# CSP-S 2019 初赛试题解析

## 一、单项选择题

### 1. D

不会的去背数据类型和符号优先级.

### 2. C

其他三个是视频格式,不知道的去背.

### 3. D

不能不会.

### 4. B

编译器的作用:高级语言 $\rightarrow$ 机器语言(低级语言).

### 5. B

数据类型的强制转换,$int(x+0.5)$是四舍五入的基操.

### 6. B

排列组合的简单题,不会的问.

### 7. C

排序的稳定性:相同的元素的相对位置不变.

基于排序内在原理,不会的问.

### 8. B

求最少的顶点数,所以考虑完全图.

即求$n(n+1)/2 \geq 28$的最小正整数解.

### 9. B

题目中明确了只有$0,1,6,8,9$是可以颠倒的数字,
但是要注意的一点,第$3$位的数字必须是轴对称.

第$1,2$位有$(0,1,8,6,9)$五个数字,第$3$位有$(0,1,8)$三个数字,
对称性决定了第$4,5$位由第$1,2$位确定.

结合能被$3$整除的性质(各位数之和能被$3$整除),
由于$0,1,8$对$3$取模后分别为:$0,1,2$.

所以第$1,2$位上不管是什么数字,第$3$位都可以使得最后的数满足条件.

故答案为:$5*5$.

## 10. A

你好,小学生都会的.

## 11. D

归并算法是线性算法,最坏也只能是所有数都线性比较一遍.

故答案为 : $2n-1$.(最后一次比较可以确定两个数)

## 12. D

你好,不会的可以退赛了.

## 13. B

$Floyd$枚举了全部情况自然不是贪心.

其他算法均有取最小值(贪心).

## 14. B

设该数列为$\{a_{2n-1}\}$,公比为$q$.

有$a_1=2, a_{2n-1}=2q^{2n-1}=118098, a_n=2q^{n-1}=486.$

$\therefore q^{n-1}=248.$

$248$肉眼可见不是$2,4,5$的整数幂,故选$B$.

## 15. A

你好,最简单的$DP$模板.

# 二、阅读程序

## 1.

假设$a[5]=\{0,4,2,3,1\}$.

$i=1$, 显然,最后$ans=4$;

$i=2$, 行12的$if$成立,$ans=i$,最后$ans=2$;

$i=3$, $ans$仍为$i=2$时的值,最后$ans=4$.

不难发现行14的$while$寻找的是 :
从当前位置开始(连续),满足$a_j \le a_i (j \ge i)$的最大$j$.

$a_i$对应的答案为$ans_i$.

显然,若有$a_{i+1} \ge a_i$,必有$ans_{i+1} \ge ans_i$,所以可以从$ans_i$开始寻找答案.

故有行12的$if$.(一些作用不大的优化)

### 判断题

**(1) ×**

存在$ans=i$的情况.

**(2) √**

由行$14$的$while$条件,显然,$ans\leq n$.

**(3) √**

更改$if$条件后,即$a_i=a_{i-1}$时,可以从$ans_{i-1}$开始找,不然就重新开始,这显然不影响结果.

**(4) √**

显然.

### 单选题

**(5) D**

单调递增,所以$ans_i=i$,复杂度即为行$11$的$for$的复杂度,故为$O(N)$.

**(6) A**

$\{a_n\}$单调递减时,每次都要重新找,不难得到复杂度为$O(N^2)$.

## 2.

函数$getRoot$肉眼可见是用来找爹的.

结合行$23,23,26,27$可知,这是一个并查集.

行$25$,意义不明的运算.(可能就是做题用的)

### 判断题

**(1) √**

由行$15$的$for$的循环范围,显然.

**(2) ×**

这是并查集的另一种初始化的方法,并且不适用于下标从0开始的情况.

具体如下.

```
int pre[N];
void add(int a,int b)//据题 可能需要返回是否成功合并
{
    int fa=fd(a),fb=fd(b);
    if(fa!=fb) pre[fb]=fa;
}
int fd(int x){return pre[x]?pre[x]=fd(pre[x]):x;}
```

#### (3) √

显然.

