2020CCF非专业级别软件能力认证第一轮

（CSP-S）提高级C++语言试题卷

认证时间: 2020年10月11日9:30-11:30

考生注意事项:

●试题纸共有12页，答题纸共有1页，满分100分。请在答题纸上作答，写在试题纸上的一律无效。

●不得使用任何电子设备 (如计算器、手机、电子词典等)或查阅任何书籍资料。

一、选择题（以下共有15道题目，对于每道题目，在ABCD选项中选择正确的一项。每题2分，共30分）

1、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

2、在下列几种存储器中，访问速度最快的是（）。D

A．硬盘 B．cache C．RAM D．寄存器

3、一个圆形水池中等概率随机分布着四只鸭子，那么存在一条直径，使得鸭子全在直径一侧的概率是（）。D

A．1/16 B．1/8 C．A.1/4 D．1/2

4、应用快速排序的分治思想，可以实现一个求第K大数的程序。假定不考虑极端的最坏情况，理论上可以实现的最低的算法时间复杂度为（ ）。C

A．O (n2) B．O (n log n ) C．O (n) D． O (1)

5、建堆算法可以用递推式表示：把建堆过程想成先对左子树建堆(T(n/2))，再对右子树建堆(T(n/2))，最后对根下溯(O(lg n))，可得：

T(n) = 2\*T(n/2) + O(lg n)

求T(n)时间复杂度（ ）。C

A．O (n^2) B．O (n logn ) C．O (n) D． O (1)

6、设完全图G有n个结点，m条边，则当（）时，G中存在欧拉回路。C

A．m为奇数 B．m为偶数 C．n为奇数 D．n为偶数

7、有n个叶子的哈夫曼树的结点总数为（ ）C

A．不确定 B．2n C．2n+1 D．2n-1

8、设栈S和队列Q的初始状态为空，元素e1、e2、e3、e4、e5和e6依次进入栈S，一个元素出栈后即进入Q，若6个元素出队的序列是e2、e4、e3、e6、e5和e1，则栈S的容量至少应该是（ ）。D

A．2 B．3 C．4 D． 6

9、在单向链表中，要将s所指结点插入到p所指结点之后，其语句应为（ ）。B

A．s->next=p+1; p->next=s;

B．(\*p).next=s; (\*s).next=(\*p).next;

C．s->next=p->next; p->next=s->next;

D．s->next=p->next; p->next=s;

10、某列火车有3节旅客车厢，进站后站台上有5名旅客候车，每位旅客随意选择车厢上车，则每节车厢都有旅客上车的概率是（）。C

A．2/81 B．1/20 C．50/81 D．128/243

11、S是集合{1，2，…，2004}的子集，S中的任意两个数的差不等于4或7，S中最多含有多少个元素？（） D

A．840

B．835

C．910

D．912

12、对于不同数制之间关系的描述，正确的是（）。 D

A．任意的二进制有限小数，未必也是十进制有限小数。

B．任意的八进制有限小数，未必也是二进制有限小数。

C．任意的十六进制有限小数，不一定是十进制有限小数。

D．任意的十进制有限小数，未必也是八进制有限小数。

13、选填代码，使得程序可以实现判断一个数是否是二次幂的功能。（） D

bool is\_power\_of\_two(int n)

