计算机科学基础 **Summer 2024**

第 1 节: 基础数据结构 (链表, 栈, 队列, 堆) 和 STL

Lecturer: 张桄玮 *Scribes:* ________

学习编程最重要的事情就是把内心所想表达出来.

— Yanyan Jiang

§1 本节概述

程序 = 算法 + 数据结构。这就像拿着说明书操作某些东西一样。今天我们讲了基本的数据结构：链表（到处都是）、栈、队列、优先队列以及 STL。它们遵循的逻辑各不相同。我们先手写代码，然后再使用 STL 库来实现这些数据结构。

§2 链表

内存中的数据, 除了可以直接表示数据什么, 还可以间接表示, 即告诉我要的数据在什么地方. 这就是所谓的指针. 指针对于初学者比较难以理解 (即使不适用 C++ 指针的写法). 但是掌握了这项技能就可以得到很多的便利. 而链表就是最简单的指针练习.

$\boxed{1}$ *什么是链表? 给出熟悉的和递归的定义.*

下面考虑单向链表. 也就是每一个结构都有一个类似与 next 表示下一个链表.

$\boxed{2}$ *如何代表 "这个结构的下一个元素还是自己这样的"? 如何表示下一个元素已经不存在了?*

$\boxed{3}$ *链表的基本操作有哪些? 有什么特点? 如何用代码维护这些特点?(初始化, 插入, 删除).*

$\boxed{4}$ *为什么引入双向链表?*

下面考虑双向链表.

$\boxed{5}$ *双向链表的结构是怎么样的?*

$\boxed{6}$ *请你画出草图模拟, 说说为什么这样维护是合理的. 额外留心边界情况!*

下面考虑双向循环链表, 并且头部有一个 dummy 节点.

$\boxed{7}$ *请你画出草图, 说说这样维护为什么合理.*

```
typedef struct task{
    struct task *nxt, *prv;
    char name[8];
    // .... 别的什么东西 ....
}task_t;

typedef struct __tasks_lst{
```

```c
  struct task dummy; // 第一个节点
  int nr_node;       // 总共节点的个数
} TSKLST;



void init_tsklst(TSKLST *tsklst){
  tsklst->nr_node = 0;
  // 初始的内容让空白节点的前后都指着自己
  tsklst->dummy.prv = tsklst->dummy.nxt = &(tsklst->dummy);
}

void prepend_tnode(TSKLST *bd, task_t *tsk){
  if(bd->nr_node == 0){
    bd->dummy.nxt = bd->dummy.prv = tsk;
    tsk->nxt = tsk->prv = &bd->dummy;
    bd->nr_node++;
    return ;
  }

  task_t *u = tsk;
  task_t *w = bd->dummy.nxt;
  u->prv = w->prv;
  u->nxt = w;
  u->nxt->prv = u;
  u->prv->nxt = u;

  bd->nr_node++;
}

void remove_tnode(TSKLST *bd, task_t *curtsk){
  task_t *w = curtsk;
  w->prv->nxt = w->nxt;
  w->nxt->prv = w->prv;
  bd->nr_node--;
  panic_on(bd->nr_node < 0, "Linked list count is lower than 0!");
}
```

§3 调试技巧

$\boxed{1}$ *如何使用命令行编译程序？如何使用调试器 gdb?*

$\boxed{2}$ *assert 和 panic_on 会让程序崩溃. 这有什么作用?*

$\boxed{3}$ *预编译指令 define 和 include 以及 ifdef 有什么作用?*

§4 栈

$\boxed{1}$ *什么是栈？给出熟悉的定义.*

$\boxed{2}$ *栈的基本操作有哪些？有什么特点？如何用代码维护这些特点?(初始化，入栈，出栈).*

$\boxed{3}$ *栈是一类某些语法分析算法的基础. 使用栈是如何解析表达式的?*

$\boxed{4}$ *栈与递归有什么联系?*

§5 队列

$\boxed{1}$ *什么是队列？给出熟悉的定义.*

$\boxed{2}$ *队列的基本操作有哪些？有什么特点？如何用代码维护这些特点?(初始化，入队，出队).*

为了方便管理，使用数组模拟队列的时候，可以使用循环队列的方式. 下面考虑循环队列

$\boxed{3}$ *入队的时候，出队的时候应该怎么做？什么情况下队列满？*

有时候需要两端既能够入队，又能够出队的队列 (双端队列). 下面考虑双端队列:

