

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: 张桃玮

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.

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.

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?

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).

4 为什么引入双向链表?

下面考虑双向链表.

5 双向链表的结构是怎么样的?

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.

7 请你画出草图, 说说这样维护为什么合理.

```
1 typedef struct task{
2     struct task *nxt, *prv;
3     char name[8];
4     // .... 别的什么东西 ....
5 }task_t;
6
7 typedef struct __tasks_lst{
```

```

8  struct task dummy; // 第一个节点
9  int nr_node;        // 总共节点的个数
10 } TSKLST;
11
12
13
14 void init_tsklst(TSKLST *tsklst){
15     tsklst->nr_node = 0;
16     // 初始的内容让空白节点的前后都指着自己
17     tsklst->dummy.prv = tsklst->dummy.nxt = &(tsklst->dummy);
18 }
19
20 void prepend_tnode(TSKLST *bd, task_t *tsk){
21     if(bd->nr_node == 0){
22         bd->dummy.nxt = bd->dummy.prv = tsk;
23         tsk->nxt = tsk->prv = &bd->dummy;
24         bd->nr_node++;
25         return ;
26     }
27
28     task_t *u = tsk;
29     task_t *w = bd->dummy.nxt;
30     u->prv = w->prv;
31     u->nxt = w;
32     u->nxt->prv = u;
33     u->prv->nxt = u;
34
35     bd->nr_node++;
36 }
37
38 void remove_tnode(TSKLST *bd, task_t *curtsk){
39     task_t *w = curtsk;
40     w->prv->nxt = w->nxt;
41     w->nxt->prv = w->prv;
42     bd->nr_node--;
43     panic_on(bd->nr_node < 0, "Linked list count is lower than 0!");
44 }

```

§3 调试技巧

- 1 如何使用命令行编译程序? 如何使用调试器 *gdb*?
- 2 *assert* 和 *panic_on* 会让程序崩溃. 这有什么作用?
- 3 预编译指令 *define* 和 *include* 以及 *ifdef* 有什么作用?

§4 栈

- 1 什么是栈? 给出熟悉的定义.
- 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).
- 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?
- 4 栈与递归有什么联系?

§5 队列

1 什么是队列？给出熟悉的定义。

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

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:

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*？

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.

1 如何在内存中表示二叉树？

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？

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.

```
1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4
5 const int N = 100003;
6 int prv[N], nxt[N], idx;
7 int n, m;
8
9 void init() {
10     // 0 表示左端点, 1 表示右端点
11     nxt[0] = 1;
12     prv[1] = 0;
13     nxt[1] = -1;
14     prv[0] = -1;
15     idx = 2; // 从2开始
16     for (int i = 2; i <= n; ++i) {
17         prv[i] = nxt[i] = -1;
18     }
19 }
20
21 // 在内存池中编号为 pos 的节点右边插入编号为 x 的节点
22 inline void add_right(int pos, int x) {
23     prv[x] = pos;
24     nxt[x] = nxt[pos];
25     if (nxt[pos] != -1) prv[nxt[pos]] = x;
26     nxt[pos] = x;
```

```

27 }
28
29 // 在内存池中编号为 pos 的节点左边插入编号为 x 的节点
30 inline void add_left(int pos, int x) {
31     nxt[x] = pos;
32     prv[x] = prv[pos];
33     if (prv[pos] != -1) nxt[prv[pos]] = x;
34     prv[pos] = x;
35 }
36
37 // 删除内存池里面编号为 x 的节点
38 inline void remove(int x) {
39     if (prv[x] == -1) return;
40     nxt[prv[x]] = nxt[x];
41     if (nxt[x] != -1) prv[nxt[x]] = prv[x];
42     prv[x] = nxt[x] = -1;
43 }
44
45 // 遍历链表并输出节点的值
46 inline void traverse() {
47     int x = nxt[0];
48     while (x != -1) {
49         cout << x << " ";
50         x = nxt[x];
51     }
52     cout << endl;
53 }
54
55 int main() {
56     scanf("%d", &n);
57     int cmd1, cmd2;
58     init();
59     for (int i = 2; i <= n; ++i) {
60         scanf("%d %d", &cmd1, &cmd2);
61         if (cmd2 == 0) add_left(cmd1, i);
62         else add_right(cmd1, i);
63     }
64     scanf("%d", &m);
65     for (int i = 1; i <= m; ++i) {
66         scanf("%d", &cmd1);
67         remove(cmd1);
68     }
69     traverse();
70     return 0;
71 }

