数据结构初步

BUAA 刘子渊

一些闲话

- 计算的本质：对于信息的处理
- 处理信息的基础：存储
  - 举个例子：多项式乘法。设有两个多项式 P_1 和 P_2，计算它们的乘积 P = P_1 * P_2
  - 一种表示方法：(A_1 * x^2 + B_1 * x + C_1) * (A_2 * x^2 + B_2 * x + C_2) = ...
  - 另一种表示方法：任取三个 x 的不同值 x_1，x_2，x_3，将 P_1、P_2 在这三个点处的值相乘便得到 P 在这三个点处的值

一些闲话

- 得到 P 在这三个点处的取值，以及知道了 P 的最高项次数，便相当于得到了 P 的全部信息。（其具体机理需要较高数学知识，详细见拉格朗日插值法。）
- 让我们来比较一下两种表示方法下多项式乘法的复杂度。
- 第一种需要分别相乘，复杂度为 O(n^2)，第二种则只需要对应点值处相乘，复杂度为 O(n)，其中 n 为多项式的阶数。
- 但是注意到，在一个点处取值的复杂度是 O(n)，而总共有 O(n) 个点，于是转换表示方式的复杂度是 O(n^2)，并没有什么区别。

一些闲话

- 而在实际应用中，FFT，即快速傅里叶变换，是通过选点的特殊性（所谓复单位根），将转换的复杂度降到了 O(nlogn)。
- 在这个例子中，我们可以看到，正是由于存储方式的不同，导致了效率上的不同，由此可见数据存储方式的重要性。

- 在算法中数据的存储：数据结构

对于闲话的闲话

- 按照我的预期，在“一些闲话”里，你们应该有很多东西看不懂：拉格朗日插值法（大学知识）、快速傅里叶变换（大学知识）、O 记号（涉及大学知识）、log 记号（高一知识，如果预习的话应该能看懂，我记得是必修一？）。
- 不过这都没什么关系，你们毕竟还年轻 233
- 牛奶会有的，面包会有的！

对于闲话的闲话

- 这些知识你们以后会懂的。
- 但是数学很重要，很重要，很重要。
- 趁着还年轻，赶紧多看看数学书，增长增长姿势水平。
- 记住：数学以后在你的生产生活中无处不在。

练习的重要性

- 算法这个东西，本质上是一门数学，里面充斥着各种各样的技巧
- 这意味着，没有一个通用的思路，必须通过大量的练习来体悟其中的技巧
- 可以说，这是一门手艺活
- 我个人的程序设计学习可以说是一个完全的实践派

练习的重要性

- 没完整的看过什么语言书，因此像许多 C++ 的语言特性我到现在都不会
- 所有的能力都是通过做题做出来的
- 上了大学之后，一些术语可能没有听过，但是一点就会
- 一些注意事项也没什么人给我讲过，但就是知道，遇到什么问题该怎么解决
- 实践出真知

练习的重要性

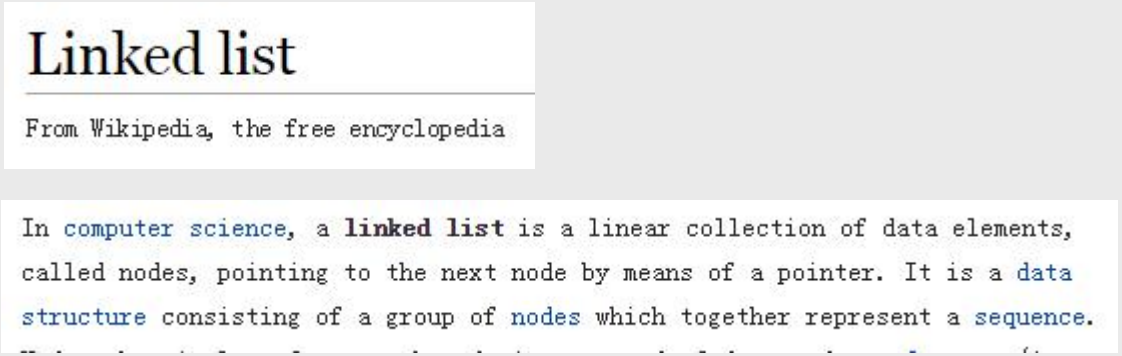
- 练习多了少了差异还是很明显的
- 比如大学先修你拿了 A+，别人拿了 B
- 不能说水平和练习量有定量关系
- 但毫无疑问，肯定成正相关
- 凡事靠自己，尤其是学习

