

CSP-S 初赛模拟题

一、单项选择题（共 15 题，每题 2 分，共 30 分）

1、在计算机硬件系统中，cache 是()存储器。

A、只读 B、可编程只读 C、可擦除可编程只读 D、高速缓冲

2、算式 $(2047)_{10} - (3FF)_{16} + (2000)_8$ 的结果是()。

A、 $(2048)_{10}$ B、 $(2049)_{10}$ C、 $(3746)_8$ D、 $(1AF7)_{16}$

3、一棵非空二叉树的先序遍历序列和后序遍历序列正好相反，则该二叉树一定满足()。

A、所有结点均无左孩子 B、所有的结点均无右孩子

C、只有一个叶子结点 D、是任意一棵二叉树

4、将有关二叉树的概念推广到三叉树，则一棵有 244 个结点的完全三叉树的高度是()。假设只有一个结点的树的高度是 1。

A、4 B、5 C、6 D、7

5、设栈 S 的初始状态为空，元素 a, b, c, d, e, f 依次入栈 S，出栈的序列为 b, c, d, f, e, a，则栈 S 的容量至少应该是()。

A、2 B、3 C、4 D、5

6、以比较为基本运算，对于 $2n$ 个数，同时找到最大值和最小值，最坏情况下需要的最小的比较次数为()。

A、 $4n-2$ B、 $3n+1$ C、 $3n-2$ D、 $2n+1$

7、二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。

那么，2023 个顶点的二分图至多有()条边。

A. 1023132 B. 1024144 C. 2022 D. 4090506

8、定义两个字符串之间的莱温斯坦距离为它们的编辑距离，即将一个字符串通过“添加一个字符”、“修改一个字符”、“删除一个字符”这三种操作变成另一个字符串所需的最少操作次数。问字符串“olympics”和“poems”之间的莱温斯坦距离为()。

A、5 B、6 C、7 D、8

9、设一个三位数 $n=abc$ ，其中 a, b, c 均为 1 到 9 之间的整数，若以 a, b, c 作为三角形的三条边可以构成等腰三角形（包括等边），则这样的 n 有()个。

A、81 B、120 C、165 D、216

10、有一个有 n 个顶点和 m 条边的图，如果采用邻接矩阵来进行存储，那么对该图进行深度优先遍历的时间复杂度为()。

A、 $O(n)$ B、 $O(n+m)$ C、 $O(n*m)$ D、 $O(n^2)$

11、观察以下代码，其时间复杂度是()。

```
for(int i=1;i<=n;i++){
    for(int j=1;j*j<=i;j++){
        for(int k=1;k<=j;k++){
            ans++;
        }
    }
}
```

A、 $O(n^{3/2})$ B、 $O(n^2)$ C、 $O(n^{5/2})$ D、 $O(n^{3/2}\log n)$

12、观察以下代码，当输入的 $x=2,y=23$ 时，输出结果为()。

```
int f(int x,int y){
    if(x==0||y==0) return x+y;
    else return f(x-1,y)+f(x,y/2)-2*f(x-1,y/2);
}
```

A、-1 B、-3 C、-4 D、-6

13、现有一个双向链表，想删除其中的一个结点 s ，用 $s->prev$ 指向 s 的前驱， $s->next$ 指向 s 的后继，则正确的删除代码为()。

A、 $s->prev->next=s->next,s->next->prev=s->prev;$

B、 $s->next=s->prev->next,s->prev=s->next->prev;$

C、 $s->prev=s->next,s->next=s->prev;$

D、 $s->prev->next=s->next->prev,s->next->prev=s->prev->next;$

14、现有 8 个苹果和 3 个有编号排成一排的盘子，将苹果放进盘子中，要求相邻盘子中苹果的数量不同，且允许有空盘子，则共有()种放法。

A、28 B、32 C、35 D、39

15、现有一个地址区间为 $[0,22]$ 的哈希表（即大小为 23 的哈希表），对于出现冲突情况，会往前找第一个空的地址存储。已知输入数据为 $(1949,1964,1976,2000,2008,2012,2023)$ ，哈希函数为 $x*(x+1)\%23$ ，问 2023 的地址为()。

A、0 B、22 C、21 D、20

二、阅读程序题（共 3 题，判断题 2 分，选择题 3 分，共 40 分）

1、

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

02 using namespace std;

03 int main(){

04 int n,a,b,l,r,ans;

05 int k[205],q[205],v[205]={0};

06 cin >> n >> a >> b;

07 for(int i=1;i<=n;i++) cin >> k[i];

08 if(a==b){

09 cout << "0" << endl;

10 return 0;

11 }

12 q[0]=a;v[a]=1;

