

暨南大学本科实验报告专用纸(附页)

实现基于循环增长数组的 deque

课程名称 数据结构 成绩评定

实验项目名称 实现基于循环增长数组的 deque 指导老师 干晓聪

实验项目编号 04 实验项目类型 设计性 实验地点 数学系机房

学生姓名 郭彦培 学号 2022101149

学院 信息科学技术学院 系 数学系 专业 信息管理与信息系统

实验时间 2024 年 6 月 13 日上午 ~ 2024 年 7 月 13 日中午

1. 实验目的

实现基于循环增长数组的双向队列，保证在某一段重复添加弹出后实际内存占用规模符合理论占用，不会出现方向性泄漏。

2. 实验环境

计算机：PC X64

操作系统：Windows + Ubuntu20.0LTS

编程语言：C++：GCC std20

IDE：Visual Studio Code

3. 程序原理

在类 `deque` 中维护了两个指针与两个循环增长数组。如果某一端的长度偏差值大于 1，即某侧数据长小于同侧空白区域长度，则触发再分配。

可以证明，在数据规模极大时，再分配的均摊复杂度为 $O(1)$

4. 程序代码

4.1. deque.h

```
1 // #define _PRIVATE_DEBUG
2 #ifndef PRVLIBCPP_DEQUE_HPP
3 #define PRVLIBCPP_DEQUE_HPP
4
5 #include <map>
6 #include <vector>
7
8 #ifdef _PRIVATE_DEBUG
9 #include <iostream>
10 #endif
11
12 namespace myDS
13 {
14     template<typename VALUE_TYPE>
15     class deque{
16     protected:
17
18     private:
19         using coddinate = std::pair<std::int32_t, std::int32_t>;
20
21         // < L : 0 , R : 1 >
22         std::vector<std::vector<VALUE_TYPE>> _indexs;
23
24         std::int32_t _size = 0;
25         std::int32_t _L = 1;
26         std::int32_t _R = -1;
27
28         VALUE_TYPE & get(coddinate p) {
29             return _indexs[p.first][p.second];
30         }
31
32         coddinate index2cod(std::int32_t p) {
33             if(p+_L > 0) return coddinate(1, p+_L-1);
34             else return coddinate(0, -p-_L);
35         }
36
37         void _reDistribute() {
38             if(_L * _R <= 0) return;
39             if(abs(_L - _R) + 1 < std::min(abs(_L), abs(_R))) {
40                 if(_L > 0) { // < --- 0 : 0 --- L -- R --- >
41                     std::vector<VALUE_TYPE> N;
42                     for(int i = _L-1; i <= _R; i++)
43                         N.push_back(_indexs[1][i]);
44                     _indexs[1] = N;
45                     _L = 1;
46                     _R = N.size() - 1;
47                 } else { // < --- L(<0) -- R(<0) --- 0 : 0 --- >
48                     std::vector<VALUE_TYPE> N;
```

暨南大学本科实验报告专用纸(附页)

```
48         for(int i = -_R-1; i <= -_L; i++)
49             N.push_back(_indexs[0][i]);
50             _indexs[0] = N;
51             _L = -N.size()+1;
52             _R = -1;
53         }
54     } else return;
55 }
56 public:
57     deque(){
58         _indexs.push_back(std::vector<VALUE_TYPE>());
59         _indexs.push_back(std::vector<VALUE_TYPE>());
60     }
61
62     void push_back(VALUE_TYPE t) {
63         _R++;
64         if(_R >= 0) {
65             _indexs[1].push_back(t);
66         } else {
67             _indexs[0][_R-1] = t;
68             _reDistribute();
69         }
70     }
71
72     void push_front(VALUE_TYPE t) {
73         _L--;
74         if(_L <= 0) {
75             _indexs[0].push_back(t);
76         } else {
77             _indexs[1][_L-1] = t;
78             _reDistribute();
79         }
80     }
81
82     VALUE_TYPE pop_back() {
83         if(!this->size()) throw std::out_of_range("Pop from empty
84         deque");
85         VALUE_TYPE t;
86         if(_R >= 0) {
87             t = _indexs[1].back();
88             _indexs[1].pop_back();
89             _R--;
90         } else {
91             t = _indexs[0][_R-1];
92             _R--;
93             _reDistribute();
94         }
95         return t;
96     }
```