```

7.2. P1996 约瑟夫问题.

```

1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4
5 const int N = 100010;
6 int val[N], prv[N], nxt[N], idx;
7 int n, m;
8
9 void init(int n) {
10     // 0 表示左端点, 1 表示右端点
11     nxt[0] = 1;

```

```
12     prv[1] = 0;
13     idx = 2; // 从2开始
14     for (int i = 1; i <= n; ++i) {
15         val[i] = i;
16         if (i != n) nxt[i] = i + 1;
17         else nxt[i] = 1; // 形成环
18         if (i != 1) prv[i] = i - 1;
19         else prv[i] = n; // 形成环
20     }
21 }
22
23 // 删除内存池里面编号为 x 的节点
24 inline void remove(int x) {
25     nxt[prv[x]] = nxt[x];
26     prv[nxt[x]] = prv[x];
27 }
28
29 int main() {
30     scanf("%d %d", &n, &m);
31     init(n);
32     int current = 1;
33
34     for (int i = 0; i < n; ++i) {
35         // 找到第 m 个要出列的人
36         for (int j = 1; j < m; ++j) {
37             current = nxt[current];
38         }
39         // 输出该人的编号
40         printf("%d ", val[current]);
41         // 删除该人
42         remove(current);
43         // 更新 current 为下一个人的编号
44         current = nxt[current];
45     }
46
47     return 0;
48 }
```

7.3. UVA11988 破碎的键盘.

```
1     #include <cstdio>
2     #include <cstring>
3
4     const int maxn = 100000 + 5;
5     int last, cur, next[maxn];
6     char s[maxn];
7
8     int main() {
9         while (scanf("%s", s + 1) == 1) {
10             int n = strlen(s + 1);
11             last = cur = 0;
12             next[0] = 0;
13             for (int i = 1; i <= n; i++) {
14                 char ch = s[i];
15                 if (ch == '[') {
16                     cur = 0;
17                 } else if (ch == ']') {
18                     cur = last;
19                 } else {
```

```

20         next[i] = next[cur];
21         next[cur] = i;
22         if (cur == last) {
23             last = i;
24         }
25         cur = i;
26     }
27 }
28 for (int i = next[0]; i != 0; i = next[i]) {
29     printf("%c", s[i]);
30 }
31 printf("\n");
32 }
33 return 0;
34 }

```

7.4. UVA12657 盒子排队.

```

1 #include <cstdio>
2 #include <iostream>
3 using namespace std;
4
5 const int maxn = 100005;
6 int nxt[maxn], prv[maxn];
7 int n, m;
8 bool reversed;
9
10 void init(int n) {
11     for (int i = 1; i <= n; ++i) {
12         nxt[i] = i + 1;
13         prv[i] = i - 1;
14     }
15     nxt[0] = 1;
16     prv[n + 1] = n;
17     reversed = false;
18 }
19
20 void remove(int x) { // 删除
21     nxt[prv[x]] = nxt[x];
22     prv[nxt[x]] = prv[x];
23 }
24
25 void insert(int l, int r) {
26     if (nxt[l] == r || l == r) return;
27     remove(l);
28     nxt[prv[r]] = l;
29     prv[l] = prv[r];
30     prv[r] = l;
31     nxt[l] = r;
32 }
33
34 void swp(int l, int r) {
35     int k = nxt[l];
36     insert(l, nxt[r]);
37     insert(r, k);
38 }
39
40 long long sumOddPositions() {
41     long long sum = 0;