一些数据结构

- 基础的数据结构：
  - 链表
  - 队列、栈
  - 堆...
- 稍微进阶一些的数据结构：
  - 树
  - 图
  - 线段树、树状数组
  - 并查集
  - 平衡树...

链表

链表



Linked list

From Wikipedia, the free encyclopedia

In computer science, a **linked list** is a linear collection of data elements, called nodes, pointing to the next node by means of a pointer. It is a data structure consisting of a group of nodes which together represent a sequence.

请同学们以后多多使用 Wikipedia

链表

- 基础写法参见 Linked_list.cpp
- 所谓基础写法，就意味着还有许许多多其他的写法
- 什么单向、双向、带头指针的单循环...

链表变种

- 单向链表：就是“基础写法”，只能找后继
- 双向链表：增加一个指针，指向前驱
- 循环链表：尾的后继是链表头
- 请意会一下“带尾指针的单循环链表”

一些注意事项

- 烦请写链表的时候一定要画图，要画图，要画图！
- 单链表涉及至少两个指针域，双链表至少四个
- 稍一不注意就容易错，最简单的方法就是画图
- 改掉一个关系在图上把它划去，减少错误的可能

用数组实现的链表

- 对于链表的一项简单优化就是用数组实现
- 数组的下标，本质上就是一种地址
- 优点
- 1. 在最开始申请一个大数组，可以减少 malloc 的次数，提升速度
- 2. 不用操作指针，使用方便

队列、栈

队列、栈

- 队列（queue）和栈（stack）经常放在一起来讲是因为他们描述两种相反的逻辑
- FIFO(First In First Out) 和 LIFO(Last In First Out)
- 先进先出和后进先出

队列、栈

Queue (abstract data type)

From Wikipedia, the free encyclopedia

In computer science, a **queue** (/ˈkjuː/ KYOO) is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position, known as *enqueue*, and removal of entities from the front terminal position, known as *dequeue*. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO

Stack (abstract data type)

From Wikipedia, the free encyclopedia

In computer science, a **stack** is an abstract data type that serves as a collection of elements, with two principal operations: *push*, which adds an element to the collection, and *pop*, which removes the most recently added element that was not yet removed. The order in which elements come off a stack gives rise to its alternative name, **LIFO** (for **last in, first out**). Additionally, a *peek* operation may give access to

队列、栈

- 具体实现见 Queue.cpp 和 Stack.cpp
- 无论是队列还是栈，都有三种方法任君选择
- 当然我个人是比较推荐使用 <deque> 双端队列标准库来实现的
- 方便

队列、栈

- 当然如果你要自己实现也没什么关系
- 但是注意一点，队列按我的实现方法很可能会出现队列空间够用，但是队列不够用的状况
- 在这种情况下，我们使用一种叫做“循环队列”的技术来解决这个问题
- 即使用 head 和 tail 的时候均要加上 %L，其中 L 是队列大小（其中 % 表示取模运算）

队列、栈

- 队列和栈由于表示的东西太过基础，所以单独使用很难有大用
- 一般是作为算法的数据结构（比如图论中的 BFS，表达式求值）
- 或者是它们存储的东西本身具有一些特殊的性质（比如单调栈，单调队列）

POJ 2796 Feel Good

- 题目大意：给一个长度为 N <= 100,000 的序列，从中选取一个连续子序列，使得该序列的最小值乘以序列和最大。每个元素的范围为 0 到 1e6。

POJ 2796 Feel Good

- 由于元素都是正的，于是区间的含于可以推出区间和的小于
- 枚举每个元素，找到以它为最小值，向左向右扩展的最大区间，暴力复杂度 O(N^2)，不可承受
- 如何快速枚举向左向右扩展的最大区间呢？
- 考虑擂台赛

