LUT-XCPC Day 1 语法 & 基础算法-1

syh.hs

兰州理工大学

July 20, 2022

Contents

知识准备

基础的递归、模拟
基本时间复杂度会看，有概念
会写暴力
视情况而定，哪里需要补充哪里的知识，临时讲一下即可

训练

C++ Reference
OI Wiki
知乎
每场 CF 的题解 & 评论区，大佬的过题代码

上述有不懂的进一步百度 | Google、交流，主要是坚持做题，少做水题、一眼题

各种比赛省赛级别获奖对比赛经验和知识准备要求不高，当然之后如果甘肃有 ICPC 的省赛，和兰大同台竞技另说、、

像 ZJ 等省份，XCPC 有省赛，拿金的难度可能还要大于区域银

题单

洛谷官方题单

2020,2021CF 简单题精选 适合有基础之后提高

搜索题单

关于做题和看题解.. 不会做看题解是正常的，尽量是一点点看，理解思路，然后尝试自己写出，不行再看代码，理解了之后再写，注意抄题解和看题解的区分，没有太多思考内容的题一般10-15min

Contents

summary

特定函数的部分主要看的是这篇 洛谷日报

模板，别的各种内容来自平时整理

algorithm 库

常见的函数有 swap,sort,unique,reverse,lower_bound,upper_bound 等...
一些别的函数
std::find
std::fill 一般用来弥补 memset 不能赋值的问题
std::max_element | min_element (bg, ed) 第三个参数可传入比较函数
std::count(bg,ed,val)
std::count_if(bg,ed,func) 常用 func 有 isdigit,islower,isupper 等
std::for_each(bg, ed, func)

numeric 库

std::accumulate(bg, ed, val)

可以用于序列求和，注意传参时 val 的类型避免溢出，第四个参数可以作为加法

std::partial_sum(bg1, ed1, bg2)

用于求前缀和，可以传入第四个参数作加法

std::adjacent_difference(bg1, ed1, bg2)

用于求差分，可以传入第四个参数作减法

cmath 库

exp(x)

返回 e^x，x 的有效范围是 $[-708.4, 709.8]$

log(x)

返回 $\ln x$，在 $x \leq 0$ 时候报错，别的还有 $log10$ 和 $log2$，其中 $log2$ 为 C++11 开始才有

floor 取上整
ceil 取下整

GNU

这些内容不在 C++ 标准中，如 clang 等其他编译器里可能没有，一般比赛提供的编译器都是 GUN C++，也就是你们 dev 里面自带的 mingw(Minimalist GNU for Windows)

GNU

__builtin 函数

__builtin_popcount(x) 统计二进制下 x 中 1 的个数

__builtin_parity(x) 统计二进制下 x 中 1 个数的奇偶性

__builtin_ffs(x) 统计二进制下最后一个 1 是从左往右第几位

__builtin_ctz(x) 返回二进制下后导 0 的个数

__builtin_clz(x) 返回二进制下前导 0 的个数

..

大多函数其实也没啥用，主要是压行，节省时间
Codeforces 打多了就能感受到

lambda 表达式

example

作用是提供一个匿名函数
sort(a.begin(), a.end(), [&](type a, type b) return ..);
这里与手写一个 cmp 填入作用相同

definition

[captures] (params) specifiers(可选) -> ret(可选) body
captures：捕获外部变量列表。
params：形参列表。
specifiers：指定符序列。C++11 中常用的为 mutable（允许 body 修改以值捕获的参数）。
ret：返回类型。若省略则由函数的 return 语句所隐含（或如函数不返回任何值则为 void）。
body：函数体。

lambda 表达式

竞赛里一般不考虑 specifier

example

```
int factorA = 1, factorB = 3;
std::vector<int> c(10);
std::iota(c.begin(), c.end(), 0);
for (auto it = c.begin(); it != c.end(); ++it)
  std::cout << *it << "\n"[it == std::prev(c.end())];
for_each(c.begin(), c.end(), [&](int &x) {
  x += factorA & factorB;
});
for (auto it : c)
  std::cout << it << ' ';
```

提一嘴 vector<bool>，这个东西不建议使用，具体原因知乎搜索

lambda 表达式

另一种方式是关于递归，如何调用自己的问题

example

```
auto sum = [](auto self, int x) -> int {
  return x == 1 ? 1 : x + self(self, x - 1);
};

std::cout << sum(sum, 10);
```

一种方法是用固定的 auto 格式
不过记得标明类型，否则可能自动推导失败
而且从 c++14 开始才支持 lambda 中参数类型有 auto，所以如果你要是 c++11 这么写递归就完了，比方说某绿桥杯、、