$\boxed{4}$ *如何使用数组模拟？ (不用处理边界溢出的情况) 如何使用 STL 的 deque?*

$\boxed{5}$ *单调队列满足了队列的单调性. 说说它为什么可以保持单调.*

§6 堆和优先队列

使用堆可以实现优先队列.

$\boxed{1}$ *如何在内存中表示二叉树？*

$\boxed{2}$ *在堆中，把一个节点往上调的条件是什么？往下沉的条件是什么？*

$\boxed{3}$ *如何用上述的两个操作构造整个堆的插入，删除？*

§7 附录: 代码片段

7.1. P1160 队列安排.

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

const int N = 100003;
int prv[N], nxt[N], idx;
int n, m;

void init() {
    // 0 表示左端点, 1 表示右端点
    nxt[0] = 1;
    prv[1] = 0;
    nxt[1] = -1;
    prv[0] = -1;
    idx = 2; // 从2开始
    for (int i = 2; i <= n; ++i) {
        prv[i] = nxt[i] = -1;
    }
}

// 在内存池中编号为 pos 的节点右边插入编号为 x 的节点
inline void add_right(int pos, int x) {
    prv[x] = pos;
    nxt[x] = nxt[pos];
    if (nxt[pos] != -1) prv[nxt[pos]] = x;
    nxt[pos] = x;
}

// 在 内 存 池 中 编 号 为 pos 的 节 点 左 边 插 入 编 号 为 x 的 节 点
inline void add_left(int pos, int x) {
    nxt[x] = pos;
    prv[x] = prv[pos];
    if (prv[pos] != -1) nxt[prv[pos]] = x;
    prv[pos] = x;
}

// 删 除 内 存 池 里 面 编 号 为 x 的 节 点
inline void remove(int x) {
    if (prv[x] == -1) return;
    nxt[prv[x]] = nxt[x];
    if (nxt[x] != -1) prv[nxt[x]] = prv[x];
    prv[x] = nxt[x] = -1;
}

// 遍 历 链 表 并 输 出 节 点 的 值
inline void traverse() {
    int x = nxt[0];
    while (x != -1) {
        cout << x << " ";
        x = nxt[x];
    }
    cout << endl;
}

int main() {
    scanf("%d", &n);
    int cmd1, cmd2;
    init();
    for (int i = 2; i <= n; ++i) {
        scanf("%d %d", &cmd1, &cmd2);
        if (cmd2 == 0) add_left(cmd1, i);
        else add_right(cmd1, i);
    }
    scanf("%d", &m);
    for (int i = 1; i <= m; ++i) {
        scanf("%d", &cmd1);
        remove(cmd1);
    }
    traverse();
    return 0;
}
```

7.2. P1996 约瑟夫问题.

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

const int N = 100010;
int val[N], prv[N], nxt[N], idx;
int n, m;

void init(int n) {
    // 0 表 示 左 端 点 , 1 表 示 右 端 点
    nxt[0] = 1;
    prv[1] = 0;
    idx = 2; // 从2开始
    for (int i = 1; i <= n; ++i) {
        val[i] = i;
        if (i != n) nxt[i] = i + 1;
        else nxt[i] = 1; // 形成环
        if (i != 1) prv[i] = i - 1;
        else prv[i] = n; // 形成环
    }
}

// 删除内存池里面编号为 x 的节点
inline void remove(int x) {
    nxt[prv[x]] = nxt[x];
    prv[nxt[x]] = prv[x];
}

int main() {
    scanf("%d %d", &n, &m);
    init(n);
    int current = 1;

    for (int i = 0; i < n; ++i) {
        // 找到第 m 个要出列的人
        for (int j = 1; j < m; ++j) {
            current = nxt[current];
        }
        // 输出该人的编号
        printf("%d ", val[current]);
        // 删除该人
        remove(current);
        // 更新 current 为下一个人的编号
        current = nxt[current];
    }

    return 0;
}
```

