

Lesson1 (数组模拟数据结构)

链表与邻接表 (数组模拟)

单链表&邻接表 (n个链表)：邻接表主要用于存储图与树

- e[N]:值 ne[N]:next
- e与ne数组通过下标关联，空结点下标用-1表示，链表最后一个有效结点ne为-1

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 100010;
6
7  // head表示头结点的下标
8  // e[i]表示结点i的值
9  // ne[i]表示结点i的后一个结点下标
10 // idx表示当前已经使用到了哪个点
11 int head, e[N], ne[N], idx;
12
13 //初始化
14 void init() {
15     head = -1, idx = 0;
16 }
17
18 //将x插到头结点
19 void add_to_head(int x) {
20     e[idx] = x, ne[idx] = head, head = idx++;
21 }
22
23 //将x插到第k个结点之后
24 void add(int k, int x) {
25     e[idx] = x, ne[idx] = ne[k], ne[k] = idx++;
26 }
27
28 //将下标是k的点的后面的点删掉
29 void remove(int k) {
30     ne[k] = ne[ne[k]];
31 }
32
33 int main(void) {
34     int m;
35     scanf("%d", &m);
36
37     init();
38
39     while (m--) {
40         int k, x;
41         char op;
42         // scanf("%c", &op);
43         cin >> op;
44
45         if (op == 'H') {
```

```

46         scanf("%d", &x);
47         add_to_head(x);
48     }
49     else if (op == 'D') {
50         scanf("%d", &k);
51         if (!k) head = ne[head];
52         else remove(k-1);
53     }
54     else {
55         scanf("%d%d", &k, &x);
56         add(k-1, x);
57     }
58 }
59
60 for (int i=head; i!=-1; i=ne[i]) printf("%d ", e[i]);
61
62 return 0;
63 }

```

双链表：优化某些特定问题

- 在单链表基础上增加指向前一个结点的指针
- $l[N], r[N]$
- 不定义头结点与尾结点, 0: head 1: tail

```

1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 100010;
6
7  int l[N], r[N], e[N], idx;
8
9  //初始化双链表
10 void init() {
11     //0表示最左端结点, 1表示最右边结点, 起标志作用, 不能被删除或修改
12     //idx从2开始
13     r[0] = 1, l[1] = 0;
14     idx = 2;
15 }
16
17 //在第k个插入的结点右侧插入一个结点
18 void add(int k, int x) {
19     e[idx] = x;
20     l[idx] = k, r[idx] = r[k];
21     l[r[k]] = idx, r[k] = idx++;
22 }
23
24 //删除第k个插入的结点
25 void remove(int k) {
26     r[l[k]] = r[k];
27     l[r[k]] = l[k];
28 }
29

```

```

30  int main(void) {
31      int m;
32      scanf("%d", &m);
33
34      init();
35
36      while (m--) {
37          int k, x;
38          string op;
39          cin >> op;
40
41          if (op == "L") {
42              scanf("%d", &x);
43              add(0, x);
44          }
45          else if (op == "R") {
46              scanf("%d", &x);
47              add(l[1], x);
48          }
49          else if (op == "D") {
50              scanf("%d", &k);
51              remove(k+1);    //注意从第k个插入的数下标为k+1
52          }
53          else if (op == "IL") {
54              scanf("%d%d", &k, &x);
55              add(l[k+1], x);
56          }
57          else {
58              scanf("%d%d", &k, &x);
59              add(k+1, x);
60          }
61      }
62
63      for (int i=r[0]; i!=1; i=r[i]) printf("%d ", e[i]);
64
65      return 0;
66  }

```

邻接表 (多个单链表)

- head[1]->...
- head[2]->...
- ...
- head[3]->...

栈 (先进后出)

```

1  //stk表示栈，tt表示栈顶
2  int stk[N], tt;
3
4  //插入x
5  stk[++tt] = x;
6

```

```

7 //弹出栈顶
8 tt--;
9
10 //判断是否为空
11 if (tt > 0) not empty
12 else empty
13
14 //栈顶
15 stk[tt];

```

- e1:

```

1 #include <iostream>
2
3 using namespace std;
4
5 const int N = 100010;
6
7 int stk[N], tt;
8
9 int main(void) {
10     int m;
11     scanf("%d", &m);
12
13     while (m--) {
14         int x;
15         string op;
16         cin >> op;
17
18         if (op == "push") {
19             scanf("%d", &x);
20             stk[++tt] = x;
21         }
22         else if (op == "pop") tt--;
23         else if (op == "empty") {
24             if (tt > 0) puts("NO");
25             else puts("YES");
26         }
27         else printf("%d\n", stk[tt]);
28     }
29
30     return 0;
31 }

```

- e2: 表达式求值

```

1 #include <iostream>
2 #include <cstring>
3 #include <unordered_map>
4
5 using namespace std;
6
7 const int N = 100010;
8

```

