CSP-S 2020模拟9

一、单项选择题（共十五题，单选）

1. 下列关于图灵奖的说法中，不正确的有（ ）。 A

A. 图灵奖是由电气和电子工程师协会（IEEE）设立的。

B. 目前获得该奖项的华人学者只有姚期智教授一人。

C. 其名称取自计算机科学的先驱、英国科学家艾伦·麦席森·图灵。

D. 它是计算机界最负盛名、最崇高的一个奖项，有“计算机界的诺贝尔奖” 之称。

2. 在8位二进制补码中，10101010表示的数是十进制下的：（ B ）

A.176 B. -86 C. -85 D. -84

3. 以下哪一台计算机是被实际制造出来的（ C ）

A. 巴贝奇差分机II号 B. 巴贝奇分析机 C. EDVAC D. 图灵机

4. 10、随机抛硬币，在连续三次得到的结果是正反正时停止。那么期望抛的次数是() D

A、7 B、8

C、9 D、10

5. 若f[1]=1,f[2]=5/6,3f[n+1]=5f[n]-2f[n-1]，则随着i的增大，f[i]将接近于（ C ）

A.1/3 B. sqrt(2)-1 C.1/2 D.1

6. Pinkex在做英语完型填空，要从7个选项中选出5个按顺序填入空中，Pinkex想知道，如果他随机选一个排列填进去，期望能做对几个空？（ ）D

A.2 B.749/640 C.5/21 D. 5/7

7. 高度为n的均衡的二叉树是指，如果去掉叶结点及相应的树枝，它应该是高度为n-1的满二叉树。在这里，树高等于叶结点的最大深度，根结点的深度为0，如果某个均衡的二叉树共有2381个结点，则该树的树高为（ B ）

A. 10 B.11 C.12. D.13

8. 在 n（n ≥ 3）枚硬币中有一枚质量不合格的硬币（质量过轻或质量过重）， 如果只有一架天平可以用来称重且称重的硬币数没有限制，下面是找出这枚 不合格的硬币的算法。请把 a-c 三行代码补全到算法中。

a. 𝐴 ← 𝑋 ∪ 𝑌

b. 𝐴 ← 𝑍

c. 𝑛 ← |𝐴|

算法 Coin(A, n)

1. 𝑘 ← ⌊𝑛/3⌋

2. 将 A 中硬币分成 X，Y，Z 三个集合，使得|𝑋| = |𝑌| = 𝑘,|𝑍| = 𝑛 − 2𝑘

3. if 𝑊(𝑋) ≠ 𝑊(𝑌) //W(X), W(Y)分别为 X 或 Y 的重量

4. then \_\_\_\_\_\_\_\_\_\_

5. else \_\_\_\_\_\_\_\_\_\_

6. \_\_\_\_\_\_\_\_\_\_

7. if n>2 then goto 1

8. if n=2 then 任取 A 中 1 枚硬币与拿走硬币比较，若不等，则它不合格； 若相等，则 A 中剩下的硬币不合格.

9. if n=1 then A 中硬币不合格

正确的填空顺序是（ ）。 D

1. b, c, a B. c, b, a C. c, a, b D. a, b, c

9. 公共汽车起点站于每小时的10分,30分,55分发车,该顾客不知发车时间,在每小时内的任一时刻随机到达车站,求乘客候车时间的数学期望（准确到秒）.（ ）B

A.5分13秒 B.10分25秒 C.22分30秒 D.35分30秒

如上图所示,三个三角形面积除以60即为期望

即(15^2+20^2+25^2)/120=10分25秒

10. 将关键字序列（7、8、30、11、18、9、14）散列存储到散列表中。散列表的存储空间是一个下标从0开始的一维数组。散列函数为： H(key) = (keyx3) MOD 7，处理冲突采用线性探测再散列法，哈希表长10，请你计算等概率情况下**查找不成功**的平均查找长度（）。 D