13 l=0;r=1;ans=0;

14 while(l<r){

15 int up=q[l]+k[q[l]],down=q[l]-k[q[l]];

16 if(up<=n)

17 if(!v[up]){

18 q[r]=up;

19 v[up]=v[q[l]]+1;

20 r++;

21 }

```

22         if(down>0)
23             if(!v[down]){
24                 q[r]=down;
25                 v[down]=v[q[l]]+1;
26                 r++;
27             }
28         l++;
29     }
30     if(v[b]) cout << v[b]-1 << endl;
31     else cout << "-1" << endl;
32     return 0;
33 }

```

判断题:

- (1) 该程序用到的算法是深度优先搜索。()
- (2) 删除该程序的第 8~11 行, 程序运行结果不变。()
- (3) 运行过程中 v 数组内可能会有大于 n 的值。()
- (4) 输入中必须有 $a < b$ 。()

单选题:

(5) 如果输入为:

5 1 5

3 3 1 2 5

输出结果为()。

A、1 B、2 C、3 D、4

(6) 如果输入的 k 数组内元素都是 1, 则输出为()。

A、a B、b C、b+a D、|b-a|

2、

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 const int maxn=1e5+3;
04 struct edge{
05     int v,next;
06 }e[2*maxn];
07 int head[maxn],a[maxn],deg[maxn],dp[maxn];
08 bool vis[maxn];
09 int cnt,num,n,m,x,y;
10 queue<int>q;
11 void add(int x,int y){
12     e[++cnt].next=head[x];head[x]=cnt;e[cnt].v=y;deg[y]++;
13 }
14 void toposort(){
15     for(int i=1;i<=n;i++) if(deg[i]==0) q.push(i),vis[i]=1;
16     while(!q.empty()){
17         int x=q.front();q.pop();

```

```

18     a[++num]=x;
19     for(int i=head[x];i;i=e[i].next){
20         int y=e[i].v;
21         if(!vis[y]) deg[y]--;
22         if(deg[y]==0) q.push(y),vis[y]=1;
23     }
24 }
25 }
26 int main(){
27     cin >> n >> m;
28     for(int i=1;i<=m;i++){
29         cin >> x >> y;
30         add(x,y);
31     }
32     toposort();
33     for(int i=1;i<=n;i++) dp[i]=1;
34     for(int i=1;i<=n;i++)
35         for(int j=head[a[i]];j;j=e[j].next){
36             int k=e[j].v;
37             dp[k]=max(dp[k],dp[a[i]]+1);
38         }
39     for(int i=1;i<=n;i++) cout << dp[i] << " ";
40 }

```

判断题：

- (1) 该程序用到的数据结构有队列和图。 ()
- (2) 将第 30 行的 add(x,y)改成 add(y,x)，输出结果不变。 ()
- (3) 将第 15 行中的循环改成 for(int i=n;i>0;i--)，输出结果不变。 ()
- (4) 本程序求的是有向无环图中终点为每个顶点的最长路径。 ()

单选题：

(5) 当输入为

5 6

1 2

1 3

2 3

2 4

3 4

2 5

时，输出为()。

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

(6) 假设输入有 n 行，第一行为正整数 n，除第一行外，第 k 行为两个正整数 $k \lfloor k/2 \rfloor$ (这里 $\lfloor x \rfloor$ 表示对 x 向下取整)，则 $dp[1]=()$ 。

A、 $\lfloor \log_2 n \rfloor$ B、 $\lfloor \log_2 n \rfloor + 1$ C、 $\lfloor n/2 \rfloor$ D、 $\lfloor n/2 \rfloor + 1$

3、

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 int ans;
04 string s1,s2;
05 string b_sm(string s){
06     int l=s.size();
07     for(int i=0;i<l;i++){
08         if(s[i]>='A'&& s[i]<='Z')s[i]+=32;
09     }
10     return s;
11 }
12 int main(){
13     cin>>s1>>s2;
14     s1=b_sm(s1);s2=b_sm(s2);
15     int n=s2.size(),m=s1.size();
16     for(int i=0;i<n;i++){
17         bool flag=true;
18         for(int j=0;j<m;j++){
19             if(s1[j]!=s2[i+j]) flag=false;
20             if(flag) ans++;
21         }
22     }
23 }

```

判断题：

- (1) 函数 b_sm 的作用是将小写字母转变为大写字母。 ()
- (2) 该程序的总时间复杂度是 $O(nm)$ 。 ()
- (3) 当 $n < m$ 时，输出为 0。 ()

