        for (int j=i+1; j< Count; j++)
            if (seq[i] < seq[j])
                if (length[i] + 1 > length[j])
                {
                    length[j] = length[i] + 1;
                    pv[j] = i;
                }

    int n = 0, pos = 0;
    for (int i=0; i< Count; i++){
        if (length[i] > n)
        {
            n = length[i];
            pos = i;
        }
    }
    trace(pos); // 印出一個LIS
}



int main(int argc, char const *argv[])
{

    cin >> Count;
    for(int i = 0 ; i <  Count ; i++){
        cin >> seq[i];
    }
    LIS();
    return 0;
}
```

3 Graph

3.1 DfsTemplate

```
void dfs_visit  (int i , int len){
  for(int j = 0 ; j < n ; j++){
      if(a[i][j]){
          if(a[i][j]){
            a[i][j] = a[j][i] = 0;
            dfs_visit(j,len+1);
            a[i][j] = a[j][i] = 1;
          }
  }
  if(len > best) best = len;
}
void dfs(){
  int i;
  best = 0 ;
  for(int i = 0 ; i < n ; i++){
    dfs_visit(i,0);
  }
  cout << best << endl;
}
```

3.2 BfsTemplate

```
void bfs(int s)
{
    queue <int> que;
    que.push(s);
    vis[s] = 1;

    while(!que.empty())
    {
        int tmp = que.front();
        que.pop();

        for(int i : G[s])
        {
            if(!vis[i])
            {
                que.push(i);
                vis[i] = 1;
            }
        }
    }
}
```

3.3 Dijkstra

```
//dijkstra
#include <iostream>
#include <cstdio>
#include <queue>
#include <vector>
using namespace std;

#define maxn 51415

struct Edge
{
    int from, to, dist;

    Edge(int _from, int _to, int _dist)
    {
        from = _from;
        to = _to;
        dist = _dist;
    }
};

struct Item
{
    int node;
    int dist;

    Item(int _node, int _dist)
    {
        node = _node;
        dist = _dist;
    }

    bool operator <(const Item& rs) const
    {
        return dist > rs.dist;
    }
};

int main(void)
{
    int n, m;
    while(~scanf("%d %d", &n, &m))
    {
        vector <Edge> edges;
        vector<int> G[maxn];
        priority_queue <Item> dij;
        int visit[maxn] = {-1};

        for(int i = 0; i < m; i++)
        {
            int a, b, c;
            scanf("%d %d %d", &a, &b, &c);
            edges.push_back(Edge(a, b, c));
            G[a].push_back(i);
        }

        int node = 1;
        dij.push(Item(1, 0));
        Item hold = Item(0, 0);
        while(!dij.empty())
        {
            hold = dij.top();
            dij.pop();

            if(visit[hold.node] == 1)
                continue;

            visit[hold.node] = 1;

            node = hold.node;
            if(node == n)
            {
                break;
            }

            for(int i = 0; i < G[node].size(); i++)
            {
               dij.push(Item(edges[G[node][i]].to, hold
                    .dist+edges[G[node][i]].dist));
            }
        }

        if(node != n) printf("-1\n");
        if(node == n)  printf("%d\n", hold.dist);
    }
    return 0;
}
```

3.4 Kruskal

```
//依題目：eMAX、nMAX

#define eMAX 205
#define nMAX 105
#define INF 1000000000

//n 點數量，e 邊數量，ise 紀錄幾號邊使用過
int n, e, ise[eMAX];
int father[nMAX];

struct Edge
{
    int from, to, dist;
}edge[eMAX];

bool cmp(Edge a, Edge b)
{
    return a.dist < b.dist;
}

//find：找尋是否為同個set
int find(int i)
{
    if(father[i] == i)  return i;
    return find(father[i]);
}

int Kruskal()
{
    int ans = 0, num = 0, id = 0;
    for(int i = 1 ; i <= n ; i++)
    {
        father[i] = i;
    }
    for(int i = 0 ; i < e ; i++)
    {
        int n1 = edge[i].from;
        int n2 = edge[i].to;
        int n1f = find(n1);
        int n2f = find(n2);
        if(n1f != n2f)
        {
            ise[id] = i;
            id++;
            father[n1f] = n2f;
            ans = ans + edge[i].dist;
        }
    }
    for(int i = 1 ; i <= n ; i++)
    {
        if(i == find(i))
        {
            num++;
        }
    }
    if(num > 1)
    {
        return INF; //no solution
    }
    else
    {
        return ans; //min spanning tree cost
    }
}
```

4 Tree

4.1 SegmentTree(完整)

```
#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))
                       ;
}
```

4.2 SegmentTree(建立)

```


typedef vector <int> vi;
#define INF 1e18

vi ST;
vi A;

void st_build(vi &ST, const vi &A, int vertex, int L,
     int R)
{
    if(L == R)  ST[vertex] = L;
    else
    {
        int nL = 2*vertex, nR = 2*vertex+1;

        st_build(ST, A, nL, L, (L+R)/2);
        st_build(ST, A, nR, (L+R)/2+1, R);

        int lContent = ST[nL], rContent = ST[nR];
        int lValue = A[lContent], rValue = A[rContent];
        ST[vertex] = (lValue <= rValue) ? lContent :
            rContent;
    }
}

void st_create(vi &ST, const vi &A)
{
    int len = (int) 4*A.size();
    ST.assign(len, 0);
    st_build(ST, A, 1, 0, (int)A.size() - 1);
}
```