20180408_自定义比较函数cmp

实际只有前三种方法。

以priority_queue的用法为例 **方法一**

```
*********************************方法一
struct Node {
    int x,y;
    bool operator <(Node a) const  {    //必须加const
        return y < a.y;
    }
    bool operator >(Node a) const  {    //必须加const
        return y > a.y;
    }
};
//  priority_queue<Node> A;   //默认  大根堆
priority_queue<Node, vector<Node>, less<Node>>A;         //大根堆
priority_queue<Node, vector<Node>, greater<Node> > B;    //小根堆
```

类似方法一

```
*********************************类似方式法一
struct Node {
    int x;
    int y;
    friend  bool operator<(const Node &a,const Node &b) {
        return  a.x < b.x;  //大顶堆
    }
    friend  bool operator>(const Node &a,const Node &b) {
        return  a.x > b.x;  //小顶堆
    }
};

priority_queue<Node> A;   //默认  大根堆
priority_queue<Node, vector<Node>, greater<Node> > B;    //小根堆
```

方法二

```
*********************************方法二:
struct Node {
    int x;
```

```
    int y;
};

bool operator<(const Node &a, const Node &b) {
    return a.x<b.x;           //大顶堆
}

bool operator>(const Node &a, const Node &b) {
    return a.x>b.x;          //小顶堆
}

priority_queue<Node,vector<Node>,less<Node> > A;     //大根堆
priority_queue<Node, vector<Node>, greater<Node> > B;     //小根堆
```

方法三

```
//*******************************方法三:
struct Node {
    int x;
    int y;
};

struct cmp {
    bool operator()(Node a,Node b) {
        return  a.x > b.x;  //小顶堆
    }
};

struct cmp1 {
    bool operator()(Node a,Node b) {
        return  a.x < b.x;  //大顶堆
    }
};

priority_queue<Node,vector<Node>,cmp1 > A;       //大根堆
priority_queue<Node, vector<Node>, cmp > B;     //小根堆
```

队列节点是指针

```
//当队列节点是指针时，用法不同
struct Node {
    int x;
    int y;
};
struct cmp {
    bool operator () (Node const *n1, Node const *n2) {
        return n1->x<n2->x;      //大顶推
```

```
    }
};
struct cmp1 {
    bool operator () (Node const *n1, Node const *n2) {
        return n1->x>n2->x;     //小顶推
    }
};

priority_queue<Node*, vector<Node*>, cmp > A;   //大根堆
priority_queue<Node*, vector<Node*>, cmp1 > B;  //小根堆
```

统一测试

```
ostream & operator <<(ostream &out,const struct Node& n) {
    out<<"n.x="<<n.x<<"    n.y="<<n.y<<endl;
    return out;
}

const vector<Node>tn= {{1,1},{2,2},{3,3},{4,4},{5,5}};

void test() {
    for (auto &a:tn) {
        A.push(a);
        B.push(a);
    }

    cout<<"A:"<<endl;
    while(!A.empty()) {
        cout<<A.top();
        A.pop();
    }

    cout<<"B:"<<endl;
    while(!B.empty()) {
        cout<<B.top();
        B.pop();
    }
}

int main() {
    test();
    return 0;
}
```