暨南大学本科实验报告专用纸(附页)

```
97     VALUE_TYPE pop_front() {
98         if(!this->size()) throw std::out_of_range("Pop from empty
deque");
99         VALUE_TYPE t;
100         if(_L <= 0) {
101             t = _indexs[0].back();
102             _indexs[0].pop_back();
103             _L ++;
104         } else {
105             t = _indexs[1][_L-1];
106             _L ++;
107             _reDistribute();
108         }
109         return t;
110     }
111
112     void clear() {
113         _indexs[0].clear();
114         _indexs[1].clear();
115         _L = 1;
116         _R = -1;
117     }
118
119     std::int32_t size() {
120         return _R - _L + 2;
121     }
122
123     #ifdef _PRIVATE_DEBUG
124     void innerPrint() {
125         std::cout << "L : " << _L << " R : " << _R << "\n";
126         std::cout << "L : ";
127         for(auto x:_indexs[0]) std::cout << x << " ";
128         std::cout << "\n";
129         std::cout << "R : ";
130         for(auto x:_indexs[1]) std::cout << x << " ";
131         std::cout << "\n";
132     }
133     #endif
134
135     // myDS::deque<VALUE_TYPE>::_iterator begin() { }
136
137     // myDS::deque<VALUE_TYPE>::_iterator rbegin() { }
138
139     // myDS::deque<VALUE_TYPE>::_iterator end() { }
140
141     // myDS::deque<VALUE_TYPE>::_iterator rend() { }
142
143     // myDS::deque<VALUE_TYPE>::_iterator get(std::int32_t p) { }
144
145     VALUE_TYPE & operator[](std::int32_t p) {
146         return get(index2cod(p));
147     }
```

暨南大学本科实验报告专用纸(附页)

```
148     };
149 }
150 #endif
```

4.2. _PRIV_TEST.cpp

```
1  #define DS_TOBE_TEST deque
2
3  #define _PRIVATE_DEBUG
4  // #define __DETIL_DEBUG_OUTPUT
5
6  #include "Dev\04\deque.h"
7
8  #include <time.h>
9  #include <iostream>
10 #include <math.h>
11 #include <vector>
12
13 using namespace std;
14
15 using TBT = myDS::deque<int>;
16
17 void accuracyTest() { //结构正确性测试
18
19     TBT tc = TBT();
20     for(;;)
21     {
22         string op;
23         cout << ">>>";
24         cin >> op;
25         if(op == "clr") { //清空
26             tc.clear();
27         } else if(op == "q") //退出测试
28         {
29             return;
30         } else if(op == "pb") //push_back
31         {
32             int c;
33             cin >> c;
34             tc.push_back(c);
35         } else if(op == "pf") //push_front
36         {
37             int c;
38             cin >> c;
39             tc.push_front(c);
40         } else if(op == "ob") //pop_back
41         {
42             cout << tc.pop_back() << "\n";
43         } else if(op == "of") //pop_front
44         {
```

暨南大学本科实验报告专用纸(附页)

```
45         cout << tc.pop_front() << "\n";
46     } else if(op == "at")//随机访问
47     {
48         int p;
49         cin >> p;
50         cout << tc[p] << "\n";
51     } else if(op == "at")//随机访问
52     {
53         int p;
54         cin >> p;
55         cout << tc[p] << "\n";
56     } else if(op == "of")//pop_front
57     {
58
59     } else if(op == "at")//随机访问
60     {
61         int p;
62         cin >> p;
63         cout << tc[p] << "\n";
64     // } else if(op == "delEL")//删除所有等于某值元素
65     // {
66     //     int p;
67     //     cin >> p;
68     //     cout << tc.erase(p) << "\n";
69     // } else if(op == "delPS")//删除某位置上的元素
70     // {
71     //     int p;
72     //     cin >> p;
73     //     cout << tc.erase(tc.get(p)) << "\n";
74     } else if(op == "iterF") //正序遍历
75     {
76         tc.innerPrint();
77         cout << "Iter with index:\n";
78         for(int i = 0;i < tc.size();i ++) cout << tc[i] << " ";cout
79         << "\n";
80         // cout << "Iter with begin end\n";
81         // for(auto x = tc.begin();x != tc.end();x ++) cout << (*x)
82         << " ";cout << "\n";
83         // cout << "Iter with AUTO&&\n";
84         // for(auto x:tc) cout << x << " ";cout << "\n";
85     } else if(op == "iterB") //倒序遍历
86     {
87         tc.innerPrint();
88         cout << "Iter with index:\n";
89         for(int i = 0;i < tc.size();i ++) cout << tc[tc.size()-1-i]
90         << " ";cout << "\n";
91         // cout << "Iter with begin end\n";
92         // for(auto x = tc.rbegin();x != tc.rend();x ++) cout <<
93         (*x) << " ";cout << "\n";
```