{

return

}

A. (n^(n >> 31)) - (n >> 31)

B. n == (n & (~n))

C. (n > 0) ? n & (n ^ (n-1)) : false;

D. (n > 0) ? (n & (n - 1)) == 0 : false;

14、下列有关图的说法不正确的是（）。 B

A.用边表示活动的网络（AOE网）的关键路径是指从源点到终点的路径长度最长的路径。

B.对于AOE网络，加速任一关键活动就能使整个工程提前完成。

C.对于AOE网络，任一关键活动延迟将导致整个工程延迟完成。

D.在AOE网络中，可能同时存在几条关键路径，称所有关键路径都需通过的有向边为桥。

15、下列关于高级语言的说法不正确的是（）。 C

A、Fortran是历史上的第一个面向科学计算的高级语言

B、Pascal和C都是编译执行的高级语言

C、BASIC是历史上的第一个支持面向对象的语言

D、编译器将高级语言程序转变为目标代码

二、阅读程序（以下共有三段程序。每段程序后面有6个题目，前4题为判断题，后2题为选择题。对于判断题，选择T代表正确，F代表错误；对于选择题，请在ABCD选项中选择正确的一项。判断题每题1分，选择题每题3分，共30分）

1、

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

2 using namespace std;

3 int arr[101];

4 int main() {

5 int n,gap;

6 cin>>n;

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

8 cin>>arr[i];

9 for (gap = n / 2; gap > 0; gap /= 2){

10 int k=0;

11 for (int i = gap; i <= n; i++){

12 for (int j= i - gap; j > 0 && arr[j] > arr[j + gap]; j -= gap){

13 swap(arr[j], arr[j + gap]);

14 k++;

15 }

16 }

17 cout<<k<<endl;

18 }

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

20 cout<<arr[i]<<" ";

21 return 0;

22 }

16、程序运行过程中，j一定小于等于n-gap。（ ） 是

17、若gap为1，20行输出结果不变。（ ） 是

18、任何时候，17行输出的k都小于n。（ ） 否

19、17行k值一定递减出现。（ ） 否

20、6个数，arr数组中数据为37 90 41 43 39 51，最后一次输出的k为（ ）。B

A．0 B．1 C．2 D．3

21、该程序时间复杂度最接近（ ）。 B

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

2、

1 struct Node {

2 ll d,id;

3 bool operator < (const Node &A) const {

4 return d > A.d);

5 }

6 };priority\_queue<Node> pq;

7 inline void f() {

8 for(int i=1; i<=n; i++) dis[i]=inf/3;

9 dis[1]=0,pq.push((Node) {0,1});

10 while(!pq.empty()) {

11 Node now=pq.top();

12 pq.pop();

13 int val=now.d,u=now.id;

14 if(dis[u]<val) continue;

15 for(int p=head[u]; p; p=nxt[p]) {

16 int v=a[p];

17 if(dis[u]+b[p]<dis[v]) {

18 dis[v]=dis[u]+b[p];

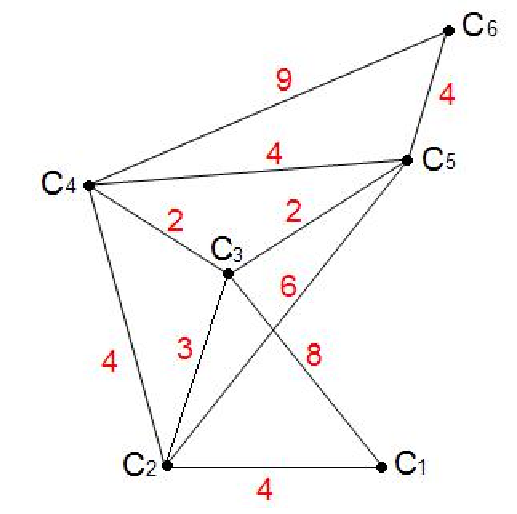
19 pq.push((Node) {dis[v],v});

20 }

21 }

22 }

23 }



注：本图仅与22、26题有关

22、对于上图，程序运行过程中，15行中的u一定不同。（ ） 是 正权情况如此

23、对于任意输入，17行b[p]的值一定不能小于0。（ ） 否

24、11行每次pop出的now其val值一定单调不降。 （ ）是

25、堆中节点不会重复。 （ ）否

26、如上图，程序运行结束dis[6] C

A．8 B．9 C．13 D．17

27、该程序时间复杂度为（ ）。 C

A．O(nlogn) B．O((n+e)logn) C．O((n+e)loge) D．O(nloge)