```

```

42     int current = nxt[0];
43     int pos = 1;
44     while (current != n + 1 && current != 0) {
45         if (pos % 2 != 0) sum += current;
46         current = nxt[current];
47         pos++;
48     }
49     return sum;
50 }
51
52 int main() {
53     int caseNum = 1;
54     while (scanf("%d %d", &n, &m) != EOF) {
55         init(n);
56         long long ans = 0;
57         for (int i = 1; i <= m; ++i) {
58             int x, l, r;
59             scanf("%d", &x);
60             if (x == 1) {
61                 scanf("%d %d", &l, &r);
62                 if (!reversed) insert(l, r);
63                 else insert(l, nxt[r]);
64             } else if (x == 2) {
65                 scanf("%d %d", &l, &r);
66                 if (!reversed) insert(l, nxt[r]);
67                 else insert(l, r);
68             } else if (x == 3) {
69                 scanf("%d %d", &l, &r);
70                 swp(l, r);
71             } else if (x == 4) {
72                 reversed = !reversed;
73             }
74         }
75         if (reversed) {
76             swap(prv[n + 1], nxt[0]);
77             for (int i = 1; i <= n; ++i) swap(prv[i], nxt[i]);
78         }
79         ans = sumOddPositions();
80         printf("Case %d: %lld\n", caseNum++, ans);
81     }
82     return 0;
83 }

```

7.5. B3614 栈.

```

1     #include <cstdio>
2     #include <cstring>
3     #include <iostream>
4     using namespace std;
5
6     const int maxn = 100000 + 5;
7
8     unsigned long long stack[maxn];
9     int top;
10
11     void push(unsigned long long x) {
12         stack[++top] = x;
13     }
14

```

```
15 void pop() {
16     if (top == 0) {
17         printf("Empty\n");
18     } else {
19         top--;
20     }
21 }
22
23 void query() {
24     if (top == 0) {
25         printf("Anguei!\n");
26     } else {
27         printf("%llu\n", stack[top]);
28     }
29 }
30
31 void size() {
32     printf("%d\n", top);
33 }
34
35 int main() {
36     int T;
37     scanf("%d", &T);
38     while (T--) {
39         int n;
40         scanf("%d", &n);
41         top = 0; // 重置栈顶
42         while (n--) {
43             char operation[10];
44             scanf("%s", operation);
45             if (strcmp(operation, "push") == 0) {
46                 unsigned long long x;
47                 scanf("%llu", &x);
48                 push(x);
49             } else if (strcmp(operation, "pop") == 0) {
50                 pop();
51             } else if (strcmp(operation, "query") == 0) {
52                 query();
53             } else if (strcmp(operation, "size") == 0) {
54                 size();
55             }
56         }
57     }
58     return 0;
59 }
```

7.6. P1739 表达式括号匹配.

```
1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4
5 #define MAX_SIZE 1000 // 定义栈的最大大小
6
7 char stack[MAX_SIZE]; // 使用数组来模拟栈
8 int top = -1; // 栈顶指针
9
10 void push(char c) {
11     if (top < MAX_SIZE - 1) {
```



```
12     stack[++top] = c;
13 }
14 }
15
16 void pop() {
17     if (top >= 0) {
18         top--;
19     }
20 }
21
22 bool isEmpty() {
23     return top == -1;
24 }
25
26 int main() {
27     char input;
28     while (cin >> input && input != '@') {
29         if (input == '(') push(input);
30         if (input == ')') {
31             if (isEmpty()) {
32                 printf("NO\n");
33                 return 0;
34             }
35             pop();
36         }
37     }
38     if (isEmpty()) cout << "YES";
39     else cout << "NO";
40     return 0;
41 }
```

