2020CSP-S模拟赛2 解析

1. 单项选择题（共15题，每题2分，共30分，每题只有一个正确选项）
2. 下列CCF主办的赛事活动，名称翻译不正确的是？（）B

A．CNCC（中国计算机大会）

B．NOIP(全国青少年信息学奥林匹克竞赛)

C．CTSC（国际信息学奥林匹克中国队选拔赛）

D．CSP-S（计算机非专业组别能力认证-提高级）

1. 二进制1011与十进制数2相乘的值是（ ）。A

A.(10110)2 B. (11010)2 C. (11100)2 D. (11111)2

3、与二进制数101.01011等值的十六进制数为(    ) D

A. A.B          B. 5.51      C. A.51            D. 5.58

小数部分：只有D选项满足1/4+1/16+1/32=5/16+8/256

4、工厂里有三根铁棒，分别长为 3，4，5，现在你可以对其中一些铁棒进行加长，但总的加长长度不能超过10，问有多少种加长的方案使得加长后的铁棒可以构成三角形。（ ）A

1. 187
2. 274
3. 286

D． 572

枚举加长长度i，那么总方案为sigma C(i,2) 插板法。

然后再减去非法的方案：非法方案计算也是插板法。

具体见XJOI 8835

5、在双向循环链表中，在p指针所指的结点后插入q所指向的新结点，其修改指针的操作是（ ）。 C

A．p->next=q; q->prior=p; p->next->prior=q; q->next=q;

B．p->next=q; p->next->prior=q; q->prior=p; q->next=p->next;

C．q->prior=p; q->next=p->next; p->next->prior=q; p->next=q;

D．q->prior=p; q->next=p->next; p->next=q; p->next->prior=q;

6、设n , m 为一棵二叉树上的两个结点，在中序遍历序列中n在m前的条件是（）。 B

A. n在m右方 B. n在m 左方

C. n是m的祖先 D. n是m的子孙

7、设哈希表长为7，哈希函数是H(key)=key%11,表中已有数据的关键字为15，38，61，84共四个，现要将关键字为49的结点加到表中，用线性探测法解决冲突，则放入的位置是( ) A

A．1 B．3 C．5 D．8

8、在含n个顶点和e条边的无向图的邻接矩阵中，零元素的个数为（　　） D

A. e B. 2e C. n^2-e D. n^2-2e

9、若（ax2+b/x）6的展开式中x3项的系数为20，则a2+b2的最小值为（）。B

A.1 B.2 C.3 D.4

由二项展开可得ab=1，a2+b2>=2ab=2

10、如果只想得到1024个元素组成的序列中的前5个最小元素，那么用（）方法最快。D

A. 冒泡排序 B. 快速排序

C. 直接选择排序 D. 堆排序

11、设有向图有n个顶点和e条边，采用邻接表作为其存储表示，在进行拓扑排序时，总的计算时间为（ ）。 B

A. O(nlog2e) B. O(n+e) C. O(ne) D. O(n2)

12、存储器ROM的特点不包括( )。A

A、ROM中的信息可读可写

B、ROM的访问速度高于光盘

C、ROM中的信息可长期保存

D、ROM是一种半导体存储器

13、T(n)=8T(n/2)+O(n^2)，

它描述了矩阵乘法问题第一个分治算法的运行时间，T(n)为（）。D

A. O（nlogn） 　　B. O（n^2）　　 C. O（n^2logn） D. O（n^3）

主定理可解

14、Internet实现了分布在世界各地的各类网络的互联，其最基础和核心的协议是（）。C

A、HTTP B、HTML C、TCP/IP D、FTP

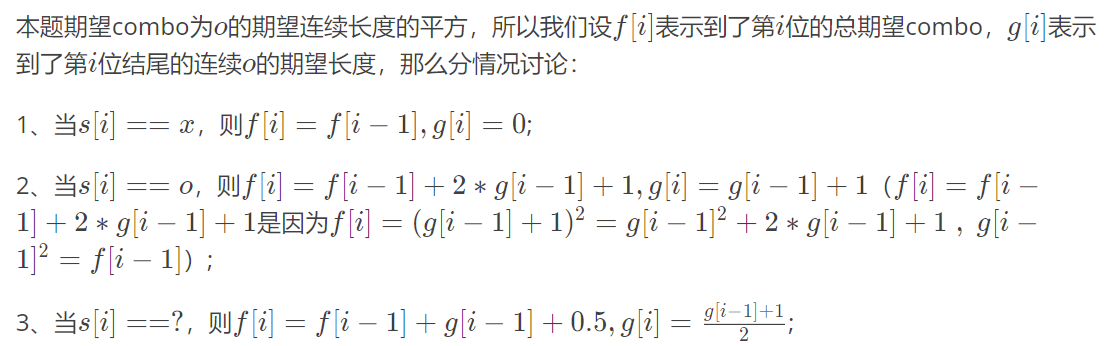
15、做n次点击游戏，成功了就是‘o’，失败了就是‘x’，计分规则：连续a个极大的连续‘o’就有a×a分。比如ooxxxxooooxxx，分数就是2×2+4×4=4+16=20。