更多内容移步 OI-wiki lambda 表达式

lambda 表达式

因为上述问题，所以我们想要在 C++11 中使用 lambda 的递归，需要用 std::function 封装该匿名函数

example

```
std::function<int(int)> f = [&](int x) -> int {
  return x == 1 ? 1 : x + f(x - 1);
};
std::cout << f(3);
```

随机数 & shuffle

definition

mt19937 是一个随机数生成器类
shuffle 需要提供头尾迭代器和一个随机数生成器作为参数
随机数生成器需要一个种子来产生一个随机数，种子不同，随机结果才会不同
一般用 1970 年 (Unix 纪元) 至今秒数作为种子

```
std::vector<int> a(10);
std::iota(a.begin(), a.end(), 0);
std::mt19937 myrand(time(0));
std::shuffle(a.begin(), a.end(), myrand);
for (auto it : a)
  std::cout << it << ' ';
```

模板

关于 template

知道 template<typename T> | template<class T> 这两种比较基本的形式就可以，更多的知识和一些魔法技巧自行了解

Contents

零碎的语法糖

eg.

C++14 中 auto 可以作为普通函数的返回值
C++17 中支持结构化绑定
搭配 range-for 使用：

```
vector<pair<int, int>> v(n);
for (auto&[x,y] : v) cin>>x>>y;
```

用于带权图的遍历
C++17 中支持省略类型的 vector 初始化方式例如

```
int n, m;
std::cin >> n >> m;
std::vector a(n, std::vector<int>(m, 1));
```

如果有想了解 C++17 的可以看
C++17 in Detail

Contents

扯淡

语言特性的权重在竞赛学习中很低，只能起到锦上添花的作用，关键还是把题目做出来

还有就是关于贪心、DP 这一类思维题，最好直接 Codeforces 找 greedy dp 的标签，按照自身 rating+200 分的题开始做，别在概括性的套话和水题上纠结

扯淡

模拟题、字符串题怎么训练?

打开 Codeforces，点开 Problem Set，加上对应 tag，找合适分段的题目做

前缀和 & 差分

前缀和

pre[i] = pre[i-1] + a[i]
或者
pre[i] = pre[i-1] xor a[i]

二维前缀和

pre[i][j] = pre[i-1][j]+pre[i][j-1]-pre[i-1][j-1]+a[i][j]

差分

minus[i] = a[i] - a[i-1]

前缀和 & 差分

还有树上差分 + 倍增等各种结合做法

单调栈

开始单调线性结构，别的内容还有很多，有经验了之后自己去看 wiki 看文档吧..

(看情况给 5-10min 思考..)
其实比较局限，只能解决一类很固定的问题
因此也很容易看出来 233

模板题

对应代码

单调栈

看一道题

题意

给定长为 n 的数列 a，令 $n \le 10^5, a_i \le 10^9$
$\mathtt{sum} = (\mathtt{r} - \mathtt{l} + 1) * \mathtt{Min}, \mathtt{Min} = \mathtt{min}\{\mathtt{a_i} \mid \mathtt{i} \in [\mathtt{l}, \mathtt{r}]\}$，要求 sum 的最大值

单调栈

看一道题

题意

给定长为 n 的数列 a，令 $n \le 10^5, a_i \le 10^9$
$\mathtt{sum} = (\mathtt{r} - \mathtt{l} + 1) * \mathtt{Min}, \mathtt{Min} = \mathtt{min}\{\mathtt{a_i} \mid \mathtt{i} \in [\mathtt{l}, \mathtt{r}]\}$，要求 sum 的最大值

问题的实质是啥

单调栈

对于每个 i，我们希望找到对应的 [l,r]，保证 a[l],a[l+1],$\cdots$,a[r] 都大于等于 a[i]

单调栈

对于每个 i，我们希望找到对应的 [l,r]，保证 a[l],a[l+1],$\cdots$,a[r] 都大于等于 a[i]

不难意识到两个单调栈就能解决该问题

单调栈

对于每个 i，我们希望找到对应的 [l,r]，保证 a[l],a[l+1],$\cdots$,a[r] 都大于等于 a[i]