7.7. UVA514 铁轨.

```
1 #include <cstdio>
2 #include <cstring>
3
4 const int MAXN = 1010;
5 int train[MAXN];
6 int stack[MAXN];
7 int top;
8
9 void push(int x) {
10     stack[++top] = x;
11 }
12
13 void pop() {
14     top--;
15 }
16
17 int query() {
18     return stack[top];
19 }
20
21 int main() {
22     int n, A, B, ok;
23     while (scanf("%d", &n), n) {
24         while (1) {
25             scanf("%d", &train[1]);
26             if (train[1] == 0) break;
```

```

27     for (int i = 2; i <= n; i++) {
28         scanf("%d", &train[i]);
29     }
30
31     A = B = ok = 1;
32     top = 0;
33
34     while (B <= n) {
35         if (A == train[B]) {
36             A++;
37             B++;
38         }else if (top > 0 && stack[top] == train[B]) {
39             pop();
40             B++;
41         }else if (A <= n) {
42             push(A++);
43         }else {
44             ok = 0;
45             break;
46         }
47     }
48     printf("%s\n", ok ? "Yes" : "No");
49 }
50 printf("\n");
51 }
52 return 0;
53 }

```

7.8. P1449 后缀表达式.

```

1  #include<iostream>
2  #include<cstdio>
3  using namespace std;
4  long long stk[1000];
5  int main() {
6      long long i=0,now=0;
7      char operators;
8      while((operators=getchar())!='@') {
9          if(operators>='0'&&operators<='9') now*=10,now+=operators-'0';
10         else if(operators=='.') {
11             stk[++i]=now;
12             now=0;
13         } else if(operators=='+') {
14             stk[i-1]=stk[i-1]+stk[i];
15             stk[i]=0;
16             i--;
17         } else if(operators=='-') {
18             stk[i-1]=stk[i-1]-stk[i];
19             stk[i]=0;
20             i--;
21         } else if(operators=='*') {
22             stk[i-1]=stk[i-1]*stk[i];
23             stk[i]=0;
24             i--;
25         } else if(operators=='/') {
26             stk[i-1]=stk[i-1]/stk[i];
27             stk[i]=0;
28             i--;
29         }

```

```
30     }
31     cout<<stk[1];
32     return 0;
33 }
```

7.9. P1175 表达式转换.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5 #include <math.h>
6 #define int long long
7
8 int priority(char ch) {
9     switch(ch) {
10         case '(': case ')': return 0;
11         case '+': case '-': return 1;
12         case '*': case '/': return 2;
13         case '^': return 3;
14     }
15     return -1;
16 }
17
18 int rassoc(char ch){
19     return ch == '^';
20 }
21
22
23 char* suffix(const char* str) {
24     char* s = (char*)malloc(strlen(str) * sizeof(char));
25     char* tmp = (char*)malloc((strlen(str) + 1) * sizeof(char));
26     int s_top = -1, tmp_len = 0;
27
28     for(int i = 0; i < strlen(str); ++i) {
29         if(isdigit(str[i])) {
30             tmp[tmp_len++] = str[i];
31         } else if(str[i] == '(') {
32             s[++s_top] = str[i];
33         } else if(str[i] == ')') {
34             while(s_top >= 0 && s[s_top] != '(') {
35                 tmp[tmp_len++] = s[s_top--];
36             }
37             --s_top;
38         } else {
39             while(s_top >= 0 &&
40                 priority(s[s_top]) >= priority(str[i]) &&
41                 !rassoc(str[i])) {
42                 tmp[tmp_len++] = s[s_top--];
43             }
44             s[++s_top] = str[i];
45         }
46     }
47     while(s_top >= 0) {
48         tmp[tmp_len++] = s[s_top--];
49     }
50     tmp[tmp_len] = '\0';
51     free(s);
52     return tmp;
53 }
```