3、

如下代码用来计算字符串s的前缀函数。

定义前缀函数pi[i]=max(k=0-i){k:s[0…k-1]=s[i-(k-1)…i]}

简单来说pi[i]就是，子串s[0…i]最长的相等的真前缀与真后缀的长度。

例如：s[0…4]=ababc，pi[3]=2，pi[4]=0

规定：单个字符没有真前缀和真后缀，pi[0]=0

1 vector<int> prefix\_function(string s) {

2 int n = (int)s.length();

3 vector<int> pi(n);

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

5 int j = pi[i - 1];

6 while (j > 0 && s[i] != s[j]) j = pi[j - 1];

7 if (s[i] == s[j]) j++;

8 pi[i] = j;

9 }

10 return pi;

11 }

28、该算法是在线算法。（ ） 是

29、pi数组中元素值域为[0,n)。（ ） 否

30、6行中j一定单调递减。 （ ）是

31、若第7行条件成立，第8行执行完时：一定有pi[j]=pi[j-1]+1。 （ ）是

32、若s为”abcabcd”，当i为5时，j的变化是：（ ） D

A．3 B．2->0->1 C．2->0->3 D．2->3

33、该程序时间复杂度为（ ）。 A

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

三、补充程序（本大题共含有2篇代码，共10小题，每小题4分，共40分。请在每道小题后所给的四条代码中选出最恰当的一项，使这段代码填入完整程序中对应的空缺处能符合题意。）

1 【Unique MST？】给定一棵无向连通图（点数n<=100），请你判断其最小生成树是否唯一。若不唯一，输出数量，否则输出“Not Unique”。

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int INF = 0x3f3f3f3f;

const int LEN = 101;

int Map[LEN][LEN];

int n, m, top, fa[LEN], dis[LEN][LEN], tag[LEN\*LEN], vis[LEN];

struct E {

int u, v, val;

} edge[LEN\*LEN];

bool cmp(E a, E b) {

return a.val < b.val;

}

void init() {

for(int i=0; i<LEN; i++) fa[i] = i;

}

int Find(int x) {

return fa[x] == x ? x : fa[x] = ① ;

}

int kruskal() {

int cnt = 0, ret = 0;

init();

memset(tag, 0, sizeof tag);

sort(edge, edge+m, cmp);

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

int pa = Find(edge[i].u), pb = Find(edge[i].v);

if(pa == pb) continue;

fa[pa] = pb;

cnt ++;

Map[edge[i].u][edge[i].v]=Map[edge[i].v][edge[i].u]=edge[i].val;

tag[i] = 1;

ret += edge[i].val;

if( ② ) break;

}

return ret;

}

void dfs(int bg, int v, int val) {

vis[v] = 1;

dis[bg][v] = val;

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

if(Map[v][i] != INF && vis[i] == 0) dfs(bg, i, ③ );

}

}

int main() {

memset(Map, 0x3f, sizeof Map);

memset(dis, 0, sizeof dis);

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

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

scanf("%d%d%d", &edge[i].u, &edge[i].v, &edge[i].val);

int ans = kruskal();

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

memset(vis, 0, sizeof vis);

dfs(i, i, 0);

}

int f = 1;

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

if( ④ )continue;

if( ⑤ ) f = 0;

}

if(f)printf("%d\n", ans);

else printf("Not Unique!\n");

return 0;

