Assignment VIII - DSAA(H)

Name: Yuxuan HOU (侯宇轩)

Student ID: 12413104

Date: 2025.11.02

Question 8.1 (0.25 marks)

Draw the following data structures after each of the following operations. Assume that the data structures are initially empty. You don't need to draw pointers for stacks and queues.

1. Consider a stack S and the operations $\textsc{Push}(S, 7)$, $\textsc{Push}(S, 4)$, $\textsc{Push}(S, 5)$, $\textsc{Pop}(S)$, $\textsc{Push}(S, 8)$, $\textsc{Pop}(S)$, $\textsc{Pop}(S)$.
2. Consider a queue Q and the operations $\textsc{Enqueue}(Q, 7)$, $\textsc{Enqueue}(Q, 4)$, $\textsc{Enqueue}(Q, 5)$, $\textsc{Dequeue}(Q)$, $\textsc{Enqueue}(Q, 8)$, $\textsc{Dequeue}(Q)$, $\textsc{Dequeue}(Q)$.
3. Consider a singly-linked list L and the operations $\textsc{List-Prepend}(L, 7)$, $\textsc{List-Prepend}(L, 4)$, $\textsc{List-Prepend}(L, 5)$, $\textsc{List-Delete}(L, 4)$, $\textsc{List-Prepend}(L, 8)$, $\textsc{List-Delete}(L, 7)$, $\textsc{List-Delete}(L, 8)$.

Sol:

1. Stack S (from bottom to top):
   - `7`
   - `7, 4`
   - `7, 4, 5`
   - `7, 4`
   - `7, 4, 8`

- `7, 4`
- `7`

2. Queue Q (from front to back):
   - `7`
   - `7, 4`
   - `7, 4, 5`
   - `4, 5`
   - `4, 5, 8`
   - `5, 8`
   - `8`
3. Singly-linked List L (from head to tail):
   - `7`
   - `4 -> 7`
   - `5 -> 4 -> 7`
   - `5 -> 7`
   - `8 -> 5 -> 7`
   - `8 -> 5`
   - `5`

Question 8.2 (0.5 marks)

Question 8.2 (0.5 marks) Explain how to implement two stacks S_1 and S_2 in one array $A[1:n]$ in such a way that neither stack overflows unless all the n elements of A are full. Present the pseudocodes for operations $\text{PUSHS}_1(A, x)$,$\text{PUSHS}_2(A, x)$,$\text{POPS}_1(A)$, and $\text{POPS}_2(A)$.

Sol: We can implement the 'opposing stacks', i.e., two stacks growing toward each other.

Pseudo Codes:

Algorithm 1: Init(A)

```
1 n ← A.length
2 top1 ← 0
3 top2 ← n + 1
```

Algorithm 2: PushS$_1$(A, x)

```
1 if top1 + 1 = top2 then
2 |   return overflow
3 end
4 top1 ← top1 + 1
5 A[top1] ← x
```

Algorithm 3: PushS$_2$(A, x)

```
1 if top1 + 1 = top2 then
2 |   return overflow
3 end
4 top2 ← top2 − 1
5 A[top2] ← x
```

Algorithm 4: PopS$_1$(A)

```
1 if top1 = 0 then
2 |   return underflow
3 end
4 y ← A[top1]
5 top1 ← top1 − 1
6 return y
```

Algorithm 5: $\text{PopS}_2(A)$

```
1 if top2 = n + 1 then
2 |   return underflow
3 end
4 y ← A[top2]
5 top2 ← top2 + 1
6 return y
```

Question 8.3 (0.25 marks)

Question 8.3 (0.25 marks) Rewrite ENQUEUE and DEQUEUE to detect underflow and overflow of a queue.

Sol:

Algorithm 1: $\text{Init}(A)$

```
1 n ← A.length
2 head ← 1
3 tail ← 1
```

Algorithm 2: $\text{Enqueue}(A, x)$

```
1 if ((tail mod n) + 1) = head then
2 |   return overflow
3 end
4 A[tail] ← x
5 tail ← (tail mod n) + 1
```

Algorithm 3: Dequeue(A)

```
1 if head = tail then
2 |   return underflow
3 end
4 y ← A[head]
5 head ← (head mod n) + 1
6 return y
```

Question 8.4 (0.5 marks)

Question 8.4 (0.5 marks) Show how to implement a Queue using 2 stacks S_1 and S_2. Provide the pseudo-code of the operations ENQUEUE and DEQUEUE. You don't need to check for underflow and overflow. Analyse the runtime of the two operations.

Sol: We can first push the enqueued elements into S_1, the when implementing dequeue, use S_2 to output - pop all the elements in S_1 and push them into S_2, which will reverse the elements to make them satisfy the queue order.

Pseudo code:

Algorithm 1: Enqueue(x)

```
1 PUSH(S_1, x)
```

Algorithm 2: Dequeue( )

```
1 if STACK-EMPTY(S_2) then
2 |   while not STACK-EMPTY(S_1) do
3 |   |   PUSH(S_2, POP(S_1))
4 |   end
5 end
6 return POP(S_2)
```

Runtime:

- `Enqueue(x)` : $\Theta(1)$, for the function only implement `Push`, which is $\Theta(1)$.
- `Dequeue()` : $\Theta(1)$ on average, $\Theta(n)$ at worst. It's obvious that each element will only be pushed into S_1, S_2 and poped from S_1, S_2 once, which guarantees that the average runtime is $\Theta(1)$, while at the worst case, all the elements will be pushed into S_1, and moved to S_2, then poped, which is $\Theta(n)$, but still $\Theta(1)$ on average.