```

53 }
54
55 int applycalc(char ident, int num1, int num2) {
56     switch(ident) {
57         case '+': return num1 + num2;
58         case '-': return num1 - num2;
59         case '*': return num1 * num2;
60         case '/': return num1 / num2;
61         case '^': return (int) pow(num1, num2);
62     }
63     return -1;
64 }
65
66 void prtsuffix(const char* tmp) {
67     for(int i = 0; i < strlen(tmp); ++i) {
68         printf("%c ", tmp[i]);
69     }
70     printf("\n");
71 }
72
73 void calcPrint(const char* str) {
74     int* ls = (int*)malloc(strlen(str) * sizeof(int));
75     int ls_len = 0;
76     prtsuffix(str);
77
78     for(int i = 0; i < strlen(str); ++i) {
79         if(isdigit(str[i])) {
80             ls[ls_len++] = str[i] - '0';
81         } else {
82             int num1 = ls[--ls_len];
83             int num2 = ls[--ls_len];
84             ls[ls_len++] = applycalc(str[i], num2, num1);
85
86             for(int j = 0; j < ls_len; ++j) {
87                 printf("%d ", ls[j]);
88             }
89             for(int j = i + 1; j < strlen(str); ++j) {
90                 printf("%c ", str[j]);
91             }
92             printf("\n");
93         }
94     }
95     free(ls);
96 }
97
98 signed main() {
99     char str[100];
100     scanf("%s", str);
101     char* psuffix = suffix(str);
102     calcPrint(psuffix);
103     free(psuffix);
104     return 0;
105 }

```

7.10. B3616 队列.

```

1 #include <iostream>
2 #include <string>
3 using namespace std;

```

```
4
5 #define NR_DAT 10003
6
7 struct cqueue {
8     int data[NR_DAT];
9     int front, rear;
10
11     bool init() {
12         front = rear = 0;
13         return true;
14     }
15
16     int size() {
17         return (rear - front + NR_DAT) % NR_DAT;
18     }
19
20     bool isempty() {
21         return (size() == 0);
22     }
23
24     bool push(int e) {
25         if ((rear + 1) % NR_DAT == front) return false; // full!
26         data[rear] = e;
27         rear = (rear + 1) % NR_DAT;
28         return true;
29     }
30
31     bool pop(int &e) {
32         if (front == rear) return false;
33         e = data[front];
34         front = (front + 1) % NR_DAT;
35         return true;
36     }
37
38     int getfront() {
39         if (front == rear) return -1; // indicate empty queue
40         return data[front];
41     }
42 };
43
44 int main() {
45     int n;
46     cin >> n;
47     cqueue q;
48     q.init();
49     for (int i = 0; i < n; ++i) {
50         int op;
51         cin >> op;
52         if (op == 1) {
53             int x;
54             cin >> x;
55             q.push(x);
56         } else if (op == 2) {
57             int x;
58             if (q.pop(x)) {
59                 // cout<<x<<endl;
60             } else {
61                 cout << "ERR_CANNOT_POP" << endl;
62             }
63         } else if (op == 3) {
```

```

64         if (q.empty()) {
65             cout << "ERR_CANNOT_QUERY" << endl;
66         } else {
67             cout << q.getfront() << endl;
68         }
69     } else if (op == 4) {
70         cout << q.size() << endl;
71     }
72 }
73 return 0;
74 }

```

7.11. P1886 滑动窗口.

```

1 #include <iostream>
2 using namespace std;
3
4 const int MAXN = 1000009;
5 int num[MAXN];
6 int n, k;
7
8 struct Deque {
9     int q[MAXN]; // 存储队列元素的数组
10    int head, tail;
11
12    Deque() {
13        head = 0;
14        tail = -1;
15    }
16
17    // 检查队列是否为空
18    bool empty() {
19        return head > tail;
20    }
21
22    // 获取队头元素
23    int front() {
24        return q[head];
25    }
26
27    // 获取队尾元素
28    int back() {
29        return q[tail];
30    }
31
32    // 弹出队头元素
33    void pop_front() {
34        if (!empty()) head++;
35    }
36
37    // 弹出队尾元素
38    void pop_back() {
39        if (!empty()) tail--;
40    }
41
42    // 向队尾添加元素
43    void push_back(int val) {
44        q[++tail] = val;
45    }

```