考虑到，在游戏过程中，有些地方是拼运气，50%是‘o’，50%是‘x’，这样的位置用‘？’表示。现在要你求“？？？？“的期望（ ）。 D

提示：oo?xx。?是o就是oooxx => 9，是x就是ooxxx => 4，期望为(4+9)/2 =6.5。

A.1.625 B.2.75 C.3.375 D.4.125

原题见洛谷1365



四、阅读程序写结果（共4题，每题8分，共计32分）

阅读程序

1. 模拟

1 #include <bits/stdc++.h>

2 using namespace std;

3 int num;

4 int ans;

5 string binary;

6 string temp;

7 int main() {

8 cin >> num;

9 while(num) {

10 binary = (char)(num % 2 + '0') + binary;

11 num /= 2;

12 }

13 for(int i = binary.length(); i < 8; i++) {

14 binary = '0' + binary;

15 }

16 for(int i = 4; i < 8; i++) {

17 temp += binary[i];

18 }

19 binary = temp + binary;

20 for(int i = 0; i < 8; i++) {

21 ans += (binary[i] - '0') \* pow(2, 8 - i -1);

22 }

23 cout << ans;

24 return 0;

25 }

题意：8位二进制数下，后四位前四位交换位置

16、10行‘0’可替换成数字48。（ ） 是

17、14行交换‘+’符号两边‘0’和binary[i]位置，输出结果不变。（ ） 否

18、最后输出的ans一定比输入时的num大。（ ）否

19、ans的最大值为255。（ ）是

20、输入23，答案为（ ）。C

A．103 B．107 C．113 D．201

21、输入157，答案为（ ）。 B

A．195 B．217 C．223 D．311

2、

1 #include <bits/stdc++.h>

2 #define Min(a,b) a>b?b:a

3 #define Max(a,b) a>b?a:b

4 using namespace std;

5 int Len[3000005];

6 char str[3000005],s[3000005];

7 int n,mx,id,len;

8 void init() {

9 int k=0;

10 str[k++] = '$';

11 for(int i=0; i<len; i++) {

12 str[k++]='#';

13 str[k++]=s[i];

14 }

15 str[k++]='#';

16 len=k;

17 }

18 int f() {

19 Len[0] = 0;

20 int sum = 0;

21 mx = 0;

22 for(int i=1; i<len; i++) {

23 if(i < mx) Len[i] = Min(mx - i, Len[2 \* id - i]);

24 else Len[i] = 1;

25 while(str[i - Len[i]]== str[i + Len[i]]) Len[i]++;

26 if(Len[i] + i > mx) {

27 mx = Len[i] + i;

28 id = i;

29 sum = Max(sum, Len[i]);

30 }

31 }

32 return (sum - 1);

33 }

34 int main() {

35 memset(str,0,sizeof(str));

36 scanf("%s",s);

37 len = strlen(s);

38 init();

39 int temp = f();

40 printf("%d",temp);

41 return 0;

42 }

Manacher算法

O(n)时间计算序列中最长的回文

https://www.jianshu.com/p/392172762e55

22、执行完38行init()后，串str的长度是s的两倍。（ ） 否

23、去掉第10行对任意输入一定有影响。（ ） 否

不加，奇数长度的回文会出现下标越界问题

24、若i+Len[i]<mx，一定有Len[2\*id-1]=mx-i。（ ）否

25、25行对于每个i，while循环执行次数为常数级。（ ）是

因为 while 循环访问 mx 右边的数字用来扩展，也就是那些还未求出的节点，然后不断扩展，而期间访问的节点下次就不会再进入 while 了，可以利用对称得到自己的解，所以每个节点访问都是常数次

26、输入acbbcabcbadcgcdatbbbaaabcddc，答案为（ ）。C

A．3 B．5 C．7 D．9

27、该算法时间复杂度为（ ）。 A

A．O(n) B．O(n^1.5) C．O(n^2) D．O(n^2logn)