Question 8.5 (0.5 marks)

Question 8.5 (0.5 marks) Implement an Integer Calculator that takes a postfix expression in input using integers as operands and $\{+, -, *\}$ as operators. The algorithm should use a stack. (See Judge)

Sol:

题目

状态	最后递交于	题目
✓ 100 Accepted	8 小时前	**37** Finding Adjacent Value
✓ 100 Accepted	10 小时前	**40** Postfix Expression
✓ 100 Accepted	14 秒前	**29** Jet Bridge Allocation

```cpp
int main(){
    int T = read();
    while(T--){
        int N = read();
        stack < int > cur;
        while(N--){
            char c = getchar();
            int val(0);
            bool isd(false);
            while(!isdigit(c) && c != '+' && c != '-' && c != '*')c = getchar();
            while(isdigit(c))isd = true, val = (val * 10) + int(c - '0'), c = getchar();
            if(isd)cur.push(val);
            else{
                int v1 = cur.top(); cur.pop();
                int v2 = cur.top(); cur.pop();
                cur.push(c == '+' ? (v1 + v2) : (c == '-' ? (v2 - v1) : v1 * v2));
            }
        }printf("%d\n", cur.top());
    }

    // fprintf(stderr, "Time: %.6lf\n", (double)clock() / CLOCKS_PER_SEC);
```

```
    return 0;
}
```

Question 8.6 (1 mark)

Question 8.6 (1 mark) Implement the problems "Finding Adjacent Value" and "Jet Bridge Allocation" on the Judge system.

Sol:


```
struct Node{
    int val;
    int idx;
    Node *left;
    Node *right;
};

int main(){
    int N = read();
    vector < pair < int, int > > A(N + 10, {0, 0});
    for(int i = 1; i <= N; ++i)A[i] = {read(), i};
    sort(next(A.begin()), next(A.begin(), N + 1));

```

```
    vector < Node* > nd(N + 10, nullptr);
    vector < pair < int, int > > res(N + 10, {INT_MAX, -1});
    vector < int > rnk(N + 10, 0);
    for(int i = 1; i <= N; ++i){
        rnk[A[i].second] = i;
        nd[i] = new Node{
            A[i].first,
            A[i].second,
            i == 1 ? nullptr : nd[i - 1],
            nullptr
        };
        if(i != 1)nd[i - 1]->right = nd[i];
    }
    for(int i = N; i > 1; --i){
        if(nd[rnk[i]]->right){
            int cur = nd[rnk[i]]->right->val - A[rnk[i]].first;
            if(cur < res[i].first)res[i] = {cur, nd[rnk[i]]->right->idx};
        }
        if(nd[rnk[i]]->left){
            int cur = A[rnk[i]].first - nd[rnk[i]]->left->val;
            if(cur <= res[i].first)res[i] = {cur, nd[rnk[i]]->left->idx};
        }
        if(nd[rnk[i]]->left)nd[rnk[i]]->left->right = nd[rnk[i]]->right;
        if(nd[rnk[i]]->right)nd[rnk[i]]->right->left = nd[rnk[i]]->left;
    }
    for(int i = 2; i <= N; ++i)printf("%d %d\n", res[i].first, res[i].second);

    // fprintf(stderr, "Time: %.6lf\n", (double)clock() / CLOCKS_PER_SEC);
    return 0;
}

```

```
int main(){

```

```
    int N = read(), M1 = read(), M2 = read();
    vector < int > D1(N + 10, 0), D2(N + 10, 0);
    vector < pair < int, int > > air1, air2;
    for(int i = 1; i <= M1; ++i){
        int s = read(), t = read();
        air1.push_back({s, t});
    }
    for(int i = 1; i <= M2; ++i){
        int s = read(), t = read();
        air2.push_back({s, t});
    }
    sort(air1.begin(), air1.end(), [](const pair < int, int > &a, const pair < int, int > &b)->bool{
        return a.first == b.first ? a.second < b.second : a.first < b.first;
    });
    sort(air2.begin(), air2.end(), [](const pair < int, int > &a, const pair < int, int > &b)->bool{
        return a.first == b.first ? a.second < b.second : a.first < b.first;
    });
    auto cmp = [](const pair < int, int > &a, const pair < int, int > &b)->bool{
        return a.first == b.first ? a.second > b.second : a.first > b.first;
    };
    priority_queue < pair < int, int >, vector < pair < int, int > >, decltype(cmp) > cur(cmp);
    priority_queue < int, vector < int >, greater < int > > fre;
    int lft(0);
    for(auto [s, t] : air1){
        while(!cur.empty() && cur.top().first < s)
            fre.push(cur.top().second), cur.pop();
        int idx(-1);
        if(!fre.empty())idx = fre.top(), fre.pop();
        else idx = ++lft;
        if(idx <= N)++D1[idx];
        cur.push({t, idx});
    }
    while(!cur.empty())cur.pop();
```

```
    while(!fre.empty())fre.pop();
    lft = 0;
    for(auto [s, t] : air2){
        while(!cur.empty() && cur.top().first < s)
            fre.push(cur.top().second), cur.pop();
        int idx(-1);
        if(!fre.empty())idx = fre.top(), fre.pop();
        else idx = ++lft;
        if(idx <= N)++D2[idx];
        cur.push({t, idx});
    }

    int res(0);
    for(int i = 1; i <= N; ++i)D1[i] += D1[i - 1], D2[i] += D2[i - 1];
    for(int i = 0; i <= N; ++i)res = max(res, D1[i] + D2[N - i]);
    printf("%d\n", res);


    // fprintf(stderr, "Time: %.6lf\n", (double)clock() /
CLOCKS_PER_SEC);
    return 0;
}
```