```

46
47 // 清空队列
48 void clear() {
49     head = 0;
50     tail = -1;
51 }
52 };
53
54 int main() {
55     cin >> n >> k;
56     for (int i = 0; i < n; i++) {
57         cin >> num[i];
58     }
59
60     Deque minDeque, maxDeque;
61
62     // 最小值队列处理
63     int t = 0;
64     for (int i = 0; i < n; i++) {
65         while (!minDeque.empty() && num[minDeque.back()] >= num[i]) minDeque.
        pop_back();
66         minDeque.push_back(i);
67
68         if (i - t >= k && minDeque.front() == t) {
69             t++;
70             minDeque.pop_front();
71         }
72         if (i - t >= k && minDeque.front() != t) t++;
73
74         if (i >= k - 1) cout << num[minDeque.front()] << ' ';
75     }
76     cout << endl;
77
78     // 最大值队列处理
79     t = 0;
80     for (int i = 0; i < n; i++) {
81         while (!maxDeque.empty() && num[maxDeque.back()] <= num[i]) maxDeque.
        pop_back();
82         maxDeque.push_back(i);
83
84         if (i - t >= k && maxDeque.front() == t) {
85             t++;
86             maxDeque.pop_front();
87         }
88         if (i - t >= k && maxDeque.front() != t) t++;
89
90         if (i >= k - 1) cout << num[maxDeque.front()] << ' ';
91     }
92
93     return 0;
94 }

```

7.12. 求 m 区间的最小值.

```

1 #include<cstdio>
2 int n,m,a[2000000],q[2000000],l=1,r=0;
3 int main(){
4     scanf("%d%d",&n,&m);
5     for(int i=1;i<=n;i++) scanf("%d",&a[i]);

```

```

6     for(int i=1;i<=n;i++){
7         printf("%d\n",a[q[l]]);
8         if(i-q[l]+1>m && l<=r) l++;
9         while(a[i]<a[q[r]] && l<=r) r--;
10        q[++r]=i;
11    }
12 }

```

7.13. P3378 堆.

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 const int MAXN = 1e6 + 10;
6
7 struct MinHeap {
8     int size;
9     int heap[MAXN];
10
11     MinHeap() : size(0) {}
12
13     void push_up(int i, int val) {
14         while (i > 1 && val < heap[i / 2]) {
15             heap[i] = heap[i / 2];
16             i /= 2;
17         }
18         heap[i] = val;
19     }
20
21     void push_down(int i, int val) {
22         int ch = i * 2;
23         while (ch <= size) {
24             if (ch < size && heap[ch + 1] < heap[ch]) ch++;
25             if (val <= heap[ch]) break;
26             heap[i] = heap[ch];
27             i = ch;
28             ch *= 2;
29         }
30         heap[i] = val;
31     }
32
33     void insert(int val) {
34         int i = ++size;
35         push_up(i, val);
36     }
37
38     void delete_min() {
39         int i = 1;
40         int val = heap[size--];
41         push_down(i, val);
42     }
43
44     int get_min() const {
45         return heap[1];
46     }
47 };
48
49 int main() {

```



```

50     int n;
51     cin >> n;
52     MinHeap minHeap;
53
54     for (int i = 1; i <= n; i++) {
55         int opt, x;
56         cin >> opt;
57         if (opt == 1) {
58             cin >> x;
59             minHeap.insert(x);
60         } else if (opt == 2) {
61             cout << minHeap.get_min() << endl;
62         } else if (opt == 3) {
63             minHeap.delete_min();
64         }
65     }
66
67     return 0;
68 }