#### (4) ×

$cnt_i$为$i$的并查集中的元素个数.

但注意,给出的代码并没有进行是否已经是同一个的并查集的判断,故可能存在$cnt_i > n$的情况.

### 选择题

#### (5) C

由题意,考虑朴素的合并过程,有$ans = 1 * (1 + 2 + ...... + n - 1)$,故$ans = 1225$.

若先合并成小集合再合并成大集合,由乘法分配律可知,最终结果相同,故$ans = 1225$.

#### (6) C

找爹的时候没有压缩路径,故为$O(N^2)$.
(这题王神仙也错了)

## 3.

前两个$for$的意义在题目的注释中已经给出了,是对$s$的前缀和后缀预处理.

并且要注意下标的$+1, -1$.

由行$22$的$while$的变量,可知$tmp$是$s_{0,i}$在$t$中的前缀子序列的最大值,
$suf_j + 1$是$s_{j,slen-1}$在$t$中的后缀子序列的最小值.

由退出$while$的结果:$tmp < suf_j + 1$,可知此时为$s$在$t$中的最大子序列(只考虑首尾).

结合$ans$的表达式,即为$s$在$t$中的子序列的相邻字符对应在$s$中的距离最大值.

### 判断题

#### (1) √

显然,更长的子串就可能有更长的子序列,故$suf_i \le suf_{i+1}$成立.

#### (2) ×

由上分析,当$t = s$,有$ans = 0$.

#### (3) ×

$i, j$反映的是$t$中的字符在$s$中的下标,行$22$的$while$的作用是找到满足条件的最小$j$.

故存在$j \le i$的情况.

#### (4) ×

显然是$pre_i \le suf_{i+1} + 1$.

### 选择题

**(5) D**

若$slen \leq tlen$,易得,对于任意的$i$都满足$pre_i <= suf_i$.

此时必有$i = j$.

$string$类型不能读入空串,故字符串长度至少为$1$.

**(6) C**

结合$ans$的实际意义,易知,$slen \geq tlen + ans$.

# 三、完善程序

## 1.

读完题后发现是一个简单的拓扑排序.

程序设计思路,首先肯定是拓扑模板,那么如何处理经验值的要求?

因为学习技术不消耗经验值,所以可以考虑贪心,每次都学习当前所需经验最少的技术.

可以用堆来维护,复杂度为$O(\sum_{i=1}^{n} m_i + N\log(N))$.
也可以用朴素的遍历,复杂度$O(\sum_{i=1}^{n} m_i + N^2)$.

从主函数开始阅读,结合选项,可知行$37$是拓扑的基本操作,即标记前驱个数,故$unlock[i] = m$.

$find$函数是寻找当前可以学习的技术.

行$15$的$if$条件就是判断能不能学习第$i$个技术.

观察选项有$unlock_i = -1$,不难推测,当学习技术$i$后,令$unlock_i = -1$,用于标记已学习.(见行$21$)

并且还有经验值的限制,故条件为:$unlock[i] == 0 \&\& threshold[i] <= points$.

学习后要增加相应的经验值,故行$22$为:$points += bonus[i]$.

并且还要解锁相应的技术,故行$24$为:$unlock[child[targe][i]] -= 1$.

**(1) C (2) D (3) D (4) C (5) B**

## 2.

~~你好,这个我也不会的.~~
~~以下是蒙题做法:~~
~~行22是在初始化二进制状态,不是全0就是全1,AB中选一个.~~
~~行25是在判断当前的石子数是否满足第j个条件,排除CD,因为A没有等号,所以猜测答案是B.~~
~~行26应当是状态转移方程,排除BC,又因为是二进制状态,故猜测答案是A.~~
~~行29就AD选一个,因为~~ ~~看着比较高级,符合二进制状压的逼格.~~
~~行30必然是改变胜负状态,排除AC,且trans显然是一个过程量,故猜测答案是D.~~
~~恭喜你在看不懂的情况下拿到了70%的分数.~~

**前置芝士: 博弈论Nim|SG函数**

$Nim$(在合法移动下有且仅有)两种局面:
$P$(先手必败),$N$(先手必胜).
有以下三条规则:
1.无法进行任何合法移动的局面是$P$
2.可以合法移动到$P$的局面是$N$.