3、

#include<bits/stdc++.h>

using namespace std;

int tot;

int ACK(int m,int n) {

cout<<m<<" "<<n<<endl;

tot++;

if(m==0&&n>=0)

return 1+n;

else if(n==0&&m>0)

return ACK(m-1,0);

else if(m>0&&n>0)

return ACK(m-1,ACK(m,n-1));

}

int main() {

int m,n;

cin>>m>>n;

cout<<ACK(m,n)<<endl;

cout<<tot;

return 0;

}

递归题，强化手模能力

28、m、n为局部变量，tot为全局变量。（ ） 是

29、ACK(1,1)结果为2。（ ） 是

30、程序运行过程中，第5行m一定大于等于n。（ ）否

31、对于任意输入，tot值一定为偶数。（ ）否

32、ACK(3,3)为（ ）。B

A．3 B．4 C．5 D．6

33、该算法时间复杂度为（ ）。 C

A．O(2^(n+m)) B．O(2^(n\*m)) C．O(C((n+m),n)) D．O((n\*m)!)

三、完善程序（共2题，共40分）

【1】异或指针双向链表

这种链表，实际上就是把一般的双向链表的next和prior两个指针通过异或运算合并为一个指针域来存储，每个结点可以减少一个指针的空间

typedef struct XorNode{ //异或指针双向链表结点的定义

ElemType data;

struct XorNode\* LR;

}XorNode, \*XorPt;

typedef struct{//无头结点的异或指针双向链表

XorPt Left, Right; //只保存指向首、末结点的指针

}XorList;

每个结点的指针域存的是它的前驱和后继的异或值，即 LR = prior ^ next，指针异或运算本质上就是整型的内存地址的按位异或。

//指针异或函数，注意指针变量不支持^运算，需要强转为整型（内存地址，长度与机器有关，C将之封装为size\_t型）

XorPt XorP(XorPt p, XorPt q){

size\_t x = (size\_t)p;

size\_t y = (size\_t)q;

return (XorPt)(x^y);

}

以下代码实现的是无头结点的循环链表，而链表的表长又不是显式维护的，所以要以回到起点作为循环退出条件之一，并设一个记录步数的变量k来避免 i 值超过表长而导致的"兜圈子"；

Status CreateList(XorList& L, int n){

//头插法建表，无头结点，入口条件：n>=1

n--;

XorPt p;

p = (XorPt)malloc(sizeof(XorNode));

L.Left = L.Right = p;

p->data = n;

p->LR = XorP(p, p);

while(n--){

p = (XorPt)malloc(sizeof(XorNode));

p->data = n;

p->LR = XorP(L.Right, L.Left);

L.Left->LR = 1 ;

L.Right->LR = 2 ;

L.Left = p;

}

}

Status Delete(XorList& L, int i, ElemType& e){

//删除第i个位置的元素，用e返回其值（i的合法值为1~表长）

XorPt p = L.Left, q = L.Right;

int k = 1;

while( 3 ){

XorPt t = p;

p = XorP(q, p->LR);

q = t;

k++;

}

if(k<i) return ERROR; //表长不够

if( 4 ) L.Right = q;

e = p->data;

XorPt m = XorP(q, p->LR); //m暂存p的后继

q->LR = XorP(XorP(q->LR, p), m);

m->LR = XorP(q, XorP(p, m->LR));

free(p);

if(i==1) 5 ; //删除第一个结点

return OK;

}

异或指针链表的前驱和后继的指针编码到一个指针域中了，那么考虑如何解码，只凭LR本身是不可以的，必须再已知prior或next才行，这得益于异或运算特有的“对称”性质。

异或运算满足结合律，0为幺元，每个元素和自身互为逆元；所以设a, b为两个内存地址，则有

a^(a^b) = (a^a)^b = b;

(a^b)^b = a^(b^b) = a;

有了这两条，则可推出取得前驱或后继的式子，即

prior = LR ^ next;

next = prior ^ LR;

根据这两个关系，把双向链表的next和prior的运算用LR替换即可。

34、1处填（ ）。B

A．XorP(p,XorP(L.Right, L.Right->LR)) B．XorP(p,XorP(L.Right, L.Left->LR))

C．XorP(p,XorP(L.Left, L.Left->LR)) D．XorP(p,XorP(L.Left, L.Right->LR))