```

7.14. P1168 中位数.

```

1  #include <cstdio>
2
3  const int MAXN = 100100;
4
5  struct Heap {
6      int heap[MAXN];
7      int size;
8      bool (*cmp)(int, int);
9
10     Heap(bool (*cmpFunc)(int, int)) : size(0), cmp(cmpFunc) {}
11
12     void push(int val) {
13         heap[++size] = val;
14         int i = size;
15         while (i > 1 && cmp(heap[i], heap[i / 2])) {
16             swap(heap[i], heap[i / 2]);
17             i /= 2;
18         }
19     }
20
21     void pop() {
22         heap[1] = heap[size--];
23         int i = 1;
24         while (i * 2 <= size) {
25             int j = i * 2;
26             if (j < size && cmp(heap[j + 1], heap[j])) j++;
27             if (cmp(heap[i], heap[j])) break;
28             swap(heap[i], heap[j]);
29             i = j;
30         }
31     }
32
33     int top() {
34         return heap[1];
35     }
36
37     bool empty() {

```

```

38     return size == 0;
39 }
40
41 void swap(int &a, int &b) {
42     int temp = a;
43     a = b;
44     b = temp;
45 }
46 };
47
48 bool cmp1(int a, int b) {
49     return a > b;
50 }
51
52 bool cmp2(int a, int b) {
53     return a < b;
54 }
55
56 int main() {
57     int n, x, y;
58     scanf("%d", &n);
59
60     Heap que1(cmp1); // max-heap
61     Heap que2(cmp2); // min-heap
62
63     scanf("%d", &x);
64     que1.push(x);
65     printf("%d\n", x);
66
67     for (int i = 3; i <= n; i += 2) {
68         scanf("%d%d", &x, &y);
69         if (x > y) que1.swap(x, y);
70         que1.push(x);
71         que2.push(y);
72
73         if (que1.top() > que2.top()) {
74             int a = que1.top(), b = que2.top();
75             que1.pop();
76             que1.push(b);
77             que2.pop();
78             que2.push(a);
79         }
80         printf("%d\n", que1.top());
81     }
82
83     return 0;
84 }

```

7.15. P1631 序列合并.

```

1     #include <iostream>
2     #include <cstdio>
3     #include <vector>
4     using namespace std;
5     const int MAXN = 100100;
6
7     struct Data {
8         int value;
9         int index;

```

```
10     bool operator<(const Data &other) const {
11         return value > other.value;
12     }
13 };
14
15 struct Heap {
16     Data heap[MAXN];
17     int size;
18     bool (*cmp)(Data, Data);
19
20     Heap(bool (*cmpFunc)(Data, Data)) : size(0), cmp(cmpFunc) {}
21
22     void push(Data val) {
23         heap[++size] = val;
24         int i = size;
25         while (i > 1 && cmp(heap[i], heap[i / 2])) {
26             swap(heap[i], heap[i / 2]);
27             i /= 2;
28         }
29     }
30
31     void pop() {
32         heap[1] = heap[size--];
33         int i = 1;
34         while (i * 2 <= size) {
35             int j = i * 2;
36             if (j < size && cmp(heap[j + 1], heap[j])) j++;
37             if (cmp(heap[i], heap[j])) break;
38             swap(heap[i], heap[j]);
39             i = j;
40         }
41     }
42
43     Data top() {
44         return heap[1];
45     }
46
47     bool empty() {
48         return size == 0;
49     }
50
51     void swap(Data &a, Data &b) {
52         Data temp = a;
53         a = b;
54         b = temp;
55     }
56 };
57
58 bool minHeap(Data a, Data b) {
59     return a.value < b.value;
60 }
61
62 int a[MAXN], b[MAXN], t[MAXN];
63 int n;
64 Heap q(minHeap);
65 int main() {
66     // 读取 n 的值
67     cin >> n;
68
69     // 读取数组 a 的值
```

```
70     for (int i = 1; i <= n; ++i) {
71         cin >> a[i];
72     }
73
74     // 读取数组 b 的值
75     for (int i = 1; i <= n; ++i) {
76         cin >> b[i];
77         t[i] = 1;
78         q.push({a[1] + b[i], i});
79     }
80
81     // 输出结果并更新堆
82     while (n--) {
83         Data top = q.top();
84         printf("%d ", top.value);
85         int i = top.index;
86         q.pop();
87         q.push({a[++t[i]] + b[i], i});
88     }
89
90     return 0;
91 }
```