A. 9/5 B. 9/7 C. 12/7 D. 18/7

解析：https://blog.csdn.net/longlovefilm/article/details/78009782

11. 某文本包含240个汉字、39个数字以及666个字母，若将其强制转换为一个char数组，则数组的长度为（ D ）

A.945 B.279 C.1851 D.1185

12. 中缀表达式A-(B+C/D)\*E的后缀表达式是（ D ）

A.AB-C+D/E\* B.ABC+D/-E\* C.ABCD/E\*+- D.ABCD/+E\*-

13. IPv4和IPv6是是目前使用的两种Internet协议版本，下列说法错误的是：（ C ）

A. IPV4的协议地址长度是32位，IPV6的协议地址长度是128位。

B. IPv4地址是以小数表示的二进制数。 IPv6地址是以十六进制表示的二进制数。

C. IPv4提供身份验证和加密，IPv6不提供身份验证和加密。

D. IPv4协议的地址可以通过手动或DHCP配置的，但IPV6不行。

14. 以下事项不违反CSP-J/S考场纪律的是（ D ）。

A． 为了看时间将电话手表带入考场。

B． 在开考2小时内离场

C． 在开始考试前在机器上默写模版代码

D． 向监考员举手示意需要去洗手间

15.下列说法正确的是（ C ） 算法

A.SPFA算法无法用来判断给定图是否存在负环

B. 当图中不存在负权环但是存在负权边，Dijkstra算法一定不能求最短路

C. 当图中不存在负权环但是存在负权边，bellman-ford算法一定能求最短路

D.相比于稀疏图，在稠密图上更适合使用SPFA算法

三、阅读程序写结果(共3题，每题10分)

1.

#include <bits/stdc++.h>

int main() {

char str[60];

int len,i,j;

int nchr[26];

char mmin;

mmin='z';

scanf("%s",str);

len=strlen(str);

for(i=len-1; i>=1; i--)

if(str[i-1]<str[i])

break;

if(i==0) {

printf("No result!\n");

return 0;

}

for(j=0; j<i-1; j++)

putchar(str[j]);

memset(nchr,0,sizeof(nchr));

for(j=i; j<len; j++) {

if(str[j]>str[i-1]&&str[j]<mmin)

mmin=str[j];

nchr[str[j]-'a']++;

}

nchr[mmin-'a']--;

nchr[str[i-1]-'a']++;

putchar(mmin);

for(i=0; i<26; i++)

for(j=0; j<nchr[i]; j++)

putchar(i+'a');

putchar('\n');

return 0;

}

16、第18行‘<’改’<=’不会影响输出结果。（ ） 否

17、若后n-i个数为严格递减序列，第i个数一定是最小的.( ) 否

18、原始序列第i位之后的序列，一定非增，结果序列第i位之后序列一定非减.( ) 是

19、若结果不为“No result!”最后输出序列不可能有序（从小到大排或者从大到小排）。（） 否

20、输入：zzyzcccbbbaaa 结果为( )： C

A. zzzycccbbbaaa

B. zzzyaaabbbccc

C. zzzaaabbbcccy

D. aaabbbcccyzzz

21、输入：zzyzxxjjaabb 结果为( )： B

A. zzzaabbjjxxy

B. zzyzxxjjabab

C. zzzyaabbjjxx

D. zzyzxxjjaabb

2.

#include<bits/stdc++.h>

using namespace std;

int map1[55][55][55];

int vis[55][55][55];

typedef struct {

int z,x,y;

} node;

int aa[6][3]= {1,0,0, -1,0,0, 0,0,1, 0,1,0, 0,0,-1, 0,-1,0};

int a,b,c,tt,yy;

void fun() {

node w,d;

w.z=1;

w.x=1;

w.y=1;

queue <node > q;

q.push(w);

while(!q.empty()) {

d=q.front();

for(int i=0; i<6; i++) {

w.z=d.z+aa[i][0];

w.x=d.x+aa[i][1];

w.y=d.y+aa[i][2];

if(w.z>=1&&w.z<=a&&w.x>=1&&w.x<=b&&w.y>=1&&w.y<=c &&map1[w.z][w.x][w.y]==0&&vis[w.z][w.x][w.y]==0) {

vis[w.z][w.x][w.y]=vis[d.z][d.x][d.y]+1;

if(vis[w.z][w.x][w.y]>tt)

return ;

if(w.z==a&&w.x==b&&w.y==c) {

yy=1;

printf("%d\n",vis[d.z][d.x][d.y]);

return ;

} else q.push(w);

}

}

q.pop();

}

return ;

}

int main () {

int i,j,k;

scanf("%d %d %d %d",&a,&b,&c,&tt);

memset(vis,0,sizeof(vis));

for(i=1; i<=a; i++)

for(j=1; j<=b; j++)

for(k=1; k<=c; k++)

scanf("%d",&map1[i][j][k]);

yy=0;

vis[1][1][1]=1;

fun();

if(yy==0) printf("-1\n");

return 0;

}

22、map1数组在内存中占665500字节。（ ） 是

23、整个程序运行结束可能没有输出。（ ） 否

24、q.front()可以改为q.top()。 （ ）否

25、若tt大于输入中所有数字之和，输出结果一定不为-1 （ ）否

26、输入：

3 3 4 20

0 1 1 1

0 0 1 1

0 1 1 1

1 1 1 1

1 0 0 1

0 1 1 1

0 0 0 0

0 1 1 0

0 1 1 0

输出（ A ）：

A.11 B.15 C.17 D,23

27、同上一题的输入，while循环执行次数为：（ ）。 D

A．11

B．12

C．13

D．14

3.

