Contents

1 Section1

1.1 basic

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long

int main() {

    cout<<"for define \n";
    return 0;
}
```

2 Section2 Math

2.1 GCD

```cpp
#include<iostream>
using namespace std;
int GCD(int x,int y){
    while(y != 0){
        return GCD(y,x%y);
    }
    return x;
}

int main(){
    int a,b;
    cin>>a>>b;
    int gcd = GCD(a,b);
    int lcm = a*b/gcd;

    cout<<"最大公因數為: "<<gcd<<'\n';
    cout<<"最小公倍數為: "<<lcm<<'\n';
    return 0;
}
```

2.2 Perfect Squares

```cpp
#include<iostream>
#include<cmath>
using namespace std;

// 計算是否為完全平方數
bool isPerfectSquare(int x) {
    int y = sqrt(x);
    return y*y == x;
}

// using Lagrange's four-square theorem
bool checkAnswer4(int n) {
    while (n%4 == 0) {
        n /= 4;
    }
    return n%8 == 7;
}

// 計算數字是否可轉換成
// 平方數們的相加
// 並取最少個數
// e.g.: 13 = 4 + 9
int main(){
    int num;
    bool isTwo = false;
    cin >> num;
    if (isPerfectSquare(num)) {
        printf("1\n");
    }
    else if (checkAnswer4(num)) {
        printf("4\n");
    }
    for (int i=0; i*i<=num ;i++) {
        int j = num-i*i;
        if (isPerfectSquare(j)) {
            printf("2\n");
            isTwo = true;
            break;
        }
    }
    if (isTwo == false) {
        printf("3\n");
    }
    return 0;
}
```

3 Section3 String

3.1 string

```cpp
#include<iostream>
#include<string>
using namespace std;

int main(){
//初始化字串
    string s1 = "",s2 = "";
    long long a;
    int b;

//吃整行(含空格)
    getline(cin,s1);

//compare,assign,串接
    s1 == s2;
    s1 = s2;
    s1 += s2[i];

//字串切割,i:起始位置,len:幾個
    s1 = s1.substr(i,len);

//轉成數字或數字轉字串
    s1 = to_string(a);
    s2 = to_string(b);
    a = stoll(s1);
    b = stoi(s2);

//判斷數字,字母
    isdigit(s1[i]);
    isalpha(s2[i]);
    return 0;
}
```

4 Section4 tools

4.1 permutation

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

int main(){
    string a = "abc";
    string b = "cba";
//一定要先排序，才會有全部的組合
    sort(a.begin(),a.end());

//產生組合的迴圈
    do{
        cout<<a<<"\n";
    }while(next_permutation(a.begin(),a.end()));

//檢查b字串是否為a字串可排出結果
    bool isSamePer =
        is_permutation(a.begin(),a.end(),b.begin());

//產生上一個排列結果
    prev_permutation(a.begin(),a.end());

    return 0;
}
```

4.2 高斯消元

```
#define maxn 500+5
int A[maxn][maxn];
int guassian_elimination(int m, int n){
  int r,i,j,k,u;
  i=j=0;
  while(i<m && j<n){
    r=i;
    for(k=i; k<m; k++){//找為1的值
      if(A[k][j]){
        r=k;
        break;
      }
    }
    if(A[r][j]){
      if(r!=i){//換到first row
        for(k=0; k<n; k++)
          swap(A[r][k],A[i][k]);
      }
      for(u=i+1; u<m; u++){
//需要減時，該row才減第一個row
        if(A[u][j]){
          for(k=0; k<n; k++)
            A[u][k]^=A[i][k];
        }
      }
      i++;
    }
    j++;
  }
  return n-i;// free variable數量
}
```

4.3 最大流

```
#define N 105
int path[N],adj[N][N];

memset(adj,0,sizeof(adj));
//建雙向邊
for(int i=0,u,v,w; i<c; i++){
    scanf("%d %d %d",&u,&v,&w);
    adj[u][v] += w;
    adj[v][u] += w;
}

int flow = 0;
while(true){
    memset(path,0,sizeof(path));
    queue<int> Q;

    path[s] = s;
    Q.push(s);
//BFS找路徑
    while(!Q.empty() && !path[t]){
        int now = Q.front();
        Q.pop();
        for(int i=1; i<=n; i++){
            if(!path[i] && adj[now][i]>0){
                Q.push(i);
                path[i] = now;
            }
        }
    }
//完全沒有路到t就break
    if(!path[t])
        break;
    int min_flow = 1e9;
//找最窄的路
    for(int from=path[t],to=t; from!=to;
         from=path[to=from]){
        min_flow = min(min_flow,adj[from][to]);
    }
//更新該路徑所有邊的額度
    for(int from=path[t],to=t; from!=to;
         from=path[to=from]){
        adj[from][to] -= min_flow;
        adj[to][from] += min_flow;
    }
    flow += min_flow;
}
```