不难意识到两个单调栈就能解决该问题

自己动手试试！

单调队列

模板题

题意

给定长为 n 的序列 a，以及一个 k，窗口从 $[1,k]$ 一直滑动到 $[n-k+1,n]$
我们想知道每次窗口内的最大 / 最小值

单调队列

以求解最大值为例
可以发现，当前区间内靠后的，偏大的数字，比靠前的且偏小的数字要好，我们可以不保存后者

单调队列

以求解最大值为例
可以发现，当前区间内靠后的，偏大的数字，比靠前的且偏小的数字要好，我们可以不保存后者

如果要保存靠前的数字，那么一定是他比后面的数字大

单调队列

以求解最大值为例
可以发现，当前区间内靠后的，偏大的数字，比靠前的且偏小的数字要好，我们可以不保存后者

如果要保存靠前的数字，那么一定是他比后面的数字大
那么像是搞一个单调递减，下标递增的序列，类似

```
deque<pair<int,int>>
```

，来维护区间最大值

单调队列

以求解最大值为例
可以发现，当前区间内靠后的，偏大的数字，比靠前的且偏小的数字要好，我们可以不保存后者

如果要保存靠前的数字，那么一定是他比后面的数字大
那么像是搞一个单调递减，下标递增的序列，类似

```
deque<pair<int,int>>
```

，来维护区间最大值
对于新来的一个数 x，我们要不停的扔掉队列末端比 x 小的数，因为那些已经没有了保存的价值，这样就维护了一个单调性

同时检索头部的元素是否还在区间内，不在就 pop 掉

“如果一个选手比你小还比你强，你就可以退役了。”

——单调队列原理 [doge]

代码

单调队列可以维护左右端点都向一个方向移动时的区间最值，这个性质常用来做动态规划的优化

倍增 & ST 表

ST 表模板题

题意

给定 n, m 和不会改变的数组 $a\{n\}$，以及 m 个询问
每次询问给定一个 l,r，要求出 a[l] - a[r] 中的最大值，
$n \leq 10^5, m \leq 2 * 10^6, a_i \in [0, 10^9]$

倍增 & ST 表

如果我们知道了两个区间 $[l, r]$ 和 $[L, R]$ 的最大值，这里不妨设 $l \leq L \leq r \leq R$ ，那么就可以得到一个更大的区间,$[l, R]$ 中的最值

倍增 & ST 表

如果我们知道了两个区间 $[l, r]$ 和 $[L, R]$ 的最大值，这里不妨设 $l \leq L \leq r \leq R$ ，那么就可以得到一个更大的区间,$[l, R]$ 中的最值

我们在有了预处理基础的条件上，这里就是 a[i]=0 ，然后传递关系的时候成倍增长，即

$$\mathsf{max}\{\mathsf{a}[\mathsf{l}, \mathsf{l} + 2^{\mathsf{i}} - 1]\} = \mathsf{max}(\mathsf{max}\{\mathsf{a}[\mathsf{l}, \mathsf{l} + 2^{\mathsf{i}-1} - 1]\}, \mathsf{max}\{\mathsf{a}[\mathsf{l} + 2^{\mathsf{i}-1}, \mathsf{l} + 2^{\mathsf{i}} - 1]\})$$

我们设 f[i][j] 表示 $[\mathsf{l}, \mathsf{l} + 2^{\mathsf{j}} - 1]$ 中的最大值，显然有 $\mathsf{f}[\mathsf{i}][\mathsf{j}] = \mathsf{max}\{\mathsf{f}[\mathsf{i}][\mathsf{j} - 1], \mathsf{f}[\mathsf{i} + 2^{\mathsf{j}-1}][\mathsf{j} - 1]\}$

倍增 & ST 表

如何查询

对于给定的 [L,R]，我们要怎么查询最值?

倍增 & ST 表

如何查询

对于给定的 [L,R]，我们要怎么查询最值?

预处理 $\log_2$ 数组，或者调函数，令 $d = \log_2(R - L + 1)$
询问 $\max(f[L][d], f[L + 2^d][d])$ 即可

这种预处理信息时成倍增长的手段，就叫倍增

代码

注意 C++ 中位运算的优先级比加减要低，不确定时加上括号

另一种解法-ST 表

回过去看最大子矩形面积这个问题，找到最小值作为决定的边界，然后递归两边
每个数都会作为区间最小值被计算一次，总复杂度就是 ST 表的复杂度，预处理 $O(N\log N)$，查询 $O(1)$

随便糊的代码，就测了下小数据，因为 HDOJ 好像只有 C++98 能用，感兴趣的同学可以自己去写写试试，我懒得改了
这也是个很好的例子，如何把各种 STL 运用起来

另一种解法

堆

根据时间和观众意愿决定讲不讲
用 OI-wiki 和写过的博客凑合一下，这里不写内容了..

并查集

同上

Contents

写题

Codeforces Div.2 / Div. 3 泛做，无难度要求，自己看着搞

洛谷或其他平台上相关题目的练习

晚上抽出时间打 CF，如果早睡早起的可以 virtual participation

格式要求

写解题报告，每道题要有题目链接 | 来源、解题思路 (随便写两句说的对就行)、代码

可以是 word，LaTeX/Markdown 生成 pdf，博客，acwing 打卡等任意形式

没有数量要求，一周交，自愿为主