}

34填入①处的代码是 B

A. fa[fa[x]]

B. Find(fa[x])

C. Find(Find(x))

D. 1

35填入②的代码是 C

A. cnt != n

B. cnt == 1

C. cnt == n-1

D. cnt == n

36.填入③处的代码是 A

A. max(val, Map[v][i])

B. max(val, Map[bg][i])

C. min(val, Map[v][i])

D. min(val, Map[bg][i])

37.填入④处的代码是 D

A. vis[i]

B. dis[i]

C. f==0

D. tag[i]

38.填入⑤处的代码分别是 C

A. edge[i].val == Map[edge[i].u][edge[i].v]

B. edge[i].val > Map[edge[i].u][edge[i].v]

C. edge[i].val == dis[edge[i].u][edge[i].v]

D. edge[i].val > dis[edge[i].u][edge[i].v]

2【好数】对于一个正整数X，如果把X化成二进制数后，如果X的二进制数至少有三个连续的1或者至少有3个连续的0（不能有前导0），那么X就是“好数”。例如8就是“好数”，因为8对应的二进制数是1000，有三个连续的0。整数15也是“好数”，因为15对应的二进制数是1111，也有三个连续的1。整数27就不是“好数”，因为27对应的二进制数是11011，既没有连续的三个1也没有连续三个0。

现在给出两个整数Low、UP，求Low和UP范围内有多少个“好数”。（0 <= Low <= UP <= 2147483647）

代码如下：

提示：F[i][j][k]表示从右往左数第i位的数字为j，第i-1位为k的“坏数”总数

#include<bits/stdc++.h>

#define ll long long

using namespace std;

ll F[125][2][2];

ll Ans,Sum,len,k,n,m;

string sm;

ll Change(ll t) {

if(t<=0)return 0;

Ans=0,k=t,len=0;

sm="";

while(k) { //求t的二进制

char s=k%2+48;

sm=s+sm,k/=2;

}

len=sm.length();

for(int i=2; i<len; ++i) //求t的二进制长度-1位的坏数

Ans+= ① ;

if(t>0) ② ;

k=0;//求t的二进制长度位的坏数

while(++k<=len)

if(sm[k]=='1') {

sm[k]='0';

int l=1;

char r=sm[0];

bool B=0;

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

if( ③ ) {

l++;

if(l==3) {

B=1;

break;

};

} else l=1,r=sm[i];

if(B==0)

Ans+= ④ ;

sm[k]='1';

}

int l=1;

char r=sm[1];

bool B=0;

for(int i=2; i<=len; ++i)

if( ⑤ ) {

l++;

if(l==3) {

B=1;

break;

};

} else l=1,r=sm[i];

if(!B)Ans++;

return t-Ans;

}

int main() {

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

F[2][1][0]=F[2][1][1]=F[2][0][0]=F[2][0][1]=1;//初值

for(int i=3; i<=31; ++i) {

F[i][0][0]=F[i-1][0][1];

F[i][1][0]= ⑥;

F[i][1][1]=F[i-1][1][0];

F[i][0][1]= ⑦;

}

Sum=Change(m)-Change(n-1);//前缀和

printf("%d",Sum);

return 0;

}

39.填入①处的代码是 C

A. F[i-1][1][0]+F[i-1][1][1]

B. F[i-1][0][1]+F[i-1][1][1]

C. F[i][1][0]+F[i][1][1]

D. F[i][0][1]+F[i][1][1]

40.填入②的代码是 B

A.len++

B. Ans++

C. F[0][0][0]

D. F[1][0][0]

41.填入③⑤处的代码是 A

A. sm[i]==r

B. sm[i]==’0’

C. sm[i]==’1’

D. sm[i]==sm[k]

42.填入④处的代码是 D

A. F[len-k][sm[k]-48][0]

B. F[len-k][sm[k-1]-48][0]

C. F[len-k+1][sm[k]-48][0]

D. F[len-k+1][sm[k-1]-48][0]

43.填入⑥和⑦处的代码分别是 D

A. F[i-1][1][1]；F[i-1][0][1]

B. F[i-1][0][1]；F[i-1][1][1]

C. F[i-1][1][1]+F[i-1][0][1]；F[i-1][0][1]+F[i-1][1][0]

D. F[i-1][0][0]+F[i-1][0][1]；F[i-1][1][1]+F[i-1][1][0]