5 Section5 Graph

5.1 kruskal

```
#define maxn 200005
#define MP make_pair
int N,M;
int par[maxn],Rank[maxn];
vector<pair<int,int>> G[maxn*2];//雙向邊，所以X2

struct edge{
    int x,y,w;
    bool operator<(const edge& rhs) const{
        return w<rhs.w;
    }
}e[maxn*2];//雙向邊，所以X2

int Find(int a){
    return par[a]==a?a:(par[a] = Find(par[a]));
}

bool Union(int a, int b){
    a = Find(a);
    b = Find(b);
    if(a==b) return false;
    int tmp = Rank[a] + Rank[b];
    if(Rank[a]>=Rank[b]){
        Rank[a] = tmp;
        par[b] = a;
    }
    else{
        par[a] = b;
        Rank[b] = tmp;
    }
    return true;
}

int kruskal(){
    for(int i=0; i<N; i++){
        G[i].clear();
        par[i] = i;
        Rank[i] = 1;
    }
    int m = 0, tot = 0;
    for(int i=0,u,v,w; i<M; i++){
        scanf("%d %d %d",&u,&v,&w);
        e[m++] = edge{u,v,w};
        e[m++] = edge{v,u,w};
        tot += w;
    }
    sort(e,e+m);

    int mst = 0, cost = 0;
    for(int i=0,u,v,w; i<m; i++){
        u = e[i].x;
        v = e[i].y;
        w = e[i].w;
        if(Union(u,v)){
            cost += w;
            mst++;
            G[u].push_back(MP(v,w));
            G[v].push_back(MP(u,w));
        }
        if(mst==N-1)
            break;
    }
    return cost;
}
```

5.2 floyd

```
//N為點的個數，G為記錄路徑長的二維振烈
for(int k=0; k<N; k++){
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++){
            G[i][j]=min(G[i][j],G[i][k]+G[k][j]);
        }
    }
}
```

5.3 Dijkstra

```
struct Data{
    int u,w;
    bool operator<(const Data&rhs) const
    {
        return w>rhs.w;
    }
};

void sol(int s){
    memset(d,0x3f,sizeof(d));
    memset(vis,0,sizeof(vis));
    d[s] = 0;
    priority_queue<Data> pq;
    pq.push(Data{s,0});

    while(!pq.empty()){
        Data k = pq.top();
        pq.pop();
        int u = k.u;
        if(vis[u]) continue;
        vis[u] = 1;

        for(int i=0; i<G[u].size(); i++){
            int v = G[u][i].first, w = G[u][i].second;
            if(d[v]>d[u]+w){
                d[v] = d[u] + w;
                pq.push(Data{v,d[v]});
            }
        }
    }
}
```

5.4 SPFA

```
#define N 1005
#define MP make_pair
typedef pair<int,int> PII;

int n,m;
int dis[N],cnt[N];
vector<PII> G[N];
bool inq[N];

bool SPFA(){
    memset(dis,0x3f,sizeof(dis));
    memset(inq,false,sizeof(inq));
    memset(cnt,0,sizeof(cnt));

    queue<int> Q;
    dis[0] = 0;
    Q.push(0);
    inq[0] = true;
    while(!Q.empty()){
        int u = Q.front();
        Q.pop();
        inq[u] = false;
        for(int i=0; i<G[u].size(); i++){
            int v = G[u][i].first, w = G[u][i].second;
            if(dis[v]>dis[u]+w){
                dis[v] = dis[u] + w;
                if(!inq[v]){
//如果鬆弛超過n次，代表有負環
                    if(++cnt[v]>=n)
                        return true;
                    inq[v] = true;
                    Q.push(v);
                }
            }
        }
    }
    return false;
}
```

6 Java

6.1 java biginterger

```
import java.io.*;
import java.util.*;
import java.math.BigInteger;

```

```java
public class z {
    public static void main(String args[]) {
        Scanner cin = new Scanner(System.in);
//Java大數資料型態: BigInteger

        BigInteger num = BigInteger.valueOf(1);
        BigInteger btwo = new BigInteger("2");
        while (cin.hasNext()){
            BigInteger a = BigInteger.valueOf(0);
            BigInteger b = BigInteger.valueOf(0);

            //讀入一整行字串
            String str = cin.next();

            //-1停止輸入
            if (str.equals("-1")) break;

            num = new BigInteger(str);
            //a += num
            a = a.add(num);

            //b -= num
            b = b.subtract(num);

        System.out.print("a+num is " + a + "\n");
        System.out.print("b-num is " + b + "\n");
//乘2
        System.out.printf("%s*2 = %s\n", num,
            num.multiply(btwo));
//除2
        System.out.printf("%s/2 = %s\n\n", num,
            num.divide(btwo));
        }
//2的100次方
        System.out.printf("2^100 = %s\n",
            btwo.pow(100));
    }
}
```

7 數學公式

7.1 thm

- 中文測試
- $\sum\limits_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$