专题三：基础数据结构

天津大学

2023 年 7 月 4 日

目录

目录

栈

Description

栈是一种只能从表的一端存取数据且遵循“先进后出”原则的线性存储结构，也被称为后进先出（last in first out）表，简称 LIFO 表。

Operation

- $stack < int >\ s$：申请一个 int 类型的栈
- $s.push(x)$：元素 x 进栈
- $s.top()$：返回栈顶元素
- $s.pop()$：删除栈顶元素
- $s.empty()$：判断栈是否为空
- $s.size()$：返回栈内元素个数

队列

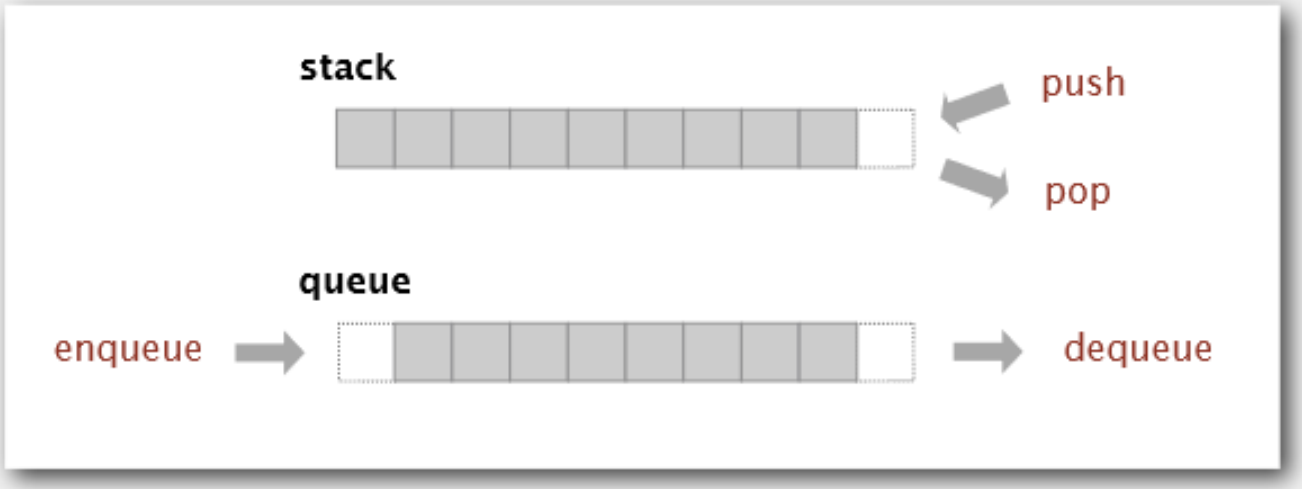
Description

队列是一种从表的一端进，另一端出的，且遵循“先进先出”原则的线性存储结构，也被称为先进先出（First In First Out）表，简称 FIFO 表。

Operation

- $queue < int > \ q$：申请一个 int 类型的队列
- $q.push(x)$：元素 x 从队尾进入队列
- $q.front()$：返回队首元素
- $q.pop()$：删除队首元素
- $q.empty()$：判断队列是否为空
- $q.size()$：返回队列内元素个数

栈与队列对比



不同点：
栈：先进后出、一端开口称为顶；队列：先进先出、两端开口称为头尾
相同点：
都是线性存储结构、操作相似、通常情况下使用数组实现该结构能够使得其更加灵活

单调队列与单调栈

单调栈

在任何时刻，都有栈中的元素是有序的栈称为单调栈。

单调队列

在任何时刻，都有队列中的元素是有序的队列称为单调队列。

栈可以用来解决哪些类型的问题？

经典应用

1. 递归
2. 火车进站
3. 表达式求值
4. 括号匹配
5. ……

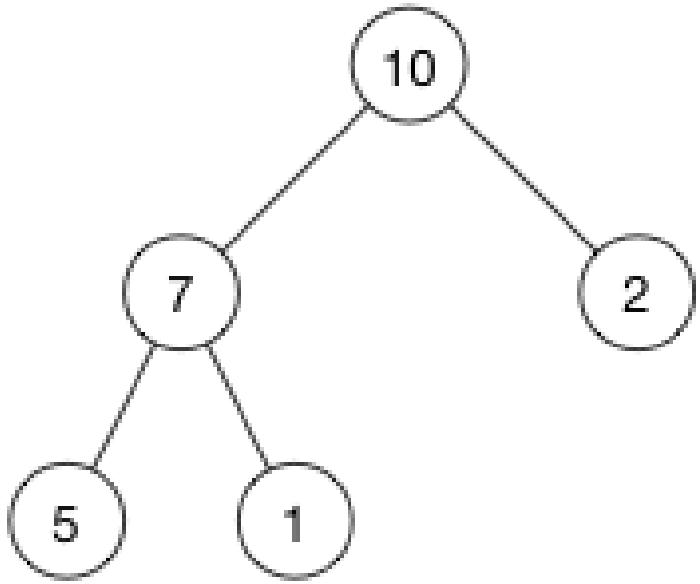
题目

1. luogu P1427
2. luogu P1165

目录

什么是堆？

堆是一棵用数组实现的、满足某性质的**完全**二叉树。
大根堆：根结点最大，满足每个节点都比儿子节点大
小根堆：根结点最小，满足每个节点都比儿子节点小
注：这里所指的大，并不仅仅是指数值，可以根据任何属性去比较



priority_queue: 优先队列

1. 优先队列就是系统封装好的大根堆
2. 通常情况下，优先队列能够满足我们的需求
3. 可以通过加负号来实现一个伪小根堆
4. 其他情况需要重载 $<$ 运算符