7.3. UVA11988 破碎的键盘.

```
    #include <cstdio>
    #include <cstring>

    const int maxn = 100000 + 5;
    int last, cur, next[maxn];
    char s[maxn];

    int main() {
        while (scanf("%s", s + 1) == 1) {
            int n = strlen(s + 1);
            last = cur = 0;
            next[0] = 0;
            for (int i = 1; i <= n; i++) {
                char ch = s[i];
                if (ch == '[') {
                    cur = 0;
                } else if (ch == ']') {
                    cur = last;
                } else {
                    next[i] = next[cur];
                    next[cur] = i;
                    if (cur == last) {
                        last = i;
                    }
                    cur = i;
                }
            }
            for (int i = next[0]; i != 0; i = next[i]) {
                printf("%c", s[i]);
            }
            printf("\n");
        }
        return 0;
    }
```

7.4. UVA12657 盒子排队.

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

const int maxn = 100005;
int nxt[maxn], prv[maxn];
int n, m;
bool reversed;

void init(int n) {
    for (int i = 1; i <= n; ++i) {
        nxt[i] = i + 1;
        prv[i] = i - 1;
    }
    nxt[0] = 1;
    prv[n + 1] = n;
    reversed = false;
}

void remove(int x) { // 删除
    nxt[prv[x]] = nxt[x];
    prv[nxt[x]] = prv[x];
}

void insert(int l, int r) {
    if (nxt[l] == r || l == r) return;
    remove(l);
    nxt[prv[r]] = l;
    prv[l] = prv[r];
    prv[r] = l;
    nxt[l] = r;
}

void swp(int l, int r) {
    int k = nxt[l];
    insert(l, nxt[r]);
    insert(r, k);
}

long long sumOddPositions() {
    long long sum = 0;
    int current = nxt[0];
    int pos = 1;
    while (current != n + 1 && current != 0) {
        if (pos % 2 != 0) sum += current;
        current = nxt[current];
        pos++;
    }
    return sum;
}

int main() {
    int caseNum = 1;
    while (scanf("%d %d", &n, &m) != EOF) {
        init(n);
        long long ans = 0;
        for (int i = 1; i <= m; ++i) {
            int x, l, r;
            scanf("%d", &x);
            if (x == 1) {
                scanf("%d %d", &l, &r);
                if (!reversed) insert(l, r);
                else insert(l, nxt[r]);
            } else if (x == 2) {
                scanf("%d %d", &l, &r);
                if (!reversed) insert(l, nxt[r]);
                else insert(l, r);
            } else if (x == 3) {
                scanf("%d %d", &l, &r);
                swp(l, r);
            } else if (x == 4) {
                reversed = !reversed;
            }
        }
        if (reversed) {
            swap(prv[n + 1], nxt[0]);
            for (int i = 1; i <= n; ++i) swap(prv[i], nxt[i]);
        }
        ans = sumOddPositions();
        printf("Case %d: %lld\n", caseNum++, ans);
    }
    return 0;
}
```

7.5. B3614 栈.