POJ 2796 Feel Good

- 从左向右进行擂台赛
- 每上来一个数，他会把在擂台上的比他大的数都打下去
- 这样每个数一旦被打下去，说明他的最长向左扩展区间就是打败他的数的下标减 1
- 如果盲目的扫一遍，时间复杂度还是 O(N^2)
- 但是注意到，一旦一个数上了擂台，说明在擂台上他是最大的数

POJ 2796 Feel Good

- 应用归纳法可以得到擂台上呈现如下状态：下标越小（即上擂台的时间越早），那么值也就越小
- 因此擂台上呈现出一种单调性
- 于是只要按照下标从小到大排成一列，新来的数与最末的数比较，如果小于等于，则进入擂台；否则，最末的数被踢下擂台
- 可以看到这个擂台符合 LIFO 规则，所以是个栈

POJ 2796 Feel Good

- 又由于这个栈具有单调性，所以叫做单调栈
- 代码详细见 POJ2796 Feel Good.cpp
- 这个代码是我优化了好几遍的
- 没加读入优化 922ms，加了 422ms，deque 改手写 250ms，改掉一个 long long 204ms，用一个变量存储中间结果 157ms
- 这个例子可以非常生动的告诉你们什么东西影响效率

堆

堆

Heap (data structure)

From Wikipedia, the free encyclopedia

In computer science, a **heap** is a specialized tree-based data structure that satisfies the *heap property:* If A is a parent node of B then the *key* (the value) of node A is ordered with respect to the key of node B with the same ordering applying across the heap. A heap can be classified further as either a "**max heap**" or a "**min heap**". In a max heap, the keys of parent nodes are always greater than or equal to those of the children and the highest key is in the root node. In a min heap, the keys of parent nodes are less than or equal to those of the children and the lowest key is in the root node.

堆

- 堆可以实现在 O(logn) 时间内取最小值，插入一个元素，删除堆顶元素
- 基于这个性质，产生了复杂度为 O(nlogn) 的堆排序算法
- 堆也是一些算法之所以保持效率的重要数据结构，比如单源最短路的 Dijkstra's algorithm

堆

- 堆排序算法见 Heap.cpp
- 注意尽管直接看来建堆操作是 O(nlogn) 的，但可以证明，其实际上是 O(n) 的

优先队列

- 堆能干的另一件事就是优先队列
- 优先队列赋予每个元素一个优先级，每次出队的元素是优先级最高的
- 从定义就可以直接看到和堆的相似性
- 优先队列用到的操作比堆排序多一个 bubble(我自己这么叫的，官方似乎叫做 increaseKey)，代码见 Heap.cpp

UVa11136 Hoax or what

- 题目大意：维护一个序列，支持取最大、最小，删除最大、最小元素，插入操作
- 数据范围：查询操作 5000 次，插入操作 1e6 次

UVa11136 Hoax or what

- 看起来和堆很相似，于是考虑一个大根堆，一个小根堆
- 但是有个问题，如何使两个堆的信息沟通起来？很可能出现某次的最小值是之前某次的最大值，即是无效的
- 解决方案：为每个堆配套一个删除堆，排序逻辑保持一致
- 每次从堆里删除一个元素，首先跟自己的删除堆堆顶比较，如果相同，说明无效，删除堆 pop

UVa11136 Hoax or what

- 否则说明有效，将该元素插入另一个堆的删除堆
- 这里用到的是 lazy 思想，凡事能不干就不干，必须干的时候再干
- 因为要维护四个堆，所以推荐使用 STL 的 priority_queue 实现

UVa11136 Hoax or what

- priority_queue 的逻辑是弹出优先级最大的元素
- 所以大根堆要用 less，小根堆要用 greater

练习题

- 你们就随便上网找一找吧
- 反正这些东西没什么技巧，题目差异不太大
- 重要的是理解并会写

就酱