1194 zipper
13295 最佳加法表达式
1947 拦截导弹

****1194.cpp:

```
//By Guo Wei
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
char a[300],b[300],c[700];
int La,Lb,Lc;
char valid[210][210][410];
bool Valid(int as,int bs,int cs)
{
    if( cs == Lc  ) {
        if( as == La && bs == Lb)
            return true;
        else
            return false;
    }
    if( valid[as][bs][cs] != -1 )
        return valid[as][bs][cs];
    bool b1 = false,b2 = false;
    if( a[as] == c[cs])
        b1 = Valid(as+1,bs,cs+1);
    if( b[bs] == c[cs])
        b2 = Valid(as,bs+1,cs+1);
    valid[as][bs][cs] =     b1 || b2;
    return b1 || b2;

}
int main()
{
    int n;
    scanf("%d",&n);
    for(int i = 0;i < n; ++i) {
        scanf("%s%s%s",a,b,c);
        La = strlen(a);
        Lb = strlen(b);
        Lc = strlen(c);
        memset(valid,0xff,sizeof(valid));
        if(Valid(0,0,0))
```

```
            printf("Data set %d: yes\n",i+1);
        else
            printf("Data set %d: no\n",i+1);
    }
    return 0;
}
```

****13295.cpp:

```
//By Guo Wei
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
struct BigInt
{

    int num[110];
    int len;
    BigInt operator+(const BigInt & n) { //重载+，使得 a + b 在 a,b 都是 BigInt 变量的时候能成立
        int ml = max(len,n.len);
        int carry = 0; //进位
        BigInt result;
        for(int i = 0;i < ml; ++i) {
            result.num[i] = num[i] + n.num[i] + carry;
            if( result.num[i] >= 10)    {
                carry = 1;
                result.num[i] -= 10;
            }
            else
                carry = 0;
        }
        if ( carry == 1) {
            result.len = ml + 1;
            result.num[ml] = 1;
        }
        else
            result.len = ml;
        return result;
    }
    bool operator<(const BigInt & n) {
        if( len > n.len )
```

```
            return false;
        else if( len < n.len)
            return true;
        else {
            for(int i = len -1; i >= 0; -- i) {
                if( num[i] < n.num[i])
                    return true;
                else if( num[i] > n.num[i])
                    return false;
            }
            return false;
        }
    }
    BigInt() {
        len = 1;
        memset(num,0,sizeof(num));
    }
    BigInt(const char * n,int L) { //由长度为 L 的 char 数组构造大整数。n 里面的元素取值范围从 1-9。
        memset(num,0,sizeof(num));
        len = L;
        for(int i = 0; n[i]; ++i)
            num[len-i-1] = n[i] - '0';
    }
};
ostream & operator <<(ostream & o,const BigInt & n)
{

    for(int i = n.len - 1;i >= 0; --i)
        o << n.num[i];
    return o;
}
const int MAXN  = 60;
char a[MAXN];
BigInt Num[MAXN][MAXN];//Num[i][j]表示从第 i 个数字到第 j 个数字所构成的整数
BigInt V[MAXN][MAXN]; //V[i][j]表示 i 个加号放到前 j 个数字中间，所能得到的最佳表达式的值。
int main()
{
    int m,n;
    BigInt inf; //无穷大
    inf.num[MAXN-2] = 1;
    inf.len = MAXN-1;
```

```
    while(cin >> m ) {
        cin >> a+1;
        n = strlen(a+1);
        for(int i = 1;i <= n; ++i)
            for(int j = i;j<= n; ++j) {
                Num[i][j] = BigInt(a+i,j-i+1);
            }
        for(int j = 1; j <= n; ++j)  {
            V[0][j] = BigInt(a+1,j);
        }

        for(int i = 1;i <= m; ++i) {
            for(int j = 1; j <= n; ++j) {
                if( j - 1 < i)
                    V[i][j] = inf;
                else {
                    BigInt tmpMin = inf;
                    for(int k = i; k < j; ++k) {
                        BigInt tmp = V[i-1][k] + Num[k+1][j];
                        if (tmp < tmpMin)
                            tmpMin = tmp;
                    }
                    V[i][j] = tmpMin;
                }
            }
        }
        cout << V[m][n] << endl;
    }
    return 0;
}
```

****1947.cpp:

```
// By Guo Wei
#include <iostream>
#include <algorithm>
using namespace std;
int k;
int a[30];
int len[30]; // len[i]是以a[i]为终点的最长不上升子序列的长度
int main()
{
```

```
    cin >> k;
    for(int i = 0;i < k; ++i) {
        cin >> a[i];
        len[i] = 1;
    }
    for(int i = 1;i < k; ++i) {
        for(int j = 0; j < i; ++j ) {
            if( a[j] >= a[i])
                len[i] = max(len[i],len[j] + 1);
        }
    }
    cout << * max_element(len,len + k) << endl;
    return 0;
}
```