```
    #include <cstdio>
    #include <cstring>
    #include <iostream>
    using namespace std;

    const int maxn = 100000 + 5;

    unsigned long long stack[maxn];
    int top;

    void push(unsigned long long x) {
        stack[++top] = x;
    }

    void pop() {
        if (top == 0) {
            printf("Empty\n");
        } else {
            top--;
        }
    }

    void query() {
        if (top == 0) {
            printf("Anguei!\n");
        } else {
            printf("%llu\n", stack[top]);
        }
    }

    void size() {
        printf("%d\n", top);
    }

    int main() {
        int T;
        scanf("%d", &T);
        while (T--) {
            int n;
            scanf("%d", &n);
            top = 0; // 重置栈顶
            while (n--) {
                char operation[10];
                scanf("%s", operation);
                if (strcmp(operation, "push") == 0) {
                    unsigned long long x;
                    scanf("%llu", &x);
                    push(x);
                } else if (strcmp(operation, "pop") == 0) {
                    pop();
                } else if (strcmp(operation, "query") == 0) {
                    query();
                } else if (strcmp(operation, "size") == 0) {
                    size();
                }
            }
        }
        return 0;
    }
```

7.6. P1739 表达式括号匹配.

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

#define MAX_SIZE 1000 // 定义栈的最大大小

char stack[MAX_SIZE]; // 使用数组来模拟栈
int top = -1; // 栈顶指针

void push(char c) {
    if (top < MAX_SIZE - 1) {
```

```cpp
        stack[++top] = c;
    }
}

void pop() {
    if (top >= 0) {
        top--;
    }
}

bool isEmpty() {
    return top == -1;
}

int main() {
    char input;
    while (cin >> input && input != '@') {
        if (input == '(') push(input);
        if (input == ')') {
            if (isEmpty()) {
                printf("NO\n");
                return 0;
            }
            pop();
        }
    }
    if (isEmpty()) cout << "YES";
    else cout << "NO";
    return 0;
}
```

7.7. UVA514 铁轨.

```cpp
#include <cstdio>
#include <cstring>

const int MAXN = 1010;
int train[MAXN];
int stack[MAXN];
int top;

void push(int x) {
    stack[++top] = x;
}

void pop() {
    top--;
}

int query() {
    return stack[top];
}

int main() {
    int n, A, B, ok;
    while (scanf("%d", &n), n) {
        while (1) {
            scanf("%d", &train[1]);
            if (train[1] == 0) break;
            for (int i = 2; i <= n; i++) {
                scanf("%d", &train[i]);
            }

            A = B = ok = 1;
            top = 0;

            while (B <= n) {
                if (A == train[B]) {
                    A++;
                    B++;
                }else if (top > 0 && stack[top] == train[B]) {
                    pop();
                    B++;
                }else if (A <= n) {
                    push(A++);
                }else {
                    ok = 0;
                    break;
                }
            }
            printf("%s\n", ok ? "Yes" : "No");
        }
        printf("\n");
    }
    return 0;
}
```

7.8. P1449 后缀表达式.

```
    #include<iostream>
    #include<cstdio>
    using namespace std;
    long long stk[1000];
    int main() {
        long long i=0,now=0;
        char operators;
        while((operators=getchar())!='@') {
            if(operators>='0'&&operators<='9') now*=10,now+=operators-'0';
            else if(operators=='.') {
                stk[++i]=now;
                now=0;
            } else if(operators=='+') {
                stk[i-1]=stk[i-1]+stk[i];
                stk[i]=0;
                i--;
            } else if(operators=='-') {
                stk[i-1]=stk[i-1]-stk[i];
                stk[i]=0;
                i--;
            } else if(operators=='*') {
                stk[i-1]=stk[i-1]*stk[i];
                stk[i]=0;
                i--;
            } else if(operators=='/') {
                stk[i-1]=stk[i-1]/stk[i];
                stk[i]=0;
                i--;
            }
    }
    cout<<stk[1];
    return 0;
}
```