$3.$只能合法移动到$N$的是$P$.
$(2, 3$其实是等价的$)$

$SG$函数定义:
$0($先手必败$), 1($先手必胜$)$.
$f(x) = 0|1$.
$(x$是一个二进制数,用来表示当前的局面或满足的规则$)$
显然, $f(0) = 0$.
$f(i) =!(\sum_{j=1}^{n} f(a_j) == n)$.
$(a_j$是$i$能合法移动到的所有局面$)$

从上面两个定义我们不难发现,在大多数情况下,我们是从游戏结束反推倒某一个具体的状态.

(因为游戏结束时的状态是唯一确定的)

现在阅读主函数,开始是在给规则按照$a_i$从小到大排序,这样保证了移动的单调性.

行22的$status$的初始化,在提示说$status$是胜负状态压缩的情况下,我会考虑是全1或者全0.

但我个人认为这个说法并不恰当,不过这不影响我们做下面的部分.
所以先跳过这个空.

行25是在判断当前$i$个石子的局面是否满足第$j$(读入的下标是从0开始的)条规则.

所以缺少的判断条件应该是: $a[j] <= i$.

但由于我们当前的状态必然是由之前的状态转移过来的,
$a[j] < i$的情况其实已经在之前转移过了,所以只要考虑等于的情况.

故行25为: $a[j] == i$.

提示很重要,二进制状压,第$i$位必然与恰好取$i$个石子的规则有关.
而$trans$是状态转移的过程量,所以我们每次移动后都应该更新$trans$.

$trans$的第$i$位用来表示我们已经有至少一条可以取$i$个石子的规则.
取走(放入)了$b[j]$块石头,所以应有 : $trans| = 1ull << (b[j] - 1)$.
(这里也证明了行25的正确性,因为$trans$只会增加可用的规则)

其实从行29和32都能看出来,其实最终决定当前局面胜负的是$win$的值,而不是$status$.
那么此时, $status$就只剩下一个身份了,用于表示当前局面.
但其实这么说也不准确.
因为从$SG$函数来看,能决定当前局面胜负的是当前局面能到达的所有局面的胜负.
从这里我们可以猜出, $status$其实就是表示当前局面能达到的所有局面的胜负
而$ull$和$b[i]$的范围可以隐约证明这一点,因为当前局面能达到的局面最多只有64种.
分别为取$1 - 64$个石子,与$trans$的每一位一一对应.

而行29,通过上述的分析,不难得出,应当是$x\&trans$.
(如果是$x|trans$的话,对于$a[i] <= m <= b[i]$,我们认为是先手必胜的,但显然这是先手必败的)

那么什么局面是先手必胜的呢?
取掉$i$个石子后的局面是先手必败的并且当前有可以取$i$个石子的规则.

从行32的输出可以得出$win$的值是全0时先手必败.
可以推出,在$status$中, 0表示先手必败, 1表示先手必胜.

那么行29就可以确定是$\sim status\&trans$.

而由于最多取64个石子,所以当我们要进入下一个$i$时,
最高位对于之后而言意味着取65个石子,而我们同时需要最低位来保存取1个石子时的胜负.

故行30为：$status = status << 1 \wedge win$.
(注意$win$是$bool$型，所以行30也可以写成$status = status << 1|win$)

到这里就可以知道$status$应当是1个石子所能到达的局面，而显然0个石子是先手必败.
故$status$的第1位应当是0.
而显然第$2-64$位对于1个石子全是不可以到达的，所以在取反后应该是0.
所以行22为：$status = \sim 0ull \wedge 1$.

**(1) C (2) B (3) A (4) D (5) D**