暨南大学本科实验报告专用纸(附页)

```
90         // cout << "Iter with AUTO&&\n";.\n";
91     } else if(op == "mv")//单点修改
92     {
93         int p;
94         cin >> p;
95         int tr;
96         cin >> tr;
97         tc[p] = tr;
98     } else if(op == "")
99     {
100
101     } else {
102         op.clear();
103     }
104 }
105 }
106
107
108 void memLeakTest1() {//内存泄漏测试
109     TBT tc = TBT();
110     for(;;){
111         tc.push_back(1);
112         tc.push_back(1);
113         tc.push_back(1);
114         tc.push_back(1);
115         tc.clear();
116     }
117 }
118
119 void memLeakTest2() {//内存泄漏测试
120     TBT tc = TBT();
121     for(;;){
122         tc.push_back(1);
123         tc.pop_frount();
124     }
125 }
126
127 void speedTest()
128 {
129     TBT tc = TBT();
130     int begin = clock();
131     int N = 1e8;
132     for(int i = 0;i < sqrt(N/2);i ++){
133     {
134         for(int j = 0;j < sqrt(N/2);j ++){
135             {
136                 tc.push_back(i);
137             }
138             for(int j = 0;j < sqrt(N/2);j ++){
139                 {
140                     tc.pop_frount();
```

暨南大学本科实验报告专用纸(附页)

```
141     }
142 }
143 cout << "myDS::deque push_back then pop_frount sqrt(5000000)
elements for sqrt(5000000) times cost:" << clock() - begin << "ms\n";
144
145 std::vector<int> tmp;
146 begin = clock();
147 for(int i = 0;i < N;i ++)
148 {
149     tmp.push_back(i);
150 }
151 cout << "std::vector push_back 10000000 elements cost:" << clock() -
begin << "ms\n";
152     system("pause");
153
154 }
155
156 signed main()
157 {
158     // accuracyTest();
159     // memLeakTest1();
160     // memLeakTest2();
161     speedTest();
162 }
```

5. 测试数据与运行结果

运行上述 `_PRIV_TEST.cpp` 测试代码中的正确性测试模块，得到以下内容：

```
pb 2
pb 3
pb 4
pf 1
pf 0
iterF
```

```
pb 5
pb 6
of
of
of
iterF
of
iterF
of
```


暨南大学本科实验报告专用纸(附页)

```
iterF

>>>pb 2
>>>pb 3
>>>pb 4
>>>pf 1
>>>pf 0
>>>iterF
L : -1 R : 2
L : 1 0
R : 2 3 4
Iter with index:
0 1 2 3 4
>>>
pb 5
>>>pb 6
>>>of
0
>>>of
1
>>>of
2
>>>iterF
L : 2 R : 4
L :
R : 2 3 4 5 6
Iter with index:
3 4 5 6
>>>of
3
>>>iterF
L : 1 R : 2
L :
R : 4 5 6
Iter with index:
4 5 6
>>>of
4
>>>iterF
L : 1 R : 1
L :
R : 5 6
Iter with index:
5 6
```

暨南大学本科实验报告专用纸(附页)

可以看出，代码运行结果与预期相符，可以认为代码正确性无误。

运行 `_PRIV_TEST.cpp` 中的内存测试模块与单向插入测试模块，在保持 CPU 高占用率运行一段时间后内存变化符合预期，可以认为代码内存安全性良好。

名称	状态	25% CPU	45% 内存
 <code>_PRIV_TEST.exe</code>		20.7%	0.6 MB

运行 `_PRIV_TEST.cpp` 中的性能测试模块，结果为

```
myDS::deque push_back then pop_frount sqrt(5000000) elements for
sqrt(5000000) times cost:3964ms
std::vector push_back 10000000 elements cost:1528ms
```

可以认为在每轮中单向插入的复杂度符合预期。