7.9. P1175 表达式转换.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#define int long long

int priority(char ch) {
    switch(ch) {
        case '(': case ')': return 0;
        case '+': case '-': return 1;
        case '*': case '/': return 2;
        case '^': return 3;
    }
    return -1;
}

int rassoc(char ch){
    return ch == '^';
}


char* suffix(const char* str) {
    char* s = (char*)malloc(strlen(str) * sizeof(char));
    char* tmp = (char*)malloc((strlen(str) + 1) * sizeof(char));
    int s_top = -1, tmp_len = 0;

    for(int i = 0; i < strlen(str); ++i) {
        if(isdigit(str[i])) {
            tmp[tmp_len++] = str[i];
        } else if(str[i] == '(') {
            s[++s_top] = str[i];
        } else if(str[i] == ')') {
            while(s_top >= 0 && s[s_top] != '(') {
                tmp[tmp_len++] = s[s_top--];
            }
            --s_top;
        } else {
            while(s_top >= 0 &&
                  priority(s[s_top]) >= priority(str[i]) &&
                  !rassoc(str[i])) {
                tmp[tmp_len++] = s[s_top--];
            }
            s[++s_top] = str[i];
        }
    }
    while(s_top >= 0) {
        tmp[tmp_len++] = s[s_top--];
    }
    tmp[tmp_len] = '\0';
    free(s);
    return tmp;
}

int applycalc(char ident, int num1, int num2) {
    switch(ident) {
        case '+': return num1 + num2;
        case '-': return num1 - num2;
        case '*': return num1 * num2;
        case '/': return num1 / num2;
        case '^': return (int) pow(num1, num2);
    }
    return -1;
}

void prtsuffix(const char* tmp) {
    for(int i = 0; i < strlen(tmp); ++i) {
        printf("%c ", tmp[i]);
    }
    printf("\n");
}

void calcPrint(const char* str) {
    int* ls = (int*)malloc(strlen(str) * sizeof(int));
    int ls_len = 0;
    prtsuffix(str);

    for(int i = 0; i < strlen(str); ++i) {
        if(isdigit(str[i])) {
            ls[ls_len++] = str[i] - '0';
        } else {
            int num1 = ls[--ls_len];
            int num2 = ls[--ls_len];
            ls[ls_len++] = applycalc(str[i], num2, num1);

            for(int j = 0; j < ls_len; ++j) {
                printf("%d ", ls[j]);
            }
            for(int j = i + 1; j < strlen(str); ++j) {
                printf("%c ", str[j]);
            }
            printf("\n");
        }
    }
    free(ls);
}

signed main() {
    char str[100];
    scanf("%s", str);
    char* psuffix = suffix(str);
    calcPrint(psuffix);
    free(psuffix);
    return 0;
}
```

7.10. B3616 队列.

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

```

#define NR_DAT 10003

struct cqueue {
    int data[NR_DAT];
    int front, rear;

    bool init() {
        front = rear = 0;
        return true;
    }

    int size() {
        return (rear - front + NR_DAT) % NR_DAT;
    }

    bool isempty() {
        return (size() == 0);
    }

    bool push(int e) {
        if ((rear + 1) % NR_DAT == front) return false; // full!
        data[rear] = e;
        rear = (rear + 1) % NR_DAT;
        return true;
    }

    bool pop(int &e) {
        if (front == rear) return false;
        e = data[front];
        front = (front + 1) % NR_DAT;
        return true;
    }

    int getfront() {
        if (front == rear) return -1; // indicate empty queue
        return data[front];
    }
};