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

2 using namespace std;

3 const int maxNum = 1005;

4 int shift[maxNum];

5 int Sd(const string& T, const string& P) {

6 int n = T.length();

7 int m = P.length();

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

9 shift[i] = m + 1;

10 }

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

12 shift[P[i]] = m - i;

13 }

14 int s = 0;

15 int j;

16 while(s <= n - m) {

17 j = 0;

18 while(T[s + j] == P[j]) {

19 j++;

20 if(j >= m) {

21 return s;

22 }

23 }

24 s += shift[T[s + m]];

25 }

26 return -1;

27 }

28 int main() {

29 string T, P;

30 getline(cin, T);

31 getline(cin, P);

32 int res = Sd(T, P);

33 cout << res << endl;

34 return 0;

35 }

28、第5行，const string& T相比const string T，可以更好避免内存资源浪费。 T

29、实际执行shift[i]不会超过m+1。（ ） T

30、输入：

Bc

B

会输出-1。 否

31、24行“s += shift[T[s + m]];”改成“s += shift[T[s + j]];”不会影响输出结果。 F

32、输入:

Pdfpdfpdf

pdf

输出：（ ） B

A.0 B.3 C.4 D.6

33、输入:

adasfasfasfasgaegagfasdf

gf

输出：（ D ）

A.4 B.7 C.13 D.18

四、程序填空

①《Qjl的比赛》

Qjl是一名神犇，他最近出了一场盛大的AK赛。比赛共含有n(1≤n≤100000)题，由于是AK赛，任何人做了任何一道题都能保证AC。但是Qjl身为神犇，要摆神犇的架子，具体做法是对于不同的题，赋予不同的满分，这样会让AK者的分数变得好看。不过，他的比赛中不断会出现出锅的情况，也就是题目满分应该被调整。他会不断调整一些题目的满分，使之更合理。现在Qjl需要你编写程序，帮他满足两种操作，如果你写出了代码，你有可能会获得2147483648%32768分的额外满分！

他首先给你两个正整数n,m，表示题目数与操作数；然后他会给你n个正整数，表示这些题目初始的满分。接下来他会不断给你发m条操作指令：

1 x y k：表示将题号i∈[x,y]的题目的满分都加上k；2 x y：表示询问题号i∈[x,y]的所有题目的AK分（即总分和。）请你编写程序，以享受AK的快感。

#include<bits/stdc++.h>

#define int long long

using namespace std;

struct point {

int lft,rgt,num,lzy;

} a[400401];

int b[100001],n,m,i,x,y,z,t;

void build(int p,int l,int r) {

a[p].lft=l;

a[p].rgt=r;

if (l==r) {

a[p].num=b[l];

return;

}

build( ① ,l,(l+r)/2);

build( ② ,(l+r)/2+1,r);

a[p].num=a[p\*2].num+a[p\*2+1].num;

}

void down(int p) {

if ( ③ ) {

a[p\*2].num+=a[p].lzy\*(a[p\*2].rgt-a[p\*2].lft+1);

a[p\*2+1].num+=a[p].lzy\*(a[p\*2+1].rgt-a[p\*2+1].lft+1);

a[p\*2].lzy+=a[p].lzy;

a[p\*2+1].lzy+=a[p].lzy;

④ ;

}

}

void add2(int nd,int st,int ed,int x) {

if (st<=a[nd].lft && ed>=a[nd].rgt) {

a[nd].num+= ⑤ ;

a[nd].lzy+=x;

return;

}

down(nd);

if (st<=(a[nd].lft+a[nd].rgt)/2) add2(nd\*2,st,ed,x);

if (ed>=(a[nd].lft+a[nd].rgt)/2+1) add2(nd\*2+1,st,ed,x);

a[nd].num=a[nd\*2].num+a[nd\*2+1].num;

}

int Find(int nd,int l,int r) {

if ( ⑥ ) return ⑦ ;

down(nd);

int bb=0;

if (l<=(a[nd].lft+a[nd].rgt)/2) bb+=Find(nd\*2,l,r);

if (r>=(a[nd].lft+a[nd].rgt)/2+1) bb+=Find(nd\*2+1,l,r);

return bb;