Operation

- $priority\_queue < int > q$：申请一个 int 类型的大根堆
- $q.push(x)$：元素 x 进入优先队列
- $q.top()$：返回堆顶元素
- $q.pop()$：删除堆顶元素
- $q.empty()$：判断是否为空
- $q.size()$：返回堆内元素个数

重载运算符

Code

```
struct Node{
    int x,y;
    bool operator < (const Node &a)const{
    //实现了一个按照x排序的小根堆
        return x<a.x;
    }
}
```

目录

STL 常用容器

1. vector
2. set
3. map
4. queue
5. priority_queue

Vector：向量

Description

vector 相当于一个变长数组，可以随机存取元素（使用索引），数组尾部添加或移除元素非常快速，但在中间或头部安插元素比较费时。

Operation

- $vector < int > v$：申请一个 int 类型的动态数组
- $v[i]$：访问数组中的 i 位置的元素
- $v.push\_back(x)$：在数组尾部插入数 x
- $v.size()$：统计数组中元素的个数
- $v.clear()$：清空数组

Set：集合

Description

就是类似数学上的集合

Operation

- $set < int > s$：申请一个 int 类型的集合
- $s.insert(x)$：在集合内插入 x
- $s.find(x)$：在集合内寻找 x
- $s.count(x)$：统计 x 的个数；由于元素唯一，所以只有 0 或 1
- $s.erase(x)$：查找 x 并将其删除
- $s.clear()$：清空集合内元素
- $s.size()$：集合内元素个数
- $s.lower\_bound(x)/s.upper\_bound(x)$：二分查找

map：映射

Description

map 是一种可以将任意类型映射到任何类型的容易 (key->value) 对。

Operation

- $map < string, int > mp$：申请一个 string 类型到 int 类型的映射
- $mp[key]$：返回键值 key 对应的 value，默认返回 0
- $mp.find(key)$：查找键值 key 是否在 map 中存在
- $mp.size()$：返回 map 的大小
- $mp.clear()$：清空 map 中的元素

queue：队列

Description

queue 是一种从表的一端进，另一端出的，且遵循“先先出”原则的线性存储结构，也被称为先进先出（First In First Out）表，简称 FIFO 表。

Operation

- $queue < int > q$：申请一个 int 类型的队列
- $q.push(x)$：元素 x 进入队列
- $q.front()$：返回队首元素
- $q.pop()$：删除队首元素
- $q.empty()$：判断队列是否为空

priority_queue：优先队列

Description

priority_queue 是一种可以在 logn 时间内得出元素最大值的容器

Operation

- $queue < int > q$：申请一个 int 类型的队列
- $q.push(x)$：元素 x 进入优先队列
- $q.top()$：返回堆顶元素
- $q.pop()$：删除队首元素
- $q.empty()$：判断队列是否为空

常用算法

需要 <algorithm> 算法库。

sort

sort(a+l+1,a+r+1,cmp) 可以对数组 a 的 [l,r] 按照 cmp 规则进行排序。

unique

unique(a+1,a+1+n) 可以实现对有序数组 a 去重，返回去重后的尾元素的下一个地址。

nth_element

nth_element(a+1,a+1+k,a+1+n) 以数组 a[1,n] 中第 k 小的数将数组 a 进行排序，排序后 k 位置后的数全部比 a[k] 大，k 位置前的数全部比 a[k] 小

常用算法

lower_bound

lower_bound(a+1,a+1+n,x) 可以返回有序数组 a 中第一个大于等于 x 的位置。

upper_bound

upper_bound(a+1,a+1+n,x) 可以返回有序数组 a 中第一个大于 x 的位置。

目录

并查集

什么是并查集

并查集（Union-Find）就是用来对集合进行合并（Union）与查询（Find）操作的一种数据结构。
合并就是将两个不相交的集合合并成一个集合。
查询就是查询两个元素是否属于同一集合。

并查集

并查集的结构其实是树形结构

并查集

初始化-查询-合并

```
1   void init(int n)
2   {
3       for (int i=1;i<=n;i++)
4           fa[i] = i;
5   }
6   int find(int x)
7   {
8       if (fa[x] == x)
9       return x;
10      else
11      return find(fa[x]);
12  }
13  void merge(int i,int j)
14  {
15      fa[find(i)] = find(j)
16  }
```

