面向对象与STL

人生苦短，我要面向对象。

~~Warning:找不到对象~~

~~STL容器的真正含义：啊！是大佬（方言ShiTaLao□□我死了~~

~~同理，数据库语言SQL的真实含义：啊！是巨佬（方言ShiQuLao□□我死了~~

vector

最简单的STL是vector□简单到不能再简单了。

其中vector的insert功能很好地模拟了链表的插入操作，虽然vector的本质是数组。

（但是写一个数组的插入岂不是很繁？人生苦短啊）

例如：

OJ编号263□jhljx又来了。

```
#include<stdio.h>
#include<string.h>

#include<algorithm>
#include<vector>

using namespace std;

char sorted;
char Add[4]="Add";
char Del[4]="Del";
char Sum[4]="Sum";
char op[5];
int tmpint;
int n;
long long result;

int main()
{
    while(~scanf("%d",&n))
    {
        vector<int> a;
        sorted=1;
        while(n--)
        {
            scanf("%s",op);
```

```
            if(!strcmp(op,Add))
            {
                scanf("%d",&tmpint);
                a.push_back(tmpint);
                sorted=0;
            }
            else if(!strcmp(op,Del))
            {
                if(!sorted)
                {
                    sort(a.begin(),a.end());
                    sorted=1;
                }
                scanf("%d",&tmpint);
                vector<int>::iterator it;
                for(it=a.begin();it!=a.end();++it)
                {
                    if(*it==tmpint)
                    {
                        a.erase(it);
                        break;
                    }
                }
            }
            else if(!strcmp(op,Sum))
            {
                result=0;
                if(!sorted)
                {
                    sort(a.begin(),a.end());
                    sorted=1;
                }
                int i;
                for(i=2;i<a.size();i+=5)
                {
                    result+=(long long)a[i];
                }
                printf("%lld\n",result);
            }
        }
    }
}
```

OJ编号266□AZY学习顺序表。

```
#include<stdio.h>
#include<string.h>
```

```
#include<vector>

using namespace std;

int n,m;
int tmpint;
char Insert[5]="Ins";
char Delete[5]="Del";
char Locate[5]="Loc";
char Getcha[5]="Get";
char operation[5];
int opA,opB;

int main()
{
    while(~scanf("%d%d",&n,&m))
    {
        vector<int> a;
        int i;
        for(i=0;i<n;++i)
        {
            scanf("%d",&tmpint);
            a.push_back(tmpint);
        }
        while(m--)
        {
            scanf("%s",operation);
            if(!strcmp(operation,Insert))
            {
                scanf("%d%d",&opA,&opB);
                if(opA>(a.size()+1)||opA<=0)
                {
                    printf("Wrong input!\n");
                }
                else
                {
                    a.insert(a.begin()+opA-1,opB);
                    vector<int>::iterator ni;
                    for(ni=a.begin();ni!=a.end();ni++)
                    {
                        printf("%d ",*ni);
                    }
                    printf("\n");
                }
            }
            else if(!strcmp(operation,Delete))
            {
                scanf("%d",&opA);
                char mark=0;
                vector<int>::iterator it;
                for(it=a.begin();it!=a.end();++it)
```

```
            {
                if(*it==opA)
                {
                    mark=1;
                    a.erase(it);
                    vector<int>::iterator ni;
                    for(ni=a.begin();ni!=a.end();ni++)
                    {
                        printf("%d ",*ni);
                    }
                    printf("\n");
                    break;
                }
            }
            if(!mark)
            {
                printf("Wrong input!\n");
            }
        }
        else if(!strcmp(operation,Locate))
        {
            scanf("%d",&opA);
            char mark=0;
            vector<int>::iterator it=a.begin();
            for(i=0;it+i!=a.end();++i)
            {
                if(*(it+i)==opA)
                {
                    mark=1;
                    printf("%d\n",++i);
                    break;
                }
            }
            if(!mark)
            {
                printf("Wrong input!\n");
            }
        }
        else if(!strcmp(operation,Getcha))
        {
            scanf("%d",&opA);
            vector<int>::iterator it=a.begin();
            if(opA>a.size()||opA<=0)
            {
                printf("Wrong input!\n");
            }
            else
            {
                printf("%d\n",*(it+opA-1));
            }
```

```
            }
        }
    }
}
```