int main() {
    int n;
    cin >> n;
    cqueue q;
    q.init();
    for (int i = 0; i < n; ++i) {
        int op;
        cin >> op;
        if (op == 1) {
            int x;
            cin >> x;
            q.push(x);
        } else if (op == 2) {
            int x;
            if (q.pop(x)) {
                // cout<<x<<endl;
            } else {
                cout << "ERR_CANNOT_POP" << endl;
            }
        } else if (op == 3) {
            if (q.isempty()) {
                cout << "ERR_CANNOT_QUERY" << endl;
            } else {
                cout << q.getfront() << endl;
            }
        } else if (op == 4) {
            cout << q.size() << endl;
        }
    }
    return 0;
}
```

7.11. P1886 滑动窗口.

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

const int MAXN = 1000009;
int num[MAXN];
int n, k;

struct Deque {
    int q[MAXN]; // 存储队列元素的数组
    int head, tail;

    Deque() {
        head = 0;
        tail = -1;
    }

    // 检查队列是否为空
    bool empty() {
        return head > tail;
    }

    // 获取队头元素
    int front() {
        return q[head];
    }

    // 获取队尾元素
    int back() {
        return q[tail];
    }

    // 弹出队头元素
    void pop_front() {
        if (!empty()) head++;
    }

    // 弹出队尾元素
    void pop_back() {
        if (!empty()) tail--;
    }

    // 向队尾添加元素
    void push_back(int val) {
        q[++tail] = val;
    }
```

```cpp

    // 清空队列
    void clear() {
        head = 0;
        tail = -1;
    }
};

int main() {
    cin >> n >> k;
    for (int i = 0; i < n; i++) {
        cin >> num[i];
    }

    Deque minDeque, maxDeque;

    // 最小值队列处理
    int t = 0;
    for (int i = 0; i < n; i++) {
        while (!minDeque.empty() && num[minDeque.back()] >= num[i]) minDeque.
    pop_back();
        minDeque.push_back(i);

        if (i - t >= k && minDeque.front() == t) {
            t++;
            minDeque.pop_front();
        }
        if (i - t >= k && minDeque.front() != t) t++;

        if (i >= k - 1) cout << num[minDeque.front()] << ' ';
    }
    cout << endl;

    // 最大值队列处理
    t = 0;
    for (int i = 0; i < n; i++) {
        while (!maxDeque.empty() && num[maxDeque.back()] <= num[i]) maxDeque.
    pop_back();
        maxDeque.push_back(i);

        if (i - t >= k && maxDeque.front() == t) {
            t++;
            maxDeque.pop_front();
        }
        if (i - t >= k && maxDeque.front() != t) t++;

        if (i >= k - 1) cout << num[maxDeque.front()] << ' ';
    }

    return 0;
}
```

7.12. 求 m 区间的最小值.

```cpp
    #include<cstdio>
int n,m,a[2000000],q[2000000],l=1,r=0;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=n;i++){
        printf("%d\n",a[q[l]]);
        if(i-q[l]+1>m && l<=r) l++;
        while(a[i]<a[q[r]] && l<=r) r--;
        q[++r]=i;
    }
}
```

7.13. P3378 堆.

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

const int MAXN = 1e6 + 10;

struct MinHeap {
    int size;
    int heap[MAXN];

    MinHeap() : size(0) {}

    void push_up(int i, int val) {
        while (i > 1 && val < heap[i / 2]) {
            heap[i] = heap[i / 2];
            i /= 2;
        }
        heap[i] = val;
    }

    void push_down(int i, int val) {
        int ch = i * 2;
        while (ch <= size) {
            if (ch < size && heap[ch + 1] < heap[ch]) ch++;
            if (val <= heap[ch]) break;
            heap[i] = heap[ch];
            i = ch;
            ch *= 2;
        }
        heap[i] = val;
    }

    void insert(int val) {
        int i = ++size;
        push_up(i, val);
    }

    void delete_min() {
        int i = 1;
        int val = heap[size--];
        push_down(i, val);
    }

    int get_min() const {
        return heap[1];
    }
};

int main() {
```

```cpp
    int n;
    cin >> n;
    MinHeap minHeap;

    for (int i = 1; i <= n; i++) {
        int opt, x;
        cin >> opt;
        if (opt == 1) {
            cin >> x;
            minHeap.insert(x);
        } else if (opt == 2) {
            cout << minHeap.get_min() << endl;
        } else if (opt == 3) {
            minHeap.delete_min();
        }
    }

    return 0;
}
```

7.14. P1168 中位数.