}

signed main() {

scanf("%lld%lld",&n,&m);

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

scanf("%lld",&b[i]);

}

build(1,1,n);

for (i=1; i<=m; i++) {

scanf("%lld",&t);

if (t==1) {

scanf("%lld%lld%lld",&x,&y,&z);

add2(1,x,y,z);

} else {

scanf("%lld%lld",&x,&y);

printf("%lld\n",Find(1,x,y));

}

}

}

34. 填入①和②的代码分别是

A.p\*2；p\*2+1

B.p\*2+1；p\*2

C.p+1；p+2

D.p+2；p+1

35填入③处的代码是

A.a[p].num==0

B.a[p].num!=0

C.a[p].lay==0

D.a[p].lzy!=0

36.填入④处的代码是

A.a[p].num=0

B.a[p].lzy=0

C.a[p].num=a[p].lzy=0

D.{down(p\*2);down(p\*2+1);}

37.填入⑤处的代码是

A.a[nd].rgt-a[nd].lft+1

B.a[nd].rgt-a[nd].lft

C.x\*(a[nd].rgt-a[nd].lft+1)

D.x\*(a[nd].rgt-a[nd].lft)

38.填入⑥和⑦处的代码分别是

A.l<=a[nd].lft && a[nd].rgt<=r；a[nd].num

B.l<=a[nd].lft && a[nd].rgt<=r；a[nd].num\*(l-r+1)

C.a[nd].lft<=l && r<=a[nd].rgt；a[nd].num\*(l-r+1)

D.a[nd].lft<=l && r<=a[nd].rgt；a[nd].num

②《奶牛也感染》

每头奶牛都不想成为被新型病毒感染的奶牛。被所有奶牛接触的奶牛就是一头被感染的奶牛。因为奶牛喜欢偷偷摸摸地行动，所有接触是单向的，所有奶牛都与自己接触。奶牛之间的接触是可以传递的——如果 A接触过 B，B接触过C，那么A也接触过C。牛栏里共有N头奶牛，给定一些奶牛之间的接触关系，请你算出有多少头奶牛不幸被感染。

输入的第一行是两个用空格分开的整数N 和 M。接下来 M 行每行两个用空格分开的整数A 和 B，表示 A接触过B。输出被感染的奶牛的数量。

#include<bits/stdc++.h>

using namespace std;

struct edge {

int v,next;

} p[50001];

int front[10001],M;

void Add(int u,int v) {

++M;

p[M].v=v;

① ;

front[u]=M;

}

int dfn[10001],low[10001],N,s;

stack<int> zhan;

int which[10001];

int Out[10001],id,large[10001];

int n,m;

int dfs(int x) {

++N;

dfn[x]=low[x]=N;

zhan.push(x);

for (int i=front[x]; i; i=p[i].next) {

if ( ② ) {

dfs(p[i].v);

③

} else

If() low[x]=min(low[x],dfn[p[i].v]);

}

if (dfn[x]==low[x]) {

++s;

int S=0;

int v;

do {

v=zhan.top();

zhan.pop();

④

++S;

} while (x!=v);

Large[s]=S;

}

}

int main() {

int i,j;

scanf("%d%d",&n,&m);

for (i=1; i<=m; ++i) {

int x,y;

scanf("%d%d",&x,&y);

Add(x,y);

}

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

if (dfn[i]==0) dfs(i);

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

for (j=front[i]; j; j=p[j].next)

if (which[p[j].v]!=which[i]) ⑤ ;

for (i=1; i<=s; ++i)

if (Out[i]==0) {

if (id!=0) {

printf("%d\n",0);

return 0;

}

id=i;

}

printf("%d\n",large[id]);

return 0;

}

39.①处填入的代码是

A.p[M].next=u

B.p[M].next=v

C.p[M].next=front[u]

D.p[M].next=front[v]

40.②处填入的代码是

A.!dfn[p[i].v]

B.dfn[p[i].v]

C.low[p[i].v]

D.!low[p[i].v]

41.③处填入的代码是

A.dfn[x]=min(dfn[x],dfn[p[i].v]);

B.dfn[x]=max(dfn[x],low[p[i].v]);

C.low[x]=max(low[x],dfn[p[i].v]);

D.low[x]=min(low[x],low[p[i].v]);

42.④处填入的代码是

A.++s;

B.which[v]=s;

C.x=zhan.top(),zhan.pop();

D.zhan.push(x+v);

43.⑤处填入的代码是

A.++Out[which[i]]

B.++which[Out[i]

C.++Out[Out[i]]

D.++which[which[i]]

ADBCA C A(D) DBA