35、2处填（ ）。C

1. XorP(p,XorP(L.Right, L.Right->LR)) B．XorP(p,XorP(L.Right, L.Left->LR))

C．XorP(XorP(L.Right->LR,L.Left),p) D．XorP(XorP(L.Left->LR,L.Left),p)

1. 3处填（ ）。A
2. p!=L.Right && k<i B．p!=L.Right

C．k<I D．p!=L.Right || k<i

1. 4处填（ ）。D

A . q==L.Left B．q==L.Right

C．p==L.Left D．p==L.Right

1. 5处填（ ）。C

A . m = XorP(q, XorP(p, m->LR)) B．L.Left = XorP(q, XorP(p, m->LR))

C．L.Left = m D．m = L.Left

【2】N件物品，有两家商店出售。第i件物品在第一家商店的价格是a[i]，在第二家商店的价格是b[i]。小猪喜欢雨露均沾，如果一件物品小猪在第一家商店买，他一定也会在第二家商店购买。小猪买了若干件物品（至少1个），购物结束时，小猪发现在第一家商店花的钱恰好是在第二家商店花的钱的k倍。现在请你计算，小猪在第一家商店花的钱的最大值。

输入数据

第一行，两个整数 n, k (1 ≤ n ≤ 100, 1 ≤ k ≤ 10)；

第二行，包含 n 个整数，a1, a2, ..., an (1 ≤ ai ≤100)，表示n个物品在第一家商店的价格；

第三行，包含 n 个整数, b1, b2, ..., bn (1≤bi≤100)，表示n个物品在第二家商店的价格。

输出数据

共一行，一个整数，表示花费的最大值，若无解则输出-1。

样例输入1

3 2

10 8 1

2 7 1

样例输出1

18

#include <bits/stdc++.h>

using namespace std;

int n,k;

int s1[101],s2[101],a[101],b[101],ta=0,tb=0,f[10001],g[10001];

int getVal(int t){

return s1[t]- 6 ;

}

int main(){

cin>>n>>k;

for (int i=1;i<=n;i++)

cin>>s1[i];

for (int i=1;i<=n;i++)

cin>>s2[i];

int temp;

for (int i=1;i<=n;i++){

temp=getVal(i);

if (temp>=0)

a[ta++]=i;

else if (temp<0)

b[tb++]=i;

}

memset(f,0xf3,sizeof(f));

memset(g,0xf3,sizeof(g));

f[0]=0;

g[0]=0;

for (int i=0;i<ta;i++)

for (int j=10000;j>=getVal(a[i]);j--)

f[j]=max(f[j], 7 );

for (int i=0;i<tb;i++)

for (int j=10000;j>=-getVal(b[i]);j--)

g[j]=max(g[j], 8 );

int ss;

int maxx=-1;

for (ss=10000;ss>=1;ss--)

if ( 9 )

{

maxx=max( 10 ,maxx);

}

cout<<maxx;

return 0;

}

原题见XJOI 7156

单用01背包会造成后效性，即最大值不一定最优的情况

那么，既然最终要达到商店1总花费是商店2总花费的k倍，不如以商店2总花费的k倍为0，将其数轴左右分成两部分计算

以s1[i]代表第i个物品的商店1单价，s2[i]代表物品i在商店2的单价。

将物品根据s1[i]-k\*s2[i]的正负进行分类，装入a,b背包。

以s1[i]-k\*s2[i]的绝对值作为重量，s1[i]为价值进行两次01背包。

f,g分别代表背包1和背包2的dp数组

当f[i]>=0且g[i]>=0时，当前i合法即把f[i]+g[i]相加，迭代最大值。

39、6处填（ ）。C

A．1 B．k\*g[t]

C. k\*s2[t] D．k\*f[t]

40、7处填（ ）。B

A .g[j-getVal(a[i])]+s1[a[i]] B．f[j-getVal(a[i])]+s1[a[i]]

C.g[j-getVal(a[i])]+s2[a[i]] D．f[j-getVal(a[i])]+s2[a[i]]

41、8处填（ ）。D

1. f[j+getVal(b[i])]+s2[b[i]] B．g[j+getVal(b[i])]+s2[b[i]]
2. f[j+getVal(b[i])]+s1[b[i]] D．g[j+getVal(b[i])]+s1[b[i]]

42、9处填（ ）。A

A . f[ss]>=0&&g[ss]>=0 B．f[ss]+g[ss]>=0

C.f[ss]-g[ss]>=0 D．f[ss]\*g[ss]>=0

43、10处填（ ）。A

A . f[ss]+g[ss] B．f[ss]

C．g[ss] D．max(f[ss],g[ss])