并查集

路径压缩

```
17      int find(int x)
18      {
19          return x == fa[x] ? x : (fa[x] = find(fa[x]));
20      }
```

带权并查集

思考问题

怎么带权？
怎么合并？
怎么查询？
怎么路径压缩？

以一道题为例子

HDU 3038

带权并查集

查找 + 路径压缩

递归向上查找父亲，向下计算权值

代码

```
21      int find(int x)
22      {
23          if (x == fa[x])
24              return x;
25          else
26          {
27              int tmp = fa[x];
28              fa[x] = find(fa[x]);
29              val[x] += val[tmp]
30              return fa[x];
31          }
32      }
```

带权并查集

合并

```
33      //x,y之间建立权为d的父亲边
34      fx = find(x);
35      fy = find(y);
36      if (fx != fy)
37      {
38          fa[fx] = fy;
39          val[fx] = -val[x] + val[y] + d;
40      }
41      //验证xy之间的权值是否为d
42      else
43      {
44          if (val[x] - val[y] != d)
45          {
46              cnt++;
47          }
48      }
```

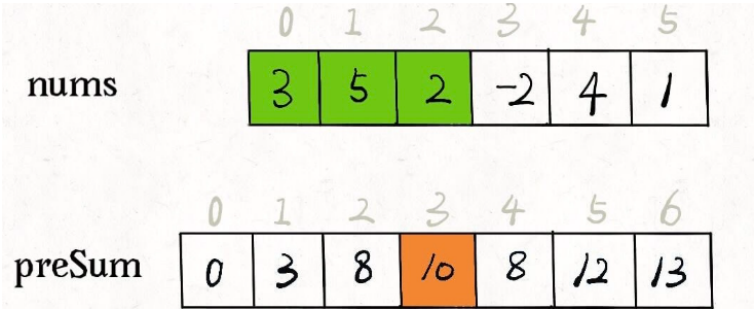
目录

差分前缀和

前缀和

差分与前缀和是针对数组的算法，而数组又分为一维与多维，因此差分与前缀和也包括一维与多维两种，不过二者原理相同，形变神不变。

$$\text{preSum[i]} = \text{preSum[i-1]} + \text{nums[i]};$$



差分前缀和

问题

问题：维护一个数组，q 次操作，每次将 [l,r] 区间内的数同时加上 x，问最后这个数组每个数的值是多少

解决

维护一个差分数组 c，对于每次操作，令 c[l]+=x,c[r+1]-=x，操作全部完成后做一次前缀和即可

思考

传统的差分有什么局限呢

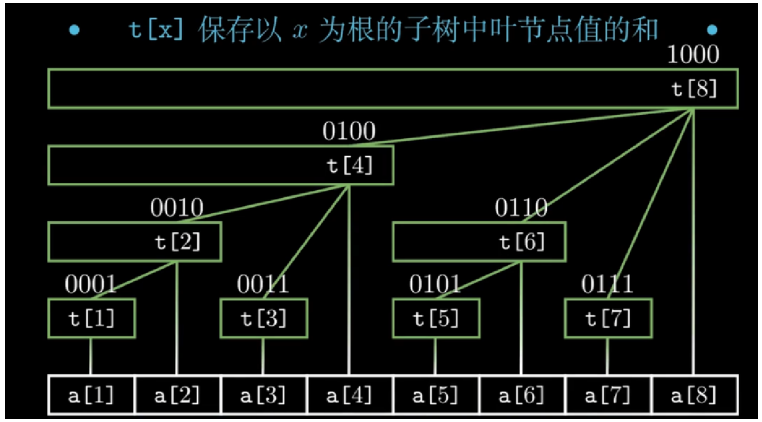
目录

树状数组

树状数组有什么用？

树状数组能在 logn 的时间内求出数组内任意一段区间的值

树状数组可视化（理解思想，背板即可！）



思考

可以用树状数组求逆序对的值吗？

目录

st 表

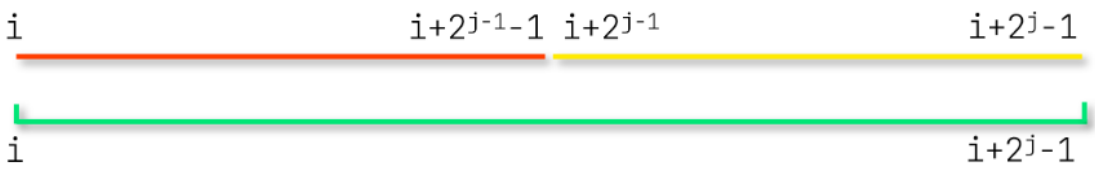
st 表有什么用？

st 表能在 logn 的时间内求出数组内任意一段区间的最大值

如何实现？

维护一个数组 dp[i][j]，代表从 i 开始 j 长度的区间的最大值

st 表可视化
本质思想是倍增！



目录

Thank you! Any Question?