```

9  char str[N];
10 int num[N];    //数字栈
11 char op[N];    //表达式栈
12 int tt1, tt2;
13
14 void eval() {
15     int b = num[tt1--];
16     int a = num[tt1--];
17     char c = op[tt2--];
18
19     int x = 0;
20     if (c == '+') x = a+b;
21     else if (c == '-') x = a-b;
22     else if (c == '*') x = a*b;
23     else if (c == '/') x = a/b;
24
25     num[++tt1] = x;
26 }
27
28 int main(void) {
29     scanf("%s", str);
30
31     unordered_map<char, int> pr{{'+', 1}, {'-', 1}, {'*', 2}, {'/', 2}};
32
33     for (int i=0; str[i]; i++) {
34         auto c = str[i];
35
36         if (isdigit(c)) {
37             int x = 0, j = i;
38             while (str[j] && isdigit(str[j])) {
39                 x = x*10+str[j++]-'0';
40             }
41
42             num[++tt1] = x;
43             i = j-1;
44         }
45         else if (c == '(') op[++tt2] = c;
46         else if (c == ')') {
47             while (op[tt2] != '(') eval();
48             tt2--;
49         }
50         else {
51             while (tt2 && op[tt2] != '(' && pr[op[tt2]] >= pr[c]) eval();
52             op[++tt2] = c;
53         }
54     }
55
56     while (tt2) eval();
57     printf("%d", num[tt1]);
58
59     return 0;
60 }

```

```

1 //在队尾插入元素，在队头弹出元素
2 int q[N], hh, tt = -1;
3
4 //插入
5 q[++tt] = x;
6
7 //弹出
8 hh++;
9
10 //判断是否为空
11 if (hh <= tt) not empty
12 else empty
13
14 //取队头
15 q[hh]

```

- e1

```

1 #include <iostream>
2
3 using namespace std;
4
5 const int N = 100010;
6
7 int q[N], hh, tt = -1;
8
9 int main(void) {
10     int m;
11     cin >> m;
12
13     while (m--) {
14         int x;
15         string op;
16         cin >> op;
17
18         if (op == "push") {
19             scanf("%d", &x);
20             q[++tt] = x;
21         }
22         else if (op == "pop") hh++;
23         else if (op == "empty") {
24             if (hh <= tt) puts("NO");
25             else puts("YES");
26         }
27         else cout << q[hh] << endl;
28     }
29
30     return 0;
31 }

```

常见模型：在一个序列中，找出每一个数左边离它最近的且满足某种性质（如 最大最小）的数的位置

先想暴力算法，再利用单调性进行优化

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 100010;
6
7  int stk[N], tt;
8
9  int main(void) {
10     int m;
11     scanf("%d", &m);
12
13     while (m--) {
14         int x;
15         scanf("%d", &x);
16
17         while (tt && stk[tt] >= x) tt--;
18         if (tt) printf("%d ", stk[tt]);
19         else printf("-1 ");
20
21         stk[++tt] = x;
22     }
23
24     return 0;
25 }
```

单调队列

经典应用：求滑动窗口中最大值or最小值

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 1000010;
6
7  int n, k;
8  int a[N];
9  //q中存储的是下标
10 int q[N], hh, tt = -1;
11
12 int main(void) {
13     scanf("%d%d", &n, &k);
14
15     for (int i=0; i<n; i++) scanf("%d", &a[i]);
16
17     //打印所有最小值
18     for (int i=0; i<n; i++) {
19         //窗口容量已满，出队
```

```

20     if (hh<=tt && i-k+1>q[hh]) hh++;
21     //单调性优化
22     while (hh<=tt && a[q[tt]]>=a[i]) tt--;
23     //入队
24     q[++tt] = i;
25
26     if (i-k+1 >= 0) printf("%d ", a[q[hh]]);
27 }
28
29 puts("");
30
31 //最大值
32 hh = 0, tt = -1;
33 for (int i=0; i<n; i++) {
34     if (hh<=tt && i-k+1>q[hh]) hh++;
35     while (hh<=tt && a[q[tt]]<=a[i]) tt--;
36     q[++tt] = i;
37
38     if (i-k+1 >= 0) printf("%d ", a[q[hh]]);
39 }
40
41 return 0;
42 }

```

KMP

习惯下标从1开始

1. 暴力算法如何做

```

1  s[N], p[M];
2
3  //朴素做法
4  for (int i=1; i<=n; i++) {
5      bool flag = true;
6      for (int j=1; j<=m; j++) {
7          if (s[i+j-1] != p[j]) {
8              flag = false;
9              break;
10         }
11     }
12
13     if (flag) 匹配成功
14 }

```

2. 如何去优化

- next[i] 表示以i为终点，且后缀和前缀相等的子串的最大长度。

next[i] = j p[1, j] = p[i-j+1, i]

```

1  #include <iostream>
2

```



```

3  using namespace std;
4
5  const int N = 100010, M = 1000010;
6
7  int n, m;
8  char p[N], s[M];
9  int ne[N];
10
11 int main(void) {
12     scanf("%d%s%d%s", &n, p+1, &m, s+1);
13
14     for (int i=2, j=0; i<=n; i++) {
15         while (j && p[i]!=p[j+1]) j = ne[j];
16         if (p[i] == p[j+1]) j++;
17         ne[i] = j;
18     }
19
20     for (int i=1, j=0; i<=m; i++) {
21         while (j && s[i]!=p[j+1]) j = ne[j];
22         if (s[i] == p[j+1]) j++;
23         if (j == n) {
24             printf("%d ", i-n);
25             j = ne[j];
26         }
27     }
28
29     return 0;
30 }
31

```

作用

- 可用于求循环节（字符串哈希无法使用）