OJ编号290□KevinFeng写作文。

```
#include<iostream>
#include<vector>

using namespace std;

string add = "Add";
string del = "Del";
string rep = "Rep";
string a;
string op;
int opA;
char opB;
int opC;
int n, m;

int main()
{
    while(cin >> n >> m)
    {
        cin >> a;
        vector<char> b;
        int i;
        for(i = 0; i < a.length(); ++i)
        {
            b.push_back(a[i]);
        }
        while(m--)
        {
            cin >> op;
            if(op == add)
            {
                cin >> opA >> opB;
                b.insert(b.begin()+opA-1, opB);
            }
            else if (op == del)
            {
                cin >> opA >> opC;
                while(opC--)
                {
                    b.erase(b.begin() + opA - 1);
                }
            }
```

```
            else
            {
                cin >> opA >> opB;
                b[opA - 1] = opB;
            }
        }
        vector<char>::iterator c;
        for(c=b.begin();c!=b.end();c++)
        {
            cout << *c;
        }
        cout << endl;
    }
}
```

看吧vector的insert功能就是这么方便。一般如果我们需要对一个数组（或者链表）里面执行插入操作，请使用vector来减少代码量。

~~什么？你想写链表？用STL里的list一个算法比赛用什么链表啊我的天哪（于是list就合理的不写了）~~

deque

~~DQ冰淇淋？~~

栈和队列都好写，因为它们都是单向增长或移动的。但是双端队列就不太好写了。这时候，我们需要合理运用deque的武器呢。

（如果您非要开大空间从中间计数，或者取模，我也拦不住对吧。）

OJ编号314：毛毛虫。

```
#include<stdio.h>
#include<queue>
#include<algorithm>

using namespace std;

deque<pair<int,int> > q;
bool flag;
int m, n;
int a, b;

int main()
{
    while(~scanf("%d%d", &m, &n))
    {
        flag = true;
```

```
        while (!q.empty())q.pop_back();
        for (int i = 0; i < m; ++i)
        {
            scanf("%d%d", &a, &b);
            q.push_back(make_pair(a, b));
        }
        while (n--)
        {
            scanf("%d%d", &a, &b);
            q.pop_front();
            q.push_back(make_pair(a, b));
        }
        if (q.front().first == q.back().first && q.front().second ==
q.back().second)
        {
            flag = false;
        }
        while (!q.empty())
        {
            printf("%d %d\n", q.back().first, q.back().second);
            q.pop_back();
        }
        if (!flag)
        {
            puts("oh no");
        }
    }
}
```