单选题：

(4) 当输入为

Aaba

AbaabaabAAabBAbaBAAabAaBabaAabB

时，输出为()。

A、 4 B、 5 C、 6 D、 7

(5) 如果输入的两个字符串中都只有字符 a 和 A，则输出为()。

A、 n B、 m C、 n-m D、 n-m+1

三、完善程序题（单选题，每小题 3 分，共 30 分）

1、求星星放法数

题目描述：现在有一个 $n \times n$ 的盘面，你需要在盘面内每行每列放置 m 个星星，使得任何两个星星不相邻也不对角相邻（即一个星星的周围 8 格范围内没有其他星星）。盘面内有一些格子是障碍物，障碍物上不能放置星星。求总的放置星星的方法数。

```

01 #include<bits/stdc++.h>
02 using namespace std;
03 int dx[8]={-1,-1,-1,0,0,1,1,1},dy[8]={-1,0,1,-1,1,-1,0,1}; //八个方向

```

```

04 int a[25][25],n,m,ans,tot,cnt[25];
05 //a 数组为地图，0 表示空格，1 表示障碍，2 表示星星；cnt 表示每列的星星数量；tot
    记录总星星数量
06 void search(int row,int k,int pos){ //找第 row 行，第 k 个星星，该行前一个星星在 pos
07     for(int i=pos+1;i<=n;i++){
08         if(a[row][i]==0&&__①__){
09             bool flag=true;
10             for(int j=0;j<8;j++){
11                 int x=row+dx[j],y=i+dy[j];
12                 if(x>0&&x<=n&&y>0&&y<=n)
13                     if(__②__) flag=false; //如果周围 8 格有星星，则该格不能放
14             }
15             if(flag){
16                 a[row][i]=2; //标记这个位置是星星
17                 cnt[i]++; //本列星星数量加 1
18                 tot++; //星星总数加 1
19                 if(__③__) ans++; //如果星星的数量够了，就是一种合法方案
20                 else if(k==m) __④__; //如果这一行已有 m 个星星，找下一行
21                 else search(row,k+1,i); //否则继续找这一行的下一个星星
22                 tot--; //回溯
23                 __⑤__;
24                 a[row][i]=0;
25             }
26         }
27     }
28 }
29 int main(){
30     cin >> n >> m;
31     for(int i=1;i<=n;i++)
32         for(int j=1;j<=n;j++) cin >> a[i][j];
33     search(1,1,0);
34     cout << ans;
35 }

```

(1) ①处应填()。

A、cnt[i]<=m B、cnt[i]<m C、cnt[i]==m D、cnt[i]>m

(2) ②处应填()。

A、a[x][y]==2 B、a[x][y]==1 C、a[row][i]==2 D、a[row][i]==1

(3) ③处应填()。

A、row==n B、k==m C、tot==n*n D、tot==m*n

(4) ④处应填()。

A、search(row+1,k+1,0) B、search(row+1,1,i) C、search(row+1,1,0) D、search(row,k,i)

(5) ⑤处应填()。

A、cnt[i]=0 B、cnt[i]-- C、cnt[i]++ D、cnt[i]=1

2、最大连续子段和

给出一个数列（元素个数不多于 100），数列元素均为负整数、正整数、0。请找出数列中的一个连续子数列，使得这个子数列中包含的所有元素之和最大，在和最大的前提下还要求该子数列包含的元素个数最多，并输出这个最大和以及该连续子数列中元素的个数。

```
01 #include<bits/stdc++.h>
02 using namespace std;
03 int a[101];
04 int n,ans,len,tmp,beg,end;
05 int main(){
06     cin >> n;
07     for (int i=1;i<=n;i++){
08         cin >> a[i];
09         for (int i=1;i<=n;i++){
10             if (tmp+a[i]>ans){
11                 ans=tmp+a[i];
12                 __①__
13             }
14             else if (__②__&& i-beg>len)
15                 len=i-beg;
16             if (__③__){
17                 __④__
18                 tmp=0;
19             }
20             else __⑤__
21         }
22         cout << ans << " " << len << endl;
23 }
```

(1) ①处应填()。

A、len=i+beg B、len=i-beg C、i-beg=len D、len=beg

(2) ②处应填()。

A、tmp+a[i]<=ans B、tmp<=ans C、ans>=tmp+a[i] D、ans==tmp+a[i]

(3) ③处应填()。

A、tmp+a[i]<=0 B、tmp+a[i]>=0 C、tmp+a[i]>0 D、tmp+a[i]<0

(4) ④处应填()。

A、beg=i B、beg+=i C、beg-=i D、i=beg

(5) ⑤处应填()。

A、tmp=a[i] B、tmp-=a[i] C、tmp+=a[i] D、a[i]=tmp