```cpp
#include <cstdio>

const int MAXN = 100100;

struct Heap {
    int heap[MAXN];
    int size;
    bool (*cmp)(int, int);

    Heap(bool (*cmpFunc)(int, int)) : size(0), cmp(cmpFunc) {}

    void push(int val) {
        heap[++size] = val;
        int i = size;
        while (i > 1 && cmp(heap[i], heap[i / 2])) {
            swap(heap[i], heap[i / 2]);
            i /= 2;
        }
    }

    void pop() {
        heap[1] = heap[size--];
        int i = 1;
        while (i * 2 <= size) {
            int j = i * 2;
            if (j < size && cmp(heap[j + 1], heap[j])) j++;
            if (cmp(heap[i], heap[j])) break;
            swap(heap[i], heap[j]);
            i = j;
        }
    }

    int top() {
        return heap[1];
    }

    bool empty() {
        return size == 0;
    }

    void swap(int &a, int &b) {
        int temp = a;
        a = b;
        b = temp;
    }
};

bool cmp1(int a, int b) {
    return a > b;
}

bool cmp2(int a, int b) {
    return a < b;
}

int main() {
    int n, x, y;
    scanf("%d", &n);

    Heap que1(cmp1); // max-heap
    Heap que2(cmp2); // min-heap

    scanf("%d", &x);
    que1.push(x);
    printf("%d\n", x);

    for (int i = 3; i <= n; i += 2) {
        scanf("%d%d", &x, &y);
        if (x > y) que1.swap(x, y);
        que1.push(x);
        que2.push(y);

        if (que1.top() > que2.top()) {
            int a = que1.top(), b = que2.top();
            que1.pop();
            que1.push(b);
            que2.pop();
            que2.push(a);
        }
        printf("%d\n", que1.top());
    }

    return 0;
}
```

7.15. P1631 序列合并.

```
    #include <iostream>
#include <cstdio>
#include <vector>
using namespace std;
const int MAXN = 100100;

struct Data {
    int value;
    int index;
    bool operator<(const Data &other) const {
        return value > other.value;
    }
};

struct Heap {
    Data heap[MAXN];
    int size;
    bool (*cmp)(Data, Data);

    Heap(bool (*cmpFunc)(Data, Data)) : size(0), cmp(cmpFunc) {}

    void push(Data val) {
        heap[++size] = val;
        int i = size;
        while (i > 1 && cmp(heap[i], heap[i / 2])) {
            swap(heap[i], heap[i / 2]);
            i /= 2;
        }
    }

    void pop() {
        heap[1] = heap[size--];
        int i = 1;
        while (i * 2 <= size) {
            int j = i * 2;
            if (j < size && cmp(heap[j + 1], heap[j])) j++;
            if (cmp(heap[i], heap[j])) break;
            swap(heap[i], heap[j]);
            i = j;
        }
    }

    Data top() {
        return heap[1];
    }

    bool empty() {
        return size == 0;
    }

    void swap(Data &a, Data &b) {
        Data temp = a;
        a = b;
        b = temp;
    }
};

bool minHeap(Data a, Data b) {
    return a.value < b.value;
}

int a[MAXN], b[MAXN], t[MAXN];
int n;
Heap q(minHeap);
int main() {
    // 读取 n 的值
    cin >> n;

    // 读取数组 a 的值
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }

    // 读取数组 b 的值
    for (int i = 1; i <= n; ++i) {
        cin >> b[i];
        t[i] = 1;
        q.push({a[1] + b[i], i});
    }

    // 输出结果并更新堆
    while (n--) {
        Data top = q.top();
        printf("%d ", top.value);
        int i = top.index;
        q.pop();
        q.push({a[++t[i]] + b[i], i});
    }

    return 0;
}
```