OJ编号315：滚动的窗口。

```
#include<stdio.h>
#include<queue>

using namespace std;

deque<int> dmin;
deque<int> dmax;

int a[100010];
int top;

inline void write(int x)
{
    if(x < 0)putchar('-'),x=-x;
    if (x > 9)write(x / 10);
    putchar(x % 10 + 48);
}
```

```
inline int read()
{
    int k = 0, f = 1;
    char c = getchar();
    while (c < '0' || c>'9')
    {
        if (c == '-')f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9')
    {
        k = (k << 1) + (k << 3) + c - 48;
        c = getchar();
    }
    return k * f;
}

int n, k;

int main()
{
    while(~scanf("%d%d", &n, &k))
    {
        int i;
        for(i = 1; i <= n; ++i)a[i] = read();
        while (!dmin.empty())dmin.pop_back();
        for (i = 1; i <= k; ++i)
        {
            if (dmin.empty())dmin.push_back(i);
            else
            {
                while (!dmin.empty() && a[dmin.back()] > a[i])
dmin.pop_back();
                dmin.push_back(i);
            }
        }
        write(a[dmin.front()]); putchar(' ');
        for (i = k + 1; i <= n; ++i)
        {
            while (!dmin.empty() && dmin.front() <= i - k)dmin.pop_front();
            if (dmin.empty())dmin.push_back(i);
            else
            {
                while (!dmin.empty() && a[dmin.back()] > a[i])
dmin.pop_back();
                dmin.push_back(i);
            }
            write(a[dmin.front()]); putchar(' ');
        }
        putchar('\n');
```

```
        while (!dmax.empty())dmax.pop_back();
        for (i = 1; i <= k; ++i)
        {
            if (dmax.empty())dmax.push_back(i);
            else
            {
                while (!dmax.empty() && a[dmax.back()] < a[i])
dmax.pop_back();
                dmax.push_back(i);
            }
        }
        write(a[dmax.front()]); putchar(' ');
        for (i = k + 1; i <= n; ++i)
        {
            while (!dmax.empty() && dmax.front() <= i - k)dmax.pop_front();
            if (dmax.empty())dmax.push_back(i);
            else
            {
                while (!dmax.empty() && a[dmax.back()] < a[i])
dmax.pop_back();
                dmax.push_back(i);
            }
            write(a[dmax.front()]); putchar(' ');
        }
        putchar('\n');
    }
}
```

set

在算法比赛里，能够用到set这种高级东西的，一般是非常困难的题目。

下面举几个例子：

OJ编号89LCSDataEnhanced

```
#include<stdio.h>
#include<string.h>
#include<math.h>

#include<string>
#include<set>

using namespace std;

char A[110],B[110];
char tmp[110];
int n,m;
int dp[110][110];
```

```
int lasta[110][27];
int lastb[110][27];
int targetLen;
set<string> ans;

void init()
{
    memset(dp,0,sizeof(dp));
    memset(lasta,0,sizeof(lasta));
    memset(lastb,0,sizeof(lastb));
    memset(tmp,0,sizeof(tmp));
    ans.clear();
}

void buildDP()
{
    int i;
    for(i=1;i<=n;++i)
    {
        int j;
        for(j=1;j<=m;++j)
        {
            if(A[i]==B[j])
            {
dp[i][j]=dp[i][j]>(dp[i-1][j-1]+1)?dp[i][j]:(dp[i-1][j-1]+1);
            }
            else
            {
                dp[i][j]=dp[i-1][j]>dp[i][j-1]?dp[i-1][j]:dp[i][j-1];
            }
        }
    }
    targetLen=dp[n][m];
}

void buildLast()
{
    int i;
    for(i=1;i<=n;++i)
    {
        int j;
        for(j=0;j<26;++j)
        {
            if(A[i]=='A'+j)
            {
                lasta[i][j]=i;
            }
            else
            {
                lasta[i][j]=lasta[i-1][j];
```

```
            }
        }
    }
    for(i=1;i<=m;++i)
    {
        int j;
        for(j=0;j<26;++j)
        {
            if(B[i]=='A'+j)
            {
                lastb[i][j]=i;
            }
            else
            {
                lastb[i][j]=lastb[i-1][j];
            }
        }
    }
}

void dfs(int i,int j,int len)
{
    if(len<=0)
    {
        ans.insert(string(tmp+1));
        return;
    }
    if(i>0&&j>0)
    {
        int k;
        for(k=0;k<26;++k)
        {
            int t1=lasta[i][k];
            int t2=lastb[j][k];
            if(dp[t1][t2]==len)
            {
                tmp[len]='A'+k;
                dfs(t1-1,t2-1,len-1);
            }
        }
    }
}

void printAns()
{
    set<string>::iterator it;
    for(it=ans.begin();it!=ans.end();++it)
    {
        puts((*it).c_str());
    }
}
```

```
int main()
{
    while(~scanf("%s%s",A+1,B+1))
    {
        init();
        n=strlen(A+1);
        m=strlen(B+1);
        buildDP();
        buildLast();
        tmp[targetLen+1]='\0';
        dfs(n,m,targetLen);
        printAns();
    }
}
```

OJ编号279□Z君的日常之集合处理。

```
#include<stdio.h>
#include<algorithm>
#include<set>
#include<vector>

using namespace std;

char tmp[3];
char op;
int t,n,m;
int tmpint;

int main()
{
    while(~scanf("%d", &t))
    {
        while(t--)
        {
            scanf("%s%d%d",tmp,&n,&m);
            op=tmp[0];
            vector<int> a,b;
            int i;
            for(i=0;i<n;++i)
            {
                scanf("%d",&tmpint);
                a.push_back(tmpint);
            }
            for(i=0;i<m;++i)
            {
                scanf("%d",&tmpint);
```

```
                b.push_back(tmpint);
            }
            sort(a.begin(),a.end());
            sort(b.begin(),b.end());
            if(op=='+')
            {
                vector<int> c;
set_union(a.begin(),a.end(),b.begin(),b.end(),back_inserter(c),less<int>())
;
                std::vector<int>::iterator m;
                for(m=c.begin();m!=c.end();m++)
                {
                    printf("%d ",*m);
                }
                printf("\n");
            }
            else
            {
                vector<int> c;
set_difference(a.begin(),a.end(),b.begin(),b.end(),back_inserter(c),less<in
t>());
                if(!c.empty())
                {
                    std::vector<int>::iterator m;
                    for(m=c.begin();m!=c.end();m++)
                    {
                        printf("%d ",*m);
                    }
                    printf("\n");
                }
                else
                {
                    printf("empty\n");
                }
            }
        }
    }
}
```

OJ编号516□ModricWang的星灵棋。

```
#include<stdio.h>
#include<string.h>

#include<set>

using namespace std;

set<unsigned int> vis[5];
```

```
bool win[171];

struct State
{
    unsigned char row[4],col[4];
    unsigned char left,right;
    unsigned char rc[2][2];
    short step,turn;
};

bool putChess(struct State *s,const int r,const int c,unsigned char p)
{
    unsigned char pl2r=p<<2*r,pl2c=p<<2*c,al2r=~(3<<2*r),al2c=~(3<<2*c);
    s->row[r]&=al2c,s->row[r]|=pl2c;
    if(win[s->row[r]])
    {
        return 1;
    }
    s->col[c]&=al2r,s->col[c]|=pl2r;
    if(win[s->col[c]])
    {
        return 1;
    }
    if(r==c)
    {
        s->left&=al2c,s->left|=pl2c;
        if(win[s->left])
        {
            return 1;
        }
    }
    if(r+c==3)
    {
        s->right&=al2r,s->right|=pl2r;
        if(win[s->right])
        {
            return 1;
        }
    }
    return 0;
}

struct State q[666666];

int head;
int tail;

bool getChess(int r,int c)
{
```

```
    if(r<0||c<0||r>3||c>3)
    {
        return 0;
    }
    else
    {
        return ((unsigned char)(q[head].row[r]>>2*c)&3)==q[head].turn;
    }
}

void before(int first,int second,int kind)
{
    q[tail]=q[head];
    q[tail].rc[first][second]+=kind;
    ++q[tail].step;
}

void after()
{
    if(!vis[q[head].turn+1].count(*(unsigned int*)q[tail].row))
    {
        vis[q[head].turn+1].insert(*(unsigned int*)q[tail].row);
        q[tail].turn=3-q[head].turn;
        tail++;
    }
}

int main()
{
    struct State src;
    memset(&src,0,sizeof src);
    win[170]=win[85]=1;
    int firstzero=1;
    int r;
    for(r=0;r<4;++r)
    {
        char str[123];
        scanf("%s",str);
        int c;
        for(c=0;c<4;++c)
        {
            switch(str[c])
            {
                case 'B':
                    if(putChess(&src,r,c,1))
                    {
                        puts("0");
                        return 0;
                    }
                    break;
                case 'W':
```

```
                    if(putChess(&src,r,c,2))
                    {
                        puts("0");
                        return 0;
                    }
                    break;
                case '0':
                    if(firstzero)
                    {
                        src.rc[1][0]=r,src.rc[0][0]=c,firstzero=0;
                    }
                    else
                    {
                        src.rc[1][1]=r,src.rc[0][1]=c;
                    }
                    break;
            }
        }
    }
    head=0,tail=2;
    q[0]=q[1]=src;
    q[1].turn=2;
    vis[2].insert(*(unsigned int*)src.row);
    vis[3].insert(*(unsigned int*)src.row);
    for(;head!=tail;head++)
    {
        if(getChess(q[head].rc[1][1]-1,q[head].rc[0][1]))
        {
            before(1,1,-1);
if(putChess(&q[tail],q[tail].rc[1][1],q[head].rc[0][1],0)||putChess(&q[tail
],q[head].rc[1][1],q[head].rc[0][1],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][1],q[head].rc[0][1]-1))
        {
            before(0,1,-1);
if(putChess(&q[tail],q[head].rc[1][1],q[tail].rc[0][1],0)||putChess(&q[tail
],q[head].rc[1][1],q[head].rc[0][1],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][1]+1,q[head].rc[0][1]))
        {
```

```
            before(1,1,1);
if(putChess(&q[tail],q[tail].rc[1][1],q[head].rc[0][1],0)||putChess(&q[tail
],q[head].rc[1][1],q[head].rc[0][1],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][1],q[head].rc[0][1]+1))
        {
            before(0,1,1);
if(putChess(&q[tail],q[head].rc[1][1],q[tail].rc[0][1],0)||putChess(&q[tail
],q[head].rc[1][1],q[head].rc[0][1],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][0]-1,q[head].rc[0][0]))
        {
            before(1,0,-1);
if(putChess(&q[tail],q[tail].rc[1][0],q[head].rc[0][0],0)||putChess(&q[tail
],q[head].rc[1][0],q[head].rc[0][0],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][0],q[head].rc[0][0]-1))
        {
            before(0,0,-1);
if(putChess(&q[tail],q[head].rc[1][0],q[tail].rc[0][0],0)||putChess(&q[tail
],q[head].rc[1][0],q[head].rc[0][0],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
        if(getChess(q[head].rc[1][0]+1,q[head].rc[0][0]))
        {
            before(1,0,1);
if(putChess(&q[tail],q[tail].rc[1][0],q[head].rc[0][0],0)||putChess(&q[tail
],q[head].rc[1][0],q[head].rc[0][0],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
```

```
            after();
        }
        if(getChess(q[head].rc[1][0],q[head].rc[0][0]+1))
        {
            before(0,0,1);
if(putChess(&q[tail],q[head].rc[1][0],q[tail].rc[0][0],0)||putChess(&q[tail
],q[head].rc[1][0],q[head].rc[0][0],q[head].turn))
            {
                printf("%d",q[tail].step);
                return 0;
            }
            after();
        }
    }
}
```

~~这些题都太难了，唉~~

map

priority_queue

来跟我一起念□PO-RI-O-RI-TI-KYU□□罗马音：**ポリオリティキュ！**）

是的。如果你不把第一个连读音节拆开念的话，一定会忘记打一个字母小r的。这可是惨痛的教训啊——

优先队列的本质是一个堆（二叉树），并且是**大顶堆**。与其他STL不同，数据结构课用的不多，但是在算法课上最经常用到。

在诸多算法里都会用优先队列来优化算法，尤其是**贪心算法**，在各种贪心算法中都会见到。比如著名的最短路Dijkstra□

在这里就不写了。参见广度优先搜索_bfs_与标数最短路_dijkstra□

这个文中也提到，如何将默认大顶堆**转换为小顶堆**呢？**取相反数**存储与读取就完了。这是个很经典的技巧，相比之下在定义中修改优先级略为难记。

不过一定要记得读取的时候也要取相反——存的时候是反着存的，很容易忘。

对于带结构体的优先队列，如果不想引入大小判定符号，也没有字典序这种糟糕的条件，就嵌套使用**pair**吧！默认结构体pair的优先级默认是先比较first再比较second